A Compositional Semantic Structure for Multi-Agent Systems Dynamics

SIKS Dissertation Series No. 2001-8.

The research reported in this thesis has been carried out under the auspices of SIKS, the Dutch Graduate School for Information and Knowledge Systems.

Printed and bound in The Netherlands by PrintPartners Ipskamp.

NUGI: 851
ISBN: 90-9014746-2

VRIJE UNIVERSITEIT

A COMPOSITIONAL SEMANTIC STRUCTURE FOR MULTI-AGENT SYSTEMS DYNAMICS

ACADEMISCH PROEFSCHRIFT

ter verkrijging van de graad van doctor aan
de Vrije Universiteit Amsterdam,
op gezag van de rector magnificus
prof.dr. T. Sminia,
in het openbaar te verdedigen
ten overstaan van de promotiecommissie
van de faculteit der Exacte Wetenschappen / Wiskunde en Informatica
op dinsdag 12 juni 2001 om 13.45 uur
in het hoofdgebouw van de universiteit,
De Boelelaan 1105

door

Pascal Antonius Theodorus van Eck

geboren te Beneden-Leeuwen

Promotoren: prof.dr. F.M.T. Brazier
prof.dr. J. Treur

Acknowledgements

First and foremost, I would like to thank Frances Brazier and Jan Treur, my supervisors, for their support and confidence during our co-operation, which started in the Autumn of 1994 with the supervision of my Master's thesis. Their availability was incredible: it has been as if I was their only Ph.D. candidate. However, in addition to me, a number of Ph.D. candidates, M.Sc. students, and other colleagues co-operated with Jan and Frances, all of them handing in 'concepts', 'preliminary drafts' and 'versions 0.1', and were equally eager to have their text reviewed as soon as possible (preferably the same day).

Playing a major role in the AI curriculum and with a strong focus on long-running, multi-person research projects, the Department of AI of the Vrije Universiteit is a team in every sense. I would like to especially thank my office mates Joeri Engelfriet, and later, Wouter Wijngaards, for their support. With Niek Wijngaards, I never actually shared an office, but if virtual offices exist (they do), we definitely shared one. With Niek, Wouter, FJ Jüngen, Pieter van Langen, Mark Sloof, and Wieke de Vries, fellow Ph.D. candidates, I had many good discussions on multi-agent systems, their advantages, and, most notably, the associated challenges. The same holds, to various extents, for Mehdi Dastani, Catholijn Jonker, Henry Prakken, Leon van der Torre, Rineke Verbrugge, and Mark Willems. Lourens van der Meij and Frank Cornelissen were always available for technical questions concerning the details of DESIRE's implementation.

In my opinion, much research in AI requires close co-operation between the members of a research team. Only in this way, substantial progress can be made. In such teams, each researcher has to stand for his or her own ideas. (My own hobbyhorse was the fight against global state and time. See Chapter 7 for why (i) global time is a bad idea and (ii) it isn't needed, to begin with. I guess my colleagues avoided all words beginning with 'gl' in relation to time in general and semantics in particular, just to be sure.) However, at the same time, for the co-operation to succeed, all participants have to trust the work of others in the project. A good mutual understanding is a pre-condition, and in the Department of AI, this precondition was fully satisfied.

With Joeri, Frank van Harmelen, Mark Willems, and two persons from other research groups at the time, Dieter Fensel and Yde Venema, I embarked on an

important endeavour not directly related to my thesis: the 'dynamics reading group', which actually turned into a writing group. I learned a lot from participating.

The postman dropped the close to 400 pages of my 'final draft' of this thesis at the front door of the members of my reading committee a few days before Christmas, 2000, most likely depriving them of their holidays. I thank Henri Bal, Frank de Boer, Fausto Giunchiglia, Jan-Willem Klop and John-Jules Meyer for their willingness to serve on the reading committee and their prompt replies to my 'final draft'.

Between 1995 and 2000, I was enrolled in SIKS, the National Graduate School for Knowledge and Information Systems. Chances are that no other Ph.D. student has ever followed as many SIKS courses as I did. I encourage all SIKS Ph.D. candidates to break this record! As their best customer (or, probably, most expensive student), I considered it important to share my thoughts about SIKS's policy with its management board. I would like to thank the scientific director, John-Jules Meyer, and the co-ordinators Koen Versmissen and Richard Starmans, for taking my comments very seriously and acting as a proxy for the Board.

This thesis was completed after I moved to the University of Twente. I would like to thank the University of Twente, especially Roel Wieringa, for the facilities and especially the confidence in the completion of this thesis, and my colleagues for their support.

Apart from colleagues and friends in the research community, many other people have had a positive influence on the quality of life during this period of my life. My Greek friends and the International Voluntary Service Organisation SIW formed, more often than they probably can imagine, a source of inspiration. The people from the university sports facility ASVU and especially from the unofficial, unnamed, unknown, but undoubtedly indispensable running group that grew out of it proved, on average twice a week, the paradox of human physiology: you can charge your batteries by draining them. If all goes well, two representatives, one from SIW and one from the running group, will assist me until the very end of the job.

Last, but not least, I would like to thank my parents and sister, who showed interest, confidence and support, I guess eternally if that's what it takes.

Enschede, April 2001,
Pascal van Eck.

Table of Contents

Chapter 1

Introduction

This thesis contributes to the field of multi-agent systems research. In this field, software systems are analysed and designed as if they are societies of autonomous, rational actors or *agents*. Section 1.1 defines multi-agent systems and discusses the notion of an agent. Section 1.2 presents some potential benefits of multi-agent systems. Section 1.3 discusses a number of issues that are encountered in the current state of the art. Section 1.4 describes the specific research aim of this thesis. Section 1.5 presents an outline of this thesis.

1.1 Multi-Agent Systems

A multi-agent system is defined as a system consisting of a number of agents that share a common environment. The definition of an agent is more involved. During the past years, quite a few researchers have attempted to define the notion of an agent (see (Franklin & Graesser, 1997) for an overview). The most influential definition is probably the 'weak notion of agency' defined by Wooldridge and Jennings (1995b). This notion is presented in Section 1.1.1 below. Wooldridge and Jennings have also defined, in the same paper, a stronger notion of agency, which is presented in Section 1.1.2. A perspective on the various notions of agency is presented in Section 1.1.3.

1.1.1 A Weak Notion of Agency

In their influential paper, Wooldridge and Jennings (1995b) propose two notions of agency, called the weak notion and the strong notion of agency. The weak notion of agency defines an agent as a hardware- or software-based computer system that is *autonomous*, *reactive*, *pro-active* and has *social ability*. The most important property is autonomy: an agent is able to set its own goals and to choose a way to achieve those goals. Autonomy is only of interest for *situated* agents: agents that are situated in an environment, possibly shared with other agents, and that are able to observe their environment and carry out actions in the environment. (Wooldridge

recently[1] refined the weak notion of agency by defining an agent as an *autonomous, embodied* hardware or software system, with *reactivity*, *pro-activeness* and *social ability* as additional properties.) The properties reactivity, pro-activeness and social ability together determine the *flexibility* of an agent:

- Reactivity of an agent refers to its ability to react, presumably in a sensible way, to unexpected situations arising in its environment;
- An agent is, by its autonomy, not only able to set its own goals, its pro-activeness ensures that it will actually do so. Thus, an agent *takes the initiative* and creates opportunities to pursue its goals instead of merely reacting to its environment;
- An agent's social ability enables it to co-operate with other agents in its environment. Co-operation is almost always needed to achieve the goals an agent has set.

The weak notion of agency of Wooldridge and Jennings raises a number of interesting issues:

- As the Latin stem of the word 'agent' (*agere*, to do) indicates, it is essential that an agent is an *active* software or hardware system. Dictionary definitions of an agent often put most emphasis on this property. (E.g., an agent is 'someone or something that acts'. Emphasis is also put on the social ability of an agent by indicating that an agent is 'someone or something that acts on behalf of someone else').
- The adjective 'active' might appear to be superfluous, especially for software systems. This adjective, however, emphasises that agents are *not* merely data structures. Objects in object-oriented development are often seen as active data structures: operations on data are encapsulated with the data itself. However, not every object is an agent. The distinction between objects and agents is best formulated by Wooldridge (1999) as follows. An *object* has no control over the execution of the operations it encapsulates (usually called its *methods* in object-oriented development). E.g., if object 1 invokes one of object 2's operations, this operation is executed regardless of the state of object 2. An *agent*, by definition (an agent is autonomous), has complete control over its own actions. Other agents may transfer requests to perform specific actions to an agent, comparable to operation invocation in object-oriented systems. The agent itself, however, decides whether the request will be granted. A rational agent bases this decision on an evaluation of the relation between the requests and its own goals, commitments and agenda.
- Wooldridge and Jennings do *not* require an agent to be mobile. Especially in the popular press, agents are depicted as software components that roam the

[1] In a lecture delivered at the SIKS Annual Meeting in Amsterdam, November 1, 1999.

Internet, collecting information for their dispatchers on the go. According to this concept of agency, mobility is essential. However, the notion of autonomous, embodied (software) components, or agents for short, can be applied to non-mobile systems as well, and this is also beneficial. Moreover, the primary focus of much research in the field of mobile agents is on implementation aspects of mobile agents, such as transportation, security, operating system support, and APIs (Application Programmer's Interfaces). Although the properties of agency distinguished by Wooldridge and Jennings are often mentioned (especially autonomy), the question how, on the one hand, mobility, and, on the other hand, autonomy, reactivity, pro-activeness and social ability interact is not addressed. Such research (Gray, Cybenko, Kotz & Rus, 1997; Johansen, Renesse & Schneider, 1997) is closer to research in the fields of object orientation or component based development than to research in multi-agent systems.

- It goes without saying that the weak notion of agency applies to human beings to a large extent: humans are generally assumed to be autonomous, situated in their society and socially able. This is of interest for agent-based approaches to requirements engineering: the same concepts can be used both for the analysis of (existing) human procedures and the analysis of automated systems consisting of hardware or software agents.

1.1.2 A Strong Notion of Agency

Wooldridge and Jennings (1995b) also define a stronger, more specific notion of agency. Literally, this stronger notion is defined as follows:

> "… a computer system, that, in addition to having the properties identified above [autonomy, reactivity, pro-activeness and social ability, PvE], is either conceptualised or implemented using concepts that are more usually applied to humans. For example, it is quite common in AI to characterise an agent using *mentalistic* notions, such as knowledge, belief, intention, and obligation (Shoham, 1993)."

Under the strong notion of agency, an agent is ascribed a mental state, consisting of e.g. belief, knowledge, intentions, goals, commitments and obligations. The behaviour of the agent is described using these mentalistic notions.

According to McCarthy (1987), the use of notions more usually applied to humans is both legitimate and often useful. It is legitimate if these notions express the same information for both humans and agents. Seel (1989) argues that this is possible for all kinds of automata. Using notions normally used for humans is useful in the sense that it provides a high-level description of the behaviour of an agent, instead of a description in terms of the actual mechanisms with which its behaviour is generated. Often, a description in terms of mentalistic notion is the only description available, or the only description simple enough to be

comprehended. As an aside, the strong notion of agency only plays a minor role in this thesis.

1.1.3 Other Notions of Agency

The notions of agency developed by Wooldridge and Jennings are not without criticism. In fact, almost every introductory text on multi-agent systems proposes a new definition for the notion of an agent. Franklin and Graesser (1997) survey ten alternative definitions found in the agent literature and then propose their own definition:

> "An autonomous agent is a system situated within and a part of an environment that senses that environment and acts on it, over time, in pursuit of its own agenda and so as to effect what it senses in the future."

Franklin and Graesser thus emphasise situatedness and continuity of an agent, and require that there is some kind of feedback: the actions of an agent should affect future observations of the agent. Both continuity as well as the feedback requirement help to distinguish agents from, in Franklin's and Graesser's terms, 'ordinary programs'.

The lack of one single accepted definition of the concept of an agent is a source of much criticism from other research areas. It is not uncommon that central notions in a newly established research field are subject to debate within the field for a long time. Often, valuable research results are obtained as a side effect of these debates. However, such debates should not hide the central focus of the field of multi-agent systems: the creation of software systems that are designed for flexibility. An agent is autonomous and is able to devise its own goals and pursue them in a rational way, all by itself. Unexpected events in its environment should not hinder the agent in the completion of its tasks. The central issue in the field of multi-agent systems research is to answer the question how software with this level of flexibility can be engineered.

1.2 The Promise of Multi-Agent Systems

In the field of multi-agent systems research, software systems are analysed and designed as if they are societies of autonomous agents. The benefits of this focus are twofold.

First, autonomous agents are a natural metaphor for today's complex, networked software systems. In such systems, there is no central authority to which the software components are enslaved. In other words, such components are autonomous. Network connections between the components enable them to co-operate and to act upon other components, but the precise behaviour of other components is beyond their control. The subject of multi-agent systems research precisely matches this context: autonomous agents that are situated in an environment in which they may influence, and are influenced themselves, by other

agents. This metaphor naturally leads to decentralised systems, which, together with the flexibility of an agent—its reactivity, pro-activeness and social ability—is needed for robustness: the possibilities to cope with unexpected events in the environment. It is possible to design decentralised, robust systems using conventional approaches. However, in conventional approaches, there is no inherent support for agent-like properties. In agent-oriented approaches, such support is the norm.

Second, analysing and designing software systems as if they are societies of autonomous agents paves the way for the application of theories from the Social Sciences to software systems. A multi-agent system is a system that consists of a number of agents that share a common environment. Just like humans and their society, agents both have the potential to mutually benefit from co-operation and are constrained by interdependence. The Social Sciences (e.g., Economics, Management Science, Sociology, Political Science, and Social Psychology) investigate how social functions (i.e., specific behaviour of a society as a whole, such as, for instance, the occurrence of a pareto-optimal market equilibrium or a traffic jam) relate to the individual behaviour of (human) agents. Results obtained in the Social Sciences can be used to support the design of multi-agent systems, in which the overall functionality of the multi-agent system emerges from the individual behaviour of rational agents that pursue their own goals.

On the one hand, solutions inspired by the Social Sciences are implemented in current-day multi-agent systems, the most well-know example being the contract net protocol for delegation of tasks (Smith, 1980). On the other hand, it seems that the Social Sciences themselves currently do not possess the definite answer to the question of how social functions emerge from the behaviour of individual agents. Moreover, it may be the case that, to obtain such an answer, the Social Sciences need to rely on advances in the area of multi-agent systems. The relation between the Social Sciences and multi-agent systems research thus seems to be a beneficial one for both areas (Eck, 1998).

From an Artificial Intelligence point of view, the two promises together offer a new level of abstraction for communication and co-ordination in knowledge-intensive distributed systems comparable to the knowledge level (Newell, 1984): the social level. The Social Sciences may become an additional source of inspiration for Artificial Intelligence, complementing Psychology (Cognitive Science).

1.3 The Practise of Multi-Agent Systems

Practitioners in the area of multi-agent systems research currently face a number of issues that need to be solved:

- First of all, the notion of an agent is used both as a metaphor and as a reference to a type of technology. This is a source of much confusion in the field of multi-agent systems, and of even more confusion in related research areas. In the field of agent technology, the main focus is on the development

of software components that are agents. Such software components are programmed to provide autonomy, reactivity, pro-activity, social ability and (often) mobility. Software engineers who employ agent-based technology only have to refine the components by adding task-specific functionality. Agent technology is best viewed as an extension of object technology. Much research in the area of multi-agent systems, however, is not directly concerned with the development of software components that are agents. Instead, the notion of an agent is used as a metaphor: software systems are analysed and designed as if they consist of autonomous, rational agents. Research topics include the development of theories about the behaviour of such agents, about behaviour that emerges from the interplay between agents in a multi-agent system, and generic architectures for such agents. However, there is no commitment to a specific technology for implementing multi-agent systems, although, of course agent technology is often the most promising candidate.

- Currently, there is no solution to the problem of how the behaviour of a complex multi-agent system can be predicted based on the goals of individual agents. In terms of the Social Sciences, there is no *middle ground theory* (Castelfranchi & Conte, 1996). The promise that multi-agent systems research will enable the Social Sciences to develop such a theory has not yet been met.
- As stated in Section 1.1.1, autonomy is an essential property of an agent. However, currently there is no philosophically satisfactory definition of autonomy. (See (Dennett, 1984) for a contemporary philosophical treatment of autonomy.) The field of multi-agent systems research is often criticised for its reliance on such a vague notion. For practitioners, it is difficult to determine if a specific software component is an agent, or "just a program" (Franklin & Graesser, 1997).
- A multi-agent system consists of agents that are active simultaneously. Consequently, the analysis and design of multi-agent system also involves the analysis and design of concurrent behaviour. The central focus in the area of concurrent behaviour is how to relate the different time scales (or clocks, or, in other words, behaviour) of the different simultaneous processes. In multi-agent systems, this problem has to be solved without making (implicit or explicit) assumptions about synchronicity that are not compatible with the autonomy of agents. As explained in the next section, this issue is closely related to the research aim of this thesis.
- Only recently (Jennings, 1999) has the multi-agent systems research community realised that the development of a multi-agent system not only involves state-of-the-art Artificial Intelligence and results in agent theory, but also complex software engineering. Software engineering aspects of

multi-agent system development have therefore only recently been addressed, and results are preliminary.

Wooldridge and Jennings (1998) present a list of many more, as they call it, pitfalls of agent-oriented software development.

This thesis develops an approach for the representation of the behaviour of agents and of multi-agent systems in a precise, mathematical way. This approach facilitates the development of solutions for the issues listed above.

1.4 Research Aims, Method, Relevance, Context and Demarcation

This section discusses the aims of the research presented in this thesis, as well as the relevance of the research aims and the method followed. The context in which the research is placed is briefly discussed and a demarcation of the performed research is sketched.

1.4.1 Research Aims

The research aim of this thesis is to develop a *formal, compositional, semantic structure for multi-agent systems dynamics*. This research aim is characterised in more detail as follows:

- As indicated at the beginning of this chapter, agents are first and foremost active entities. Therefore, the primary characteristic of a multi-agent system is that different and simultaneous processes take place. These processes result in a continuous change of properties of the agents in a multi-agent system. The focus of this thesis is on these continuous processes of change, or, in other words, on the *dynamics* of a multi-agent system. The activity of a multi-agent system or of an individual agent is also called the *behaviour* of the multi-agent system or agent. In this thesis, behaviour, activity and dynamics are used as synonyms.
- Design of a multi-agent system is based on a model in which different aspects of the multi-agent system, e.g. the behaviour, knowledge, intentions and goals of the agents, are explicitly distinguished. Which aspects of a multi-agent system are covered by a model depend on a number of factors, such as, for instance, the intended usage of the model and the level of abstraction. A model of a multi-agent system is itself intangible. However, a model can be given a tangible form by a description, or *specification*, of the model. A description of aspects of the dynamics of a multi-agent system is called a specification of multi-agent systems dynamics.
- A specification is an expression formulated in a (natural or formal) language, called the specification language. As such, a specification itself is merely a syntactic entity, or, in other words, a string of symbols. The raison d'être of the specification is its *intended meaning*: the model it denotes. Thus,

the need arises to express the intended meaning of a specification. A *semantic structure* provides a set of constructs and relations between these constructs with which the intended meaning of a specification can be defined.

- The semantic structure is identified as a *compositional* semantic structure, which indicates the two most important properties of the semantic structure. First, the primary construct in the semantic structure is the *component:* the basic building block for models of multi-agent systems. In this thesis, multi-agent systems are assumed to be modelled as *compositional systems*. Second, the semantic structure supports the principle of *compositionality*: the dynamics of a system composed of a number of components is defined by a composition relation in terms of the dynamics of these components.
- The semantic structure is also identified as a *formal* semantic structure. The constructs provided by the structure to define the meaning of a specification are mathematical constructs. The use of mathematical constructs enables a precise and unambiguous definition of the intended meaning of a specification.

To summarise, the primary research aim of this thesis is *to develop a set of mathematical constructs and relations between these constructs that can be used to define the intended meaning of a specification for a multi-agent system*. Such a set of constructs is called a semantic structure. As a specification is an expression formulated in a specific language (the *specification language*), a semantic structure can also be used to define the intended meaning (semantics) of the specification language. The semantics of a language, by definition, consists of the semantics of expressions in that language: the semantic structure serves as the *semantic domain* of the specification language.

To illustrate possible applications of the semantic structure, in this thesis, two example multi-agent systems are modelled. First, a model for co-ordinating exclusive access to resources in a multi-agent system is presented in Chapter 10. Second, in Chapter 11, a society of a relatively large number of relatively simple agents is modelled, with the purpose of studying emergent social behaviour.

1.4.2 Method

The method employed in this thesis to develop the semantic structure consists of the following steps:

- The starting point for the development of a semantic structure consists of a commitment to a set of constructs and relations between these constructs. Together, the set of constructs and relations constitute the semantic structure. To establish such a commitment, different possibilities for specific sets of constructs and relations are identified. This commitment is made by the developers of the semantic structure and is fixed for all applications of

the semantic structure. The primary commitment in this thesis is the commitment to components as the basic construct. This commitment supports the basic assumption mentioned above: multi-agent systems are modelled as compositional systems.

- The specific set of constructs and relations between these constructs, which is fixed for the semantic structure, is designed to be as general as possible. As a consequence, for each application of the semantic structure, a number of additional choices have to be made that determine how the semantic structure is applied for the specific application. These choices are made by a user of the semantic structure and are fixed for each specific application. After establishing the set of constructs and relations between these constructs, this thesis explores different choices for applications of the semantic structure.
- The set of constructs, properties and relations is described in a detailed, mathematical way. (This constitutes the main part of this thesis.)
- The semantic structure is applied to provide a semantic domain for the compositional multi-agent systems development method DESIRE. (See (Brazier, Jonker & Treur, 1998) for an overview of the principles behind DESIRE. A generic agent model modelled in DESIRE is described in (Brazier, Jonker & Treur, 2000). The generic agent model has been applied in many domains including electricity transportation management (Brazier, Dunin-Keplicz, Jennings & Treur, 1997), electricity load balance management (Brazier, Cornelissen, Gustavsson, Jonker, Lindeberg, Polak & Treur, 2000) and as a basis for the co-operative agent model which has been applied, for example, in distributed call centre support (Brazier, Cornelissen, Jonker & Treur, 2000). An earlier version of DESIRE is described in (Langevelde, Philips & Treur, 1992)).
- The last step consists of the analysis of two example multi-agent systems modelled in DESIRE. The first system multi-agent system consists of agents that have to co-operate to obtain mutually exclusive access to a shared resource. The second multi-agent system represents a society of 30 simple agents used for experimental Social Science research.

1.4.3 Relevance of the Research Aim

As explained above, dynamics are an important aspect of every multi-agent system. The dynamics of a multi-agent system emerge from complex mutual influence between concurrent processes in a system. Wooldridge (1996, Section 5) acknowledges the importance of concurrency in multi-agent systems as follows: "The issue of concurrency has scarcely been addressed in formal treatments of DAI, and yet concurrency is at the very heart of the area."

The formal representation of concurrency has long been studied in mainstream Theoretical Computer Science. However, the results obtained are not directly applicable to concurrency in multi-agent systems due to several specific characteristics of multi-agent systems and the individual agents of which a multi-agent system is constituted. These special characteristics can be described in the form of requirements for a semantic structure for multi-agent systems dynamics as follows:

- A multi-agent system consists of a number of agents and the environment shared by these agents. The agents deliberately and mutually influence one another by executing actions in the environment, observing the environment, and by communicating with one another. Communication, action execution, and observation are together called interaction. Individual agents have to deliberate to evaluate information received from other agents and to determine which actions to execute and information to exchange to pursue their own goals. The dynamics of a multi-agent system thus consists of *deliberation and interaction.* A semantic structure for multi-agent systems dynamics should therefore provide a natural representation of deliberation and interaction.
- An important phenomenon of multi-agent systems is that a multi-agent system is composed of a number of heterogeneous agents: e.g., agents may have completely different structure and they may perform different tasks, either collaboratively or in competition. A multi-agent system is often not designed by a single designer. In general, agents in a multi-agent system are themselves complex systems consisting of a number of components, such as e.g. planning components and knowledge base components. Analysis and design of such heterogeneous systems is greatly improved by the *compositionality principle*. A semantic structure for agent dynamics should support this compositionality principle.
- In, for instance, the strong notion of agency put forward by Wooldridge and Jennings mentioned at the beginning of this chapter, an agent is characterised as an (active, computational) entity whose *state* is described by mentalistic notions. This characterisation suggests that the dynamics of multi-agent systems should be modelled in terms of *states and state transformations*. This view may be contrasted with a number of common semantic structures for dynamics in mainstream Theoretical Computer Science, where dynamics is often modelled as partial or total orders of named actions[2] (also called events), without an explicit notion of state and

[2] There seems to be a difference between the notion of action in Theoretical Computer Science and Multi-Agent Systems. In Theoretical Computer Science, any observable activity of an agent is called an action, whether or not this activity influences any other part of the

state transformations. (Nevertheless, Burkhard (1993) chooses to use events as the basis for his semantic structure for multi-agent systems.) In this thesis, the dynamics of a multi-agent system are defined in terms of state and state transformation.

- The semantic structure should model dynamics as a composition of *local* behaviours of agents and agent components, together with interrelationships between these local behaviours. This *locality* is motivated by the following two phenomena. In the first place, as already mentioned, most multi-agent systems consist of a heterogeneous collection of agents. Therefore, a global view is difficult to obtain, while a collection of local views with interrelationships is more natural. In the second place, the agents in a multi-agent system are often widely separated. Due to this wide-area distribution, communication is asynchronous and cannot be assumed to be durationless. Under these conditions, as explained in Chapter 7, global time is not available. Consequently, if a notion of global state is desired, it has to be defined such that the existence of global time is not implied by the definition. In this thesis, a notion of global state is defined using dependence relations between local states (a form of locality). Most common semantic structures for dynamics in Theoretical Computer Science (often implicitly) assume that global time is available and thus do not support this form of locality.

The use of dependence relations to define a notion of global state is inspired by the *true concurrency* view on modelling distributed systems (see among others Pratt, 1986; Schwarz & Mattern, 1994), as opposed to the *atomic mutual exclusion*, or *interleaving*, view. Several authors (e.g., Pratt 1986) argue that truly concurrent processes are not properly modelled in the atomic mutual exclusion view, and point at disadvantages of this view. In the atomic mutual exclusion view, concurrency is modelled as nondeterministic global choice between different sequences of atomic actions of individual processors in a distributed system. The atomic mutual exclusion view implicitly assumes that a notion of global time is available, i.e., the notion of time implied by the (global) sequences of atomic actions. As concurrency is modelled as nondeterministic choice, made by a (sequential) global automaton representing an entire distributed system, the atomic mutual exclusion view does not support the principle of locality. Moreover, in the atomic mutual exclusion view, the behaviour of a multi-agent system may possibly change if actions are refined. (In the atomic mutual exclusion view, the sequences of atomic actions are not preserved under action refinement (Castellano *et al.*, 1987; Glabbeek & Goltz, 1989). For example, consider two actions a and b that are performed concurrently. Seen as atomic actions, in the interleaving view either

---

system. However, in the multi-agent systems discipline, usually only agent activities that attempt to change the state of the environment are called actions.

a preceeds b or b preceeds a. However, if a is refined to the sequence of actions a_1a_2, a possible behaviour is a_1ba_2: neither a_1a_2 preceeds b nor the other way around.) Consequently, when applying the atomic mutual exclusion view, the behaviour of a multi-agent system always needs to be described at the most detailed level, or it has to be accepted that behaviour is not preserved if later on, a more detailed description is developed. The true concurrency view does not have these drawbacks.

The relevance of the research aim can also be indicated by reference to the literature. Wooldridge (1996, Section 5) acknowledges the special position of multi-agent systems (which he refers to as Distributed AI or DAI systems): "… DAI systems are *not* simply concurrent systems." The development of a formal semantic structure specifically designed for multi-agent systems dynamics should help to remedy this situation.

1.4.4 Research Context

The research presented in this thesis is related to three disciplines: Artificial Intelligence, Theoretical Computer Science and Multi-Agent Systems. More specifically, the relations between these disciplines and the development of a semantic structure for multi-agent systems dynamics is as follows:

- In Artificial Intelligence, locality is identified as a concept in modelling common-sense contextual reasoning, as witnessed by e.g. (Giunchiglia, 1993).
- Theoretical Computer Science is used as a source of concepts for the representation of concurrency, e.g. (Lamport, 1978, 1986; Schwarz & Mattern, 1994). As an aside, the work of Lamport, Schwarz and Mattern referenced can also be classified in the area of Distributed Systems research.
- In the area of Multi-Agent Systems, a number of formal specification languages exist that have been specifically designed for this area. In Chapter 12, a comparison with a number of alternative approaches is provided. Surveys of this area have been published by for example Wooldridge and Jennings (1995b) and O'Hare and Jennings (1996).

1.4.5 Demarcation

With respect to the demarcation of the research presented in this thesis, please note:

- This thesis approaches the subject of multi-agent systems from an Artificial Intelligence point of view. Artificial Intelligence is an engineering discipline: the focus is on *designing* systems that exhibit intelligent behaviour. Consequently, the subject of multi-agent systems is approached in this thesis with the ultimate goal of contributing to the development of engineering

methods that support the design of multi-agent systems. The thesis is not pursuing to contribute to the development of Sociology or other branches of the Social Sciences. However, the semantic structure developed in this thesis can be used to model multi-agent systems that simulate social phenomena. An example is provided in Chapter 11.

- The semantic structure developed in this thesis is not biased towards specific processes in a multi-agent system, such as e.g. goal adaptation or emergence of norms. Instead, the semantic structure is general: it is applicable to all processes in a multi-agent system. Specific processes such as goal adaptation and emergence of norms may be studied in the context of applications of the semantic structure. The multi-agent systems analysed in Chapter 10 and Chapter 11 are examples of such studies.
- As stated in Section 1.4.3, the semantic structure is state-based, which conforms to the common approach of characterising agents in terms of their (mentalistic) state. However, the semantic structure abstracts from the contents of these states. In other words, questions of which (probably mentalistic) concepts to use to describe an agent's state are not addressed in this thesis.
- The semantic structure developed in this thesis can be compared to formalisms developed to study concurrency. The semantic structure developed in this thesis is specifically designed for the domain of dynamics of multi-agent systems. More general applicability of the semantic structure is not investigated.
- Chapter 9 presents a formal language (based on temporal logic) for precisely describing the intended dynamics of components. However, the thesis does not aim at developing a logic for reasoning about (the dynamics of) multi-agent systems. As a consequence, complexity, expressive power and inference relations of the language presented in Chapter 9 are not discussed in this thesis.

1.5 Outline of the Thesis

The structure of this thesis is as follows:

Chapter 2 introduces compositional systems and presents a number of commitments with respect to specific properties of compositional systems that can be represented by the semantic structure.

Chapter 3 discusses how multi-agent systems can be represented as compositional systems. To apply the semantic structure in the area of multi-agent systems (which is the main application of the semantic structure), it is necessary to represent a

multi-agent system as a compositional system, because compositions of components are the primary construct in the semantic structure.

Chapter 4 provides an overview of the semantic structure and introduces an example that is used in Chapter 5 to Chapter 9. The overview of the semantic structure summarises the central principles and the formal techniques employed in the semantic structure.

Chapter 5 formally defines the main constructs in the semantic structure. First, components and information transmission are formally described. After that, three views on the behaviour of a compositional system are defined, and relations between these views are discussed.

Chapter 6 discusses how different properties of information transmission, such as lossless information transmission and order-preserving information transmission, are formally represented in the semantic structure. The commitments made in Chapter 2 are formally defined in this chapter.

Chapter 7 develops a notion of a global state of a compositional system. This notion of global state does not assume that there is a global notion of time accessible to all components. Moreover, a global state is derived from the three views on the behaviour of a compositional system, which consist of local states. Thus, in the semantic structure, global states are derived from local states, instead of the other way around.

Chapter 8 develops additional constructs in the semantic structure for the representation of control. In almost every multi-agent system, some agents exercise control over other agents. The semantic structure presented in Chapter 5 and Chapter 6 enables the representation of control. Chapter 8 develops a refinement of the semantic structure that enables the representation of control in a separated, domain-independent way that supports reuse and maintainability.

Chapter 9 shows an application of the semantic structure, in which semantics are developed for the DESIRE modelling framework.

Chapter 10 presents the first example of a multi-agent system modelled with the DESIRE modelling framework. The multi-agent system in this example consists of agents that have to obtain mutually exclusive access to a shared resource.

Chapter 11 presents the second example of a multi-agent system modelled with the DESIRE modelling framework. The multi-agent system in this example represents a society of 30 simple agents. This society is used to replicate experimental Social Science research (Cesta, Miceli & Rizzo, 1996a).

Chapter 12 compares the semantic structure with a number of similar approaches. Additionally, conclusions and directions for further research are presented.

Discussions presented in Chapter 2, Chapter 8 and Chapter 9 of commitments made in the semantic structure, as well as alternatives to these commitments and

the modelling choices presented in Chapter 3 and Chapter 8, are adapted from (Brazier, Eck & Treur, 1996). A strongly abridged version of Chapters 4 to 7 can be found in (Brazier, Eck & Treur, 2001b). Details of the DESIRE modelling framework presented in Chapter 9 are taken from various publications of the DESIRE group at the Vrije Universiteit, Amsterdam. Figure 9.1 was designed by Niek Wijngaards. Chapter 10 is an extended version of (Brazier, Eck & Treur, 1997b). Chapter 11 is an extended version of (Brazier, Eck & Treur, 2001a), which also appeared in a much shorter version as (Brazier, Eck & Treur, 1997a). Figure 11.6 in Chapter 11 is taken from (Cesta, Micelli & Rizzo, 1996a). The author thanks Amadeo Cesta, Maria Miceli and Paola Rizzo for providing the raw data that is used in the experimentation. The description of OSL in Chapter 12 is taken from (Eck, Engelfriet, Fensel, Harmelen, Venema & Willems, in press), which also appeared in a much shorter version as (Eck, Engelfriet, Fensel, Harmelen, Venema & Willems, 1998).

Chapter 2

Compositional Systems

The aim of this thesis is to develop a formal, compositional semantic structure for multi-agent systems dynamics. The semantic structure consists of a set of (mathematically defined) constructs together with relations between the constructs. The first step in the development of the semantic structure consists of the identification of possible commitments to a specific set of constructs and relations. Such commitments are fixed for the semantic structure and are made by its designers. After identification of possible commitments, in this chapter a specific set of commitments is fixed for the rest of this thesis. The next chapter discusses various ways in which multi-agent systems can be modelled within the context of the commitments established in this chapter.

The basic assumption adopted in this thesis is that multi-agent systems are modelled as compositional systems. Thus, in applications of the semantic structure, the dynamics of a multi-agent system are described in terms of the dynamics of a number of components that together form a compositional system. The primary commitment in the development of the semantic structure is the commitment to components as basic constructs. In other words, the semantic structure developed in this thesis consists of, in the first place, components. However, additional commitments have to be made to further establish the semantic structure. The additional commitments determine properties of components and determine which other constructs are provided by the semantic structure and which relations exist between the constructs. This chapter discusses possibilities for such additional commitments, and fixes a specific set of commitments to establish an informal description of the semantic structure developed in this thesis.

The semantic structure established in this chapter provides constructs specifically designed for modelling multi-agent systems as compositional systems. Consequentially, an application of the semantic structure is itself a compositional system and the discussion of possible additional commitments in this chapter is a discussion of different views of what a compositional system is (hence the title of this chapter). In Section 2.1, compositional systems are introduced. An important aspect of compositional systems, information exchange, is discussed in Section 2.2.

2.1 Compositional Systems

As stated in the previous chapter, the basic assumption adopted in this thesis is that multi-agent systems are modelled as compositional systems. Therefore, the basic construct provided by the semantic structure is the *component*, the building block of a compositional system. In Section 2.1.1, properties of the component construct provided by the semantic structure are discussed. Section 2.1.2 discusses components from the point of view of related areas of research.

Components in a compositional system are related with one another, for instance by information exchange between components. In other words, information exchange is the glue that holds together the building blocks of a compositional system. This section assumes that the semantic structure provides a construct for information exchange. The next section, Section 2.2, presents an extensive discussion of different properties of such a construct.

2.1.1 Components and Compositional Systems

A component is a locus of information and computation. A component is related to other components in two ways: by information exchange and by composition, which together determine the structure of a compositional system. Three aspects of a component are distinguished: its *state*, its *interfaces* and its *composition structure*.

In general, a component stores information, which can change over time because of new information becoming available due to computation or information exchange. The information contents of a component determines its *information state*, or *state* for short. All computation performed by a component is based on its information contents. Information exchange in a compositional system is aimed at changing the state of the components that exchange information. Components in a compositional system can be active concurrently or sequentially, and thus state changes in a compositional system can occur concurrently or sequentially.

The information state of a component is only accessible via the *interface* of the component. As a consequence, a component can (only) exchange information with another component via the interfaces of the two components. A property of the component structure provided by the semantic structure developed in this thesis is that the interface of each component consists of two parts: the *input* and *output interface*. The input interface is that part of a component's state that is used as input for the computations of the component. Thus, the input interface can be used to provide information to a component. The output interface is the part of a component's state that contains the output of the component. This interface can be used to make results of computations performed by the component accessible to other components. Changes in the output part of a component's state are visible to other components.

The third aspect of a component is its *composition structure*: a component may itself be composed of other components. Thus, the composition structure is recursive: a component in a compositional system may consist of other

components, so the component is itself a compositional system. A component that is composed of other components is called a *composed component*, and the components of which it consists are called the component's *subcomponents*. The component of which a specific other component is a subcomponent is called the subcomponent's *parent component*. (In this thesis, the term 'subcomponent' is exclusively used for *direct* subcomponents, or children of a parent component, and not transitively. Thus, the 'grandchildren' of a specific parent component are not called subcomponents of the parent component.)

2.1.2 Some Perspectives on Components

The previous subsection characterised a component as a locus of information and computation. Different perspectives on the concept of a component can be found in related areas.

- In Software Engineering, there is currently much interest in *component based software development*. The central idea in component based software development is that software is constructed by connecting ready-made software components. D'Souza and Wills (1998, p. xvii) call such components 'plugable software' and compare component based development with earlier approaches to modularization of software: "People have always divided programs into modules, but the original reasons were meant to divide work across a team and to reduce recompilation. With plugable software, the idea is that you can combine components in different ways to make different software products." As D'Souza and Wills indicate, component based development in Software Engineering is focused on providing technology that supports the division of programs in program modules. Earlier technologies are, for instance, the module construct provided by CLU (Liskov, Moss, Schaffert, Scheifler & Snyder, 1981) and Modula-2 (Wirth, 1982), and object orientation. The main difference between component based software development on the one hand and more traditional technology on the other hand is that component based development not only provides design-time technology, but also runtime support for 'plugable software'. With the traditional technologies, after compilation of all modules, at run time only a large, monolithic composed system remains. Component based software is also at runtime distinguishable as a collection of components that can only exchange information via well-defined interfaces, and new components can be added dynamically. So, also at run time a component is distinguishable as a concrete locus of information and computation.
- A component is a locus of information and computation. A compositional system thus consists of a number of loci of information where processes of computation take place. As stated in the previous subsection, these processes may take place simultaneously. Therefore, a compositional system

can be studied from the perspective of the *Distributed Systems* area. However, this area usually abstracts from the concept of a component with its properties such as e.g. interfaces and focuses on the processes that take place simultaneously and exchange information. Concepts used in the theoretical analysis of Distributed Systems are employed in Chapter 7.

- The concept of a component is also of importance in a number of *coordination languages* The coordination paradigm views programming distributed systems as "the combination of two distinct activities: the actual computing part comprising a number of processes involved in manipulating data and a coordination part responsible for the communication and cooperation between the processes." (Papadopoulos & Arbab, 1998). Most coordination languages completely abstract from the computing part by only distinguishing a number of *components* that together contain all computational processes. The coordination part abstracts from the computations that take place within components and focuses on coordination between components.
- A compositional system is a system in which the dynamics of a system composed of a number of components is defined by a composition relation. The composition relation itself defines the dynamics of a system composed of a number of components in terms of the dynamics of those components. This compositionality principle gives rise to the study of *compositional reasoning* about compositional systems: assertions about a compositional system are proved solely on the basis of assertions about the components of the compositional system and a composition relation, without additional knowledge about the internal structure of those components (Roever, 1998). Thus, the concept of a component is central in the study of compositional reasoning. However, the precise properties of a component are usually left abstract.

2.2 Constructs for Information Exchange

The previous section identified information exchange as the glue that holds together the building blocks of a compositional system: components. This section discusses constructs provided by the semantic structure that support information exchange, properties of these constructs and relations with other constructs and properties, such as components, interfaces and information states.

Both the terms communication and information exchange (which may be considered to be synonyms[3]) have a connotation of mutual influence: if a component A exchanges information with a component B, (different) information is transferred from A to B and from B to A. However, a number of forms of

[3] Merriam-Webster WWWebster Dictionary.

communication, such as e.g. television broadcasts, are better characterised as unilateral: information is not exchanged between two agents, but is transmitted from one agent to another. Information transmission seems to be a more basic concept than communication or information exchange (which can be seen as bi-directional information transmission). Therefore, in the rest of this chapter the term information transmission is used. (If, however, in a certain context focus is on mutual influence, both of the terms communication or information exchange may be used.)

As information transmission is assumed to be directed, it is important to distinguish between the component from which an information transmission departs and the component to which the transmission is directed. The component that initiates a transmission is called the *source component*, while the component to which the transmission is directed is called the *destination component*.

There is a multitude of choices with respect to which constructs, properties of constructs and relations between constructs a semantic structure may provide to facilitate information exchange. Consequentially, this section presents an extensive discussion of the concept of information transmission. As a starting point, the following are the eight most important commitments made for the semantic structure, each of which is discussed in a separate subsection, together with alternative commitments:

- Information transmission is restricted to components that are subcomponents of the same parent component or between a parent component and its subcomponents (Section 2.2.1);
- The semantic structure only provides constructs and relations between constructs for point-to-point information transmission (Section 2.2.2);
- The semantic structure provides a construct that represents a communication channel between components (Section 2.2.3);
- To transmit information from one component to another, the source component composes a message that consists of the information to be transmitted. This message is volatile: it cannot exist before or after the transmission. (Section 2.2.4);
- Information transmission may have an effect on the component that initiated the transmission (Section 2.2.5);
- A component has the ability to dynamically enable or disable receipt of information (Section 2.2.6);
- Information transmission is asynchronous (Section 2.2.7);
- State changes are logically instantaneous (they do not seem to have any duration at all) and exclusive with respect to transmissions (a number of consecutive transmissions results in the same number of state changes per agent) (Section 2.2.8).

Some commitments give rise to additional choices with respect to properties of new constructs introduced or the details of relations with other constructs. All commitments are summarised in Section 2.2.9.

2.2.1 Restriction on Direct Information Transmission

A property of the component construct provided by the semantic structure developed in this thesis is that in a compositional system, (only) six kinds of information transmission are allowed. Direct information transmission (without relay by intermediate components) between arbitrary components is not allowed, except for the following six cases:

- Information transmission between components that are subcomponents of the same component (called private information transmission),
- Transmission from the input interface of a component to the input interface of one of its subcomponents (called import mediating information transmission),
- Transmission from the output interface of a subcomponent to the output interface of its parent component (called export mediating information transmission),
- Transmission from the input interface of a component to the output interface of the same component (called cross-mediating information transmission),
- Transmission from a component to an information link between two other components that are all subcomponents of the same component (called link modifier information transmission), and
- Transmission from an information link between two other components that are all subcomponents of the same component to a third component (called link monitoring information transmission).

The role of link modifier and monitoring information transmissions is discussed in Section 2.2.3.2 and in Chapter 8.

Results of computations carried out by a component can only influence other components via information transmission, which is restricted as explained in this section. In other words, side effects of computations can only be used outside the component where they originate if the information produced as a side effects is transmitted via information links. The semantic structure does not support other side effects.

The pictorial representation of a component in this thesis is provided in Figure 2.1 below. This figure shows three components, one of which is a composed component that consists of two subcomponents. All six allowed forms of information transmission are shown, represented by arrows. The information

transmission from one of the primitive components at the left-hand side to the composed component is called private because there is an implicit component of which all depicted components are subcomponents.

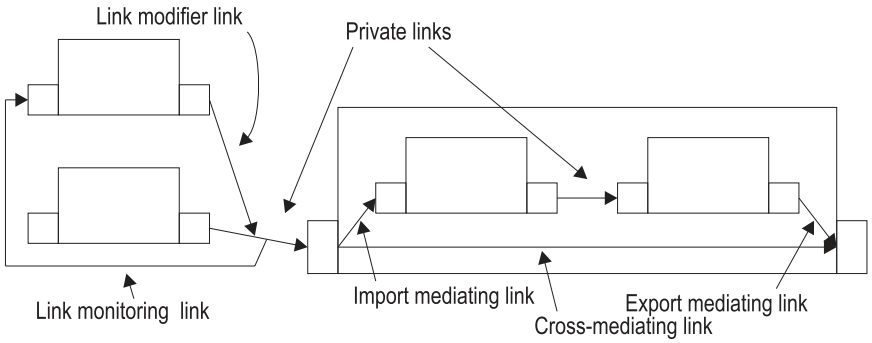

Figure 2.1: Compositional system.

2.2.2 *Point-to-point Transmissions, Multicast or Broadcast*

The semantic structure (only) provides a construct for information transmission from one component to a single other component (or possibly the same component). This is called point-to-point transmission or sometimes unicast transmission. The semantic structure does not provide constructs for multicast or broadcast transmissions. In a multicast transmission, information is transmitted to a subset of all components in a compositional system. Several forms of multicast are possible, differing in the flexibility provided with respect to how the subset of agents involved is determined. In a broadcast transmission, information is transmitted to all components in a compositional system.

Whether or not constructs for multicast and broadcast transmission are provided is primarily a matter of demarcation. Multicast and broadcast transmissions can also be realised as a set of point-to-point transmissions. An application of the semantic structure can thus introduce multicast and broadcast transmissions and define these forms of transmission in terms of point-to-point transmissions.

2.2.3 *The Information Transmission Channel*

The semantic structure provides a construct that represents communication channels between components. This construct is called an *information link*, or *link* for short. The commitment to a construct that represents communication channels enables more details of information transmission to be defined in applications of the semantic structure. (It is not strictly necessary to provide a construct for representing communications channels. If it is not provided, information transmission is defined directly in terms of the components involved.)

A number of additional commitments can be distinguished with respect to how a communications channel, and information transmission, are represented in the semantic structure. Each additional commitment is discussed in a separate subsection below. The additional commitments distinguished are:

- An information link is explicitly coupled to two named components. (Section 2.2.3.1);
- The state of a communications channel is represented by the information link construct (Section 2.2.3.2);
- An information transmission process results in *two* changes of the state of an information link (Section 2.2.3.3).

2.2.3.1 Anonymous or Named Transmission

In the semantic structure presented in this thesis, for each link two fixed components are named as its source and destination, respectively. As a consequence, for each information transmission, the destination of the transmission (the component that receives the information transmitted) is determined by the link and is fixed as soon as the source component initiates the transmission.

An alternative possibility enabled by the commitment to provide a construct that represents communication channels (the information link), is to determine the destination at a later time. This possibility amounts to a form of anonymous information transmission, in which the source component for the transmission does not have to know the identity of the receiver. Instead, the source component just places the information to be transmitted on the channel. Once information is placed on the channel, the (copy of the) information on the channel is independent from the component that placed it on the channel. It is assumed that all components have access to the channel and each component may, independently of the source, try to take information from the channel. This form of information transmission is similar to the blackboard architecture used in distributed knowledge based systems (Nii, 1986a, 1986b). As all components in principle have access to the channel, this form of transmission can be seen as a form between multicast and broadcast. Both anonymous transmission and independence of the source component are key principles of the *generative communications* paradigm originally proposed by Carriero & Gelernter (1992). Generative communication is the defining feature of one class of coordination languages, called data-driven coordination languages (Papadopoulos & Arbab, 1998). Gaspari (1998, p. 7) lists anonymous communication as a possible way to abstract from symbol level addressing issues in knowledge-level models of multi-agent systems. A necessary (but not sufficient) condition for generative communications is that the semantic structure represents the state of an information link.

2.2.3.2 *The State of an Information Link*

The semantic structure not only distinguishes the state of components, but also of information links. In the semantic structure, the state of an information link may depend on both (or either of) the *contents* of the channel it represents at a particular point in time and on the *activity* of the channel at a particular point in time. As an example, consider a TCP/IP network connection. The contents of the connection at a particular point in time is, for instance, a set of TCP messages. A TCP/IP connection may be open, closed, busy or idle, which describes the state of the transmission as an activity.

Like the state of a component, the state of a link changes over time. For example, suppose the state of a link is determined by the contents of the link represented as a set of messages. At the beginning of a transmission, the information to be transmitted is, in the form of a message, added to the set of messages in transit. Thus, the state of the link changes from a state for which the set did not contain the new message to a state for which it does. After some time, the message is delivered, ending the transmission. The state of the link changes again, from a state for which the set of messages in transit contains the message that is delivered, to a state for which the set of messages in transit no longer contains this message. The following commitment further supports such detailed representation of information transmission, enabled by the decision to represent the state of an information link. The state of an information link is accessible by other components for control purposes via link control information transmission (see Section 2.2.1).

2.2.3.3 *Information Transmission and Link State Changes*

The commitment to distinguish the state of information links opens up possibilities for further commitments with respect to how the state aspect of an information link can be used. In fact, the semantic structure developed in this thesis supports detailed representation of information transmission. This support is enabled by the commitment to distinguish link state changes for each information transmission. More specific, for each information transmission, *two* state changes are distinguished. With this commitment, it is possible to model information transmission, for example, in the following way. Suppose that the state of an information link reflects both whether the link is enabled (ready to transmit information) and the contents of a queue of messages in transit. Information transmission can, in this example, only take place if, according to the current state of the link, the link is enabled. If this is the case, the state of the link changes to a new state in which the message to be transmitted is added to the queue of messages in transit. After some time, the message is delivered and the state of the link changes again to a state in which the message just delivered is no longer on

the queue of messages in transit. Additional state changes may occur in between the two state changes associated with the transmission in this example.

Several relations between decisions presented in Section 2.2.3 are indicated in the text above. An overview of these relations is depicted in Figure 2.2 below. Similar figures are provided for Sections 2.2.4 to 2.2.6, and in Section 2.2.9 (summary), the figures from Section 2.2.3 to 2.2.6 are collated to provide an overview of all relations between the commitments. In these figures, the commitments set for the semantic structure are listed in italics. Alternative possibilities for these commitments are grouped with the commitments in boxes, one for each issue. Solid arrows indicate that a specific possibility gives rise to an additional issue. These solid arrows always start from a specific possibility (upper or lower part of a box) and end at an issue (middle of a box).

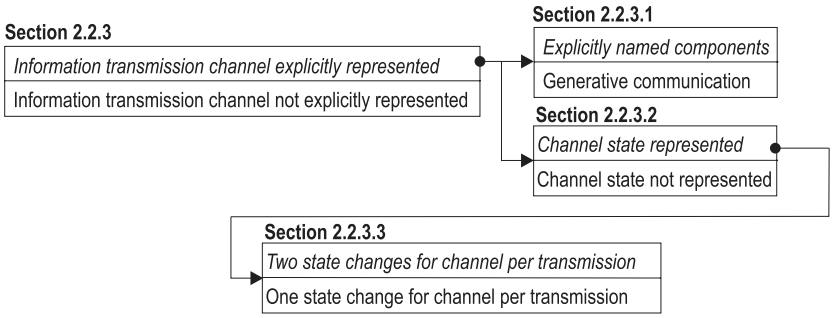

Figure 2.2: Relations between commitments in Section 2.2.3.

2.2.4 Relationship between Transmission and Transmitted Information

The primary goal of information transmission is to change the state of another component (and not merely to send a message). However, in a number of examples, the information transmitted (the message) is mentioned explicitly, while the different states of the components are not mentioned at all. This suggests that messages themselves can be important as well.

The semantic structure is committed to *volatile messages*: messages cannot exist before or after transmission. This commitment is based on a distinction between a message and its content information. As stated above, a component transmits information to change the state of another component. To this end, the component prepares some information that is to be transmitted to the other component. This information, the content information, is then encapsulated in a unit that can be transmitted. The encapsulated content information is called a message. A specific piece of content information can be encapsulated in different kinds of units, depending on the kind of transmission unit used. As an example, consider the following situation. An agent, called *A*, wants to change the state of another agent, *B*, such that *B* comes to believe that a resource is available. Agent *A* knows that

agent *B* only believes that information is true if the source is explicitly mentioned. Therefore, agent *A* prepares specific information for *B*. After this preparation phase, agent *A* reaches a state in which it has available content information saying: "According to our local newspaper, the resource is available." This information is only available inside agent *A*. If agent *A* wants to transmit this information by postal mail, he or she puts it in an envelope. In other words, the content information is encapsulated in a specific unit (the envelope) to form a mail message. If agent *A* wants to transmit this information by telephone, he or she calls *B*. The same content information is encapsulated in a completely different (and more volatile) unit (the phone conversation) to form a different message. However, the effect on the state of *B* is the same: in both cases, *B* reaches a state in which he or she knows that *A* transmitted content information stating that "According to our local newspaper, the resource is available."

In the example given above, the two messages differ in their volatility. On the one hand, a telephone conversation is not normally received, and it only exists as long as the conversation lasts. On the other hand, a postal message stays in existence after the conversation until it is deliberately deleted. In the semantic structure, all messages are volatile. This means that after a transmission ends, only the content information is available, in one or both of the agents involved in the transmission. The content information is available as part of the information present in an agent, and thus partially determines the state of the agent. Thus, there is a tight coupling between information transmission and the states of components. During the transmission, the message can exist in the source component, the link and/or the destination component. As a consequence of the commitment to volatile messages, there is no need for the semantic structure to provide an independent message construct.

The primary advantage of this decision is as follows. The commitment to a tight coupling between messages and states emphasises the goal of an information transmission: to change the state of the destination component. To achieve this goal, the independent existence of a message is not important or possibly incorrect. For example, consider a human agent *A* who wants to inform another agent *B* of the availability of a certain resource. The goal of agent *A* is to change the state of *B* such that *B* knows that the resource is available. To achieve this goal, *A* can make a telephone call or send an email message. In the case of an email message, it is relatively simple to distinguish a message that exists independent of the states of both *A* and *B*, but in many applications of the semantic structure, it is not important to distinguish this message as such. However, in the case of a telephone call, the telephone is used to deliver a message on the availability of the resource. This message only exists in the minds of *A* and *B*, and not independently in the telephone system.

Applications of the semantic structure in which the independent existence of a message cannot be avoided are still supported. Such applications are, for instance, applications in which generative communications are analysed (see Section 2.2.3.1),

models of multi-agent systems based on speech act theory (Chaib-draa & Vanderveken, to appear)[4] and protocol specifications, such as, for instance, the HTTP protocol. Such applications are supported because, as explained in Section 2.2.3.2, the semantic structure not only distinguishes the state of components, but also the state of information links. In the applications mentioned, messages that exist independent of the state of a component appear in the state of an information link.

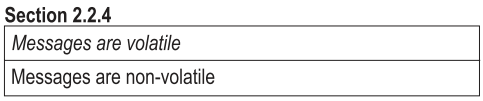

Figure 2.3: Relations between commitments in Section 2.2.4.

2.2.5 State Changes in Information Transmission

Information transmission is performed by one component. This component initiates the transmission to attempt to change the state of another component. Thus, if the transmission succeeds, the state of the other component changes. As stated in the previous subsection, the semantic structure is committed to defining information transmission as a relation between the states of the components involved, without assuming independent existence of messages. This relation is an aspect of the link construct provided by the semantic structure. This subsection and the next subsection further discuss this relation.

The semantic structure defines the transmission relation in terms of *two* states of the source component for each transmission. The first state is the state in which the transmission is initiated. The second state is a new state that is determined by the *result* of the transmission. This state reflects that the transmission has been initiated, but for specific applications of the semantic structure, it may also reflect the result of the transmission.

In the semantic structure, the second state need *not* be the immediate successor state of the first state in the dynamics of the source component. In other words, initiating a transmission is similar to executing a *non-blocking send operation* provided by operating systems or (agent) programming languages. This commitment is made to support applications in which the second state is used to reflect the result of a transmission. Consider the point in time at which a component initiates a transmission. From that moment on, each state of the

[4] Speech act theory forms the foundation of a number of agent communication languages such as KQML (Finin, Labrou & Mayfield, 1997) and the FIPA Agent Communication Language (ACL), ([FIPA]). The agent programming languages Agent0 (Shoham, 1993) and PLACA (Rebecca-Thomas, 1995) are based on Speech Act Theory. Applications of these languages are examples of multi-agent systems in which information transmission is defined in terms of messages.

component should reflect the fact that a transmission has been initiated. However, if these new states also reflect success of the transmission, between the moment the transmission was initiated and the moment at which information on success of the transmission is available, additional state changes may take place. If the second state needs to be the immediate successor of the first state, such additional changes would not be allowed and the source component would be inactive until information on the success of the transmission is available (*blocking send operation*).

As an aside, the relation that defines information transmission only defines how information transmission affects the component (or, more precisely, the states of the components involved). The relation does not specify *that* the state changes defined by the relation should occur, nor does it specify anything about at which time the state changes should occur. Extending the example from Section 2.2.4, consider the relation that defines the information transmission from *A* to *B*. Such a relation would, in the easiest case, only express that *if*, at a certain point in time, agent *A* is in a state in which he or she knows that a certain resource is available, *then* agent *A* should arrange that agent *B* reaches a state in which agent *B* also knows that this resource is available.

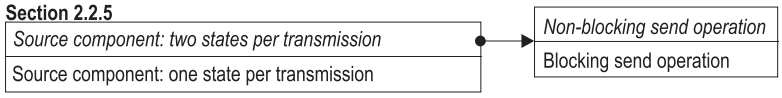

Figure 2.4: Relations between commitments in Section 2.2.5.

2.2.6 *Conditions for Enabling Information Transmission*

As explained at the beginning of Section 2.2, a component transmits information with the aim of changing the state of another agent. This implies that because of information transmission directed at a specific component, the state of this specific component changes. Such a state change differs from state changes caused by the activity of the component itself. State changes caused by a component's activity originate within that component, and are thus fully controlled by the component. However, a state change resulting from information transmission directed at a component originates from another component and is not controlled by the component to which it is directed. However, the semantic structure supports defining information transmission in such a way that a destination component dynamically decides whether it allows state changes originating outside it. In other words, at some moments in time, the component is not ready to receive transmissions, while at other moments it is.

This commitment has both global consequences and local consequences. The global consequences are consequences for the component that initiated the transmission, and are discussed in Section 2.2.7. The local consequences are consequences for the component to which the transmission is directed. An obvious

consequence is that the relation that defines information transmission in terms of the states of the components involved, needs to define the transmission using *two* states for the destination component. First, there is the state that results from the transmission and is its aim. Second, there is the state that determines whether a state change leading to the first state is allowed.

In the context of software components, the issue of how state changes originating outside the component are interleaved with state changes inside the component is related to the availability and semantics of a 'receive' operation. Such an operation may be provided by the operating system, a programming language, or a model specification language. In some frameworks, these operations are implicit: the semantics of the framework ensures that at certain moments in time, the component allows state changes originating outside it, without explicit reference in the specification of the component.

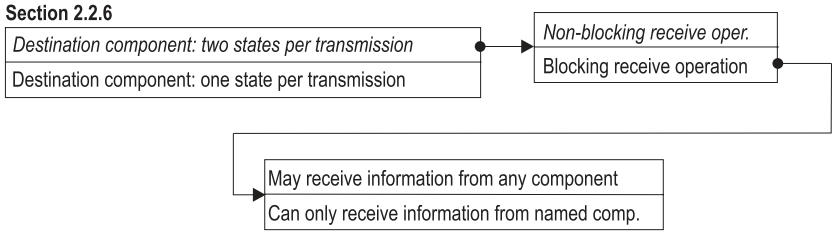

Figure 2.5: Relations between commitments in Section 2.2.6.

As the relation with read operations in a software component context suggests, there are additional choices with respect to how a component deals with state changes beyond its control. Consider the point in time at which a component, possibly as the result of some deliberation, decides to allow state changes to occur that originate outside the agent. There are two possibilities with respect to which state changes can occur after this point. The first possibility is to allow any state change, both state changes that originate outside the component and state changes that results from the component's own activity. In other words, after this point in time, the agent is able to continue with its own activities. This possibility is similar to the semantics of so-called *non-blocking receive operations* provided by operating systems or programming languages. The second possibility is to *only* allow state changes originating outside the component. In this case, after allowing state changes that originate outside the component, the component cannot continue with its own activities and has to wait for a state change originating outside it. (It is assumed that after such a change has occurred, the component regains its ability to carry out its own activities.) This possibility is similar to the semantics of *blocking receive operations*. With respect to the second possibility, an additional division is possible: either state changes originating at a specific component are allowed, or state changes originating from any other component, or it is possible to dynamically choose between these two possibilities. The constructs provided by

the semantic structure allow maximum flexibility and therefore allow any state change, both state changes originating outside the component and state changes that results from the component's own activity.

2.2.7 *Synchronous or Asynchronous Transmission*

To illustrate the difference between synchronous and asynchronous information transmission, consider the following examples from everyday life. Suppose a human agent *A* needs to transmit some information to another human agent *B*. Suppose *A* considers a telephone call to be the appropriate means of transmission. If *B* is available at the time of the call, the information can be transmitted and after transmission, *A* considers the transmission to have been successful. This is an example of synchronous transmission. A characterising property for synchronous transmission is that an arbitrary event in the life of *A* or *B* happened before *A* started the transmission if and only if it happened before *B* answered the phone. Similarly, an arbitrary event in the life of *A* or *B* happened after *A* ended the transmission if and only if it happened after *B* ended the conversation. (It is not necessary to assume that there is a global clock that determines whether an event happened before another event. This is further explained in Chapter 7.) If agent *A*, however, considered sending a (postal or electronic) mail message as the appropriate means of transmission, the situation differs. After *A* composed the message and either dropped it in a mail box or hit the 'send' button in his or her email program, agent *A* can continue with other activities and probably considers the transmission to have been successful. As a consequence, while the message is in transit, events may happen. Events that happen while the message is in transit necessarily happen after *A* sent the message, but before *B* receives the message. Thus, the characterisation of synchronous transmission does not apply in the case of sending mail. Sending mail is an example of asynchronous transmission. (As an aside, also if agent *A* only considers the transmission to have been successful after receiving acknowledgement of receipt from *B*, the transmission is still asynchronous, because it is assumed that other events may happen while either the message from *A* to *B* or the acknowledgement from *B* to *A* is in transit. If, somehow, *A* were to be completely inactive until acknowledgement was received, then the transmission would be synchronous. Moreover, the example would then show how a synchronous transmission can be the result of two asynchronous transmissions.)

The reason for committing to asynchronous transmission in the semantic structure is to create independence between the source and destination components of a transmission. In the previous subsection, the commitment to dynamically allow or disallow state changes that originate outside a component was discussed. If such changes are disallowed for some period of time, and during this period another component initiates a transmission, with synchronous transmission the initiating component has to remain inactive for some time, which is not the case if asynchronous transmission is assumed. (A third possibility would

be to assume that under these circumstances, transmission fails.) This interference constitutes the global consequences mentioned in the previous subsection.

Committing to asynchronous transmission still supports applications of the semantic structure based on synchronous transmission, as synchronous and asynchronous transmission can each be defined in terms of one another. On the one hand, synchronous transmission can be achieved using asynchronous transmission by first asynchronously transmitting information and then remain passive until acknowledgement of receipt is sent back asynchronously. (As an aside, there are more sophisticated schemes for achieving synchronous transmission, see for instance (Soneoka & Ibaraki, 1994)). On the other hand, asynchronous transmission can be achieved using synchronous transmission by introducing a separate entity, possibly the communications channel itself, that is connected to both components involved in the transmission. One of the components synchronously transmits information to this separate entity, which in turn synchronously transmits it to the other component. The combined effect is an asynchronous transmission from one component to the other.

2.2.8 Exclusive, Logically Instantaneous State Changes

The final commitment listed at the beginning of Section 2.2 is the commitment to provide state changes that are logically instantaneous (they do not seem to have any duration at all) and exclusive with respect to transmissions (a number of consecutive transmissions results in the same number of state changes per agent). This commitment is related to the issue of state changes in information transmission (Section 2.2.5) and the issue of enabling conditions for information transmission (Section 2.2.6). As explained in those sections, information transmission results in state changes in both the source and destination components. These state changes reflect the arrival of new information in the input interface of the destination component and, possibly, information on the result of the transmission becoming available in the source component.

Consider a compositional system in which two components, C_1 and C_2, both transmit information to a third component, D, at the same time. According to Section 2.2.5, each transmission results in two state changes for D. Each of these four state changes is exclusive, i.e. each state change only reflects the effect of one transmission, and is logically instantaneous, i.e. the state change does not seem to have a duration. In other words, state changes are not combined, and they cannot be interrupted. As state changes cannot be combined, even though two transmissions may happen at the same time, four states in the behaviour of D can be distinguished that are related to the two transmissions. However, the order in which these states occur is not fixed.

If state changes were not exclusive with respect to transmissions and logically instantaneous, then the following situation could possibly arise. Assume that C_1 is the first to start transmitting information to D. During C_1's transmission, C_2 also

starts a transmission, and assume that C_2's transmission finishes at the same time as C_1's transmission. In this case, the state of D that results from information transmission has to reflect the combined effect of both transmissions. Two situations can arise. The combined effect of both transmissions can be inconsistent. In this situation, the new state of D has to represent inconsistent information. Alternatively, the effect of one transmission may cancel the effect of the other, in which case, the state determined by the combined effect equals the state just prior to the two transmission. From the point of view of D, nothing seems to have happened. These situations can be avoided by the commitment presented in this section.

2.2.9 Summary

The previous sections presented a number of commitments with respect to information transmission made in the development of the semantic structure. In this section, all commitments are summarised in two ways. First, the figures presented in Sections 2.2.3 to 2.2.6 are collated in Figure 2.6. In this figure, the dashed arrow indicates that the commitment from which it departs is required by the commitment to which it is directed.

Second, the commitments are summarised by depicting an instance of information transmission (see Figure 2.7). This figure depicts the behaviour of two components, A and B, and of an information link L from A to B. The behaviour is depicted in the form of *time lines.* Each time line depicts a sequence of states (represented by rectangles) and state transitions (represented by straight arrows). The instance of information transmission depicted consists of the eight states $\nu_{A,i}$, $\nu_{A,j}$, $\nu_{L,i''}$, $\nu_{L,j''}$, $\nu_{L,k}$, $\nu_{L,l}$, $\nu_{B,i'}$ and $\nu_{B,j'}$. State $\nu_{A,i}$ is the state of component A in which the transmission starts. State $\nu_{B,j'}$ is the state of component B that is the result of the transmission, or, in other words, the state that reflects that information present in A in state $\nu_{A,i}$ is also present in B. As explained in Section 2.2.5, the semantic structure distinguishes a second state for each information transmission in the source component. This is state $\nu_{A,j}$. This state can be used by applications of the semantic structure to mark the end of the transmission. Furthermore, it can be used to represent the receipt of an acknowledgement sent by B. Independent of the usage of state $\nu_{A,j}$, according to Section 2.2.5, state $\nu_{A,j}$ may be the immediate successor of state $\nu_{A,i}$. (indicated by the curved arrows labelled '1' that show how the figure would be if this were the case). However, as stated in Section 2.2.5, state $\nu_{A,j}$ need not be the immediate successor of state $\nu_{A,i}$ (commitment to non-blocking send). Consequently, there may be other states between $\nu_{A,j}$ and $\nu_{A,i}$ (indicated by the arrows labelled '2').

Likewise, as explained in Section 2.2.6, the semantic structure distinguishes a second state for each information transmission in the destination component. This is state $\nu_{B,i'}$. This state can be used by applications of the semantic structure to reflect that component B is ready for state changes that originate outside the

component (in other words, B is ready to receive information). Independent of the usage of state $v_{B,i'}$, according to Section 2.2.6, $v_{B,j'}$ may be the immediate successor of state $v_{B,i'}$. (indicated by the arrows labelled '3'). However, as stated in Section 2.2.6, state $v_{B,j'}$ need not be the immediate successor of state $v_{B,i'}$ (commitment to non-blocking receive). Consequently, there may be other states between $v_{B,j'}$ and $v_{B,i'}$ (indicated by the arrows labelled '4').

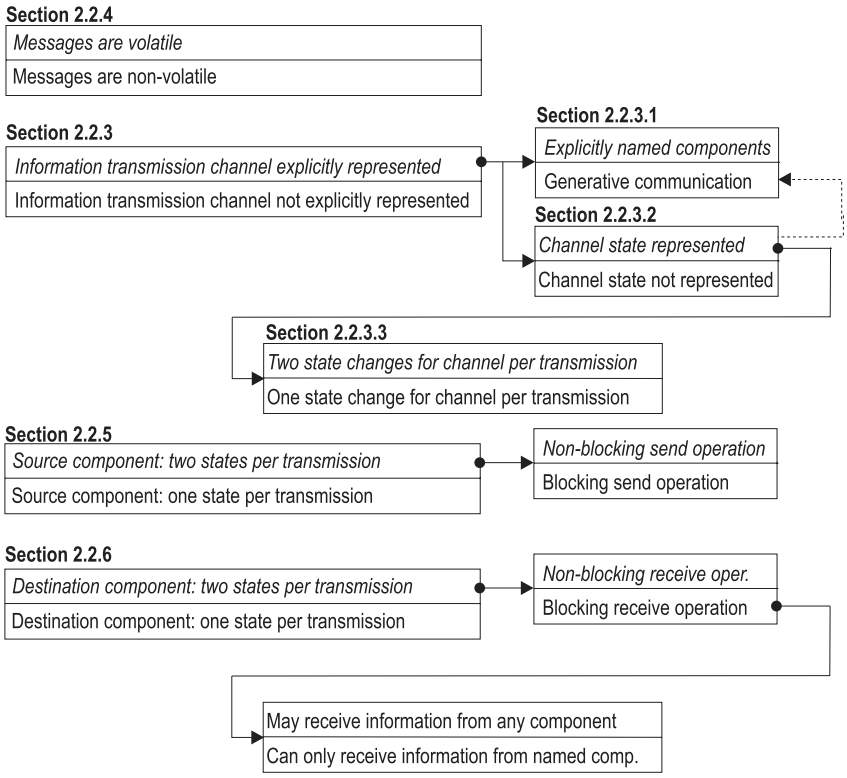

Figure 2.6: Overview of commitments.

Finally, Figure 2.7 also shows that the behaviour of information links is explicitly represented in the semantic structure (Section 2.2.3.2). State $v_{L,i''}$ is the first state in the behaviour of L associated with the transmission. Applications of the semantic structure can choose to require the link to be enabled before transmission can take place. In this case, state $v_{L,i''}$ must be an enabling state. Furthermore, applications can choose to represent messages in transit in the state of L. In this case, the transition from $v_{L.i''}$ to $v_{L,j''}$ can be used to represent addition of a message to the queue of messages in transit. Later on in the behaviour of L, the transition from $v_{L.k}$ to $v_{L,l}$ can be used to represent removal of the message from the queue of messages in transit.

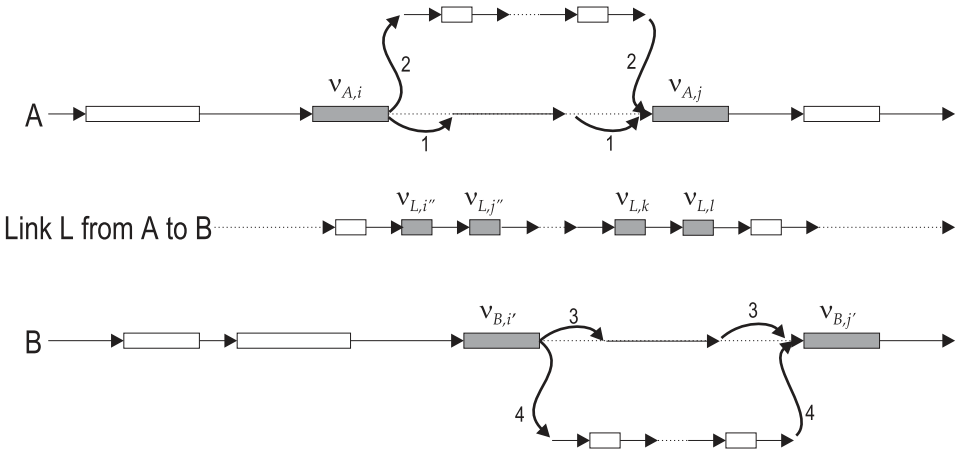

Figure 2.7: Information transmission.

Chapter 3

Multi-Agent Systems as Compositional Systems

The previous chapter presented a number of commitments with respect to the constructs and relations that together constitute the semantic structure for compositional systems developed in this thesis. The commitments, which are not specific to multi-agent systems, are fixed for the semantic structure, and consequently, all applications of the semantic structure necessarily use the constructs committed to in the previous chapter.

As stated in Chapter 1, the aim of this thesis is to develop a compositional, formal semantic structure that can be used for multi-agent systems dynamics. Thus, the semantic structure is intended to be applied in the domain of *multi-agent systems*. More specifically, the intended use of the semantics structure is to provide semantics for models of multi-agent systems, (in particular their dynamics), or for specification languages. As the basic assumption, stated in Chapter 1, is that multi-agent systems are modelled as *compositional systems*, the semantic structure presented in the previous chapter provides constructs for building compositional systems. The most important issue for applications of the semantic structure is thus: *how can multi-agent systems be modelled as compositional systems*. This is the topic of this chapter.

The previous chapter discussed different possibilities for commitments to specific constructs and relations between constructs, and explicitly fixed a set of commitments. As the commitments discussed in the previous chapter determine the contents of the semantic structure, it was necessary to commit to a specific set of constructs. However, in this chapter, different possibilities for modelling multi-agent systems are discussed, without committing to specific choices. There is no need to commit to a specific way of modelling multi-agent systems, because for each application, different choices can be made.

The discussion of modelling multi-agent systems is organised as follows. First, the basic principle of modelling multi-agent systems as compositional systems is explained in Section 3.1. Section 3.2 provides further guidelines for modelling multi-agent systems as compositional systems by presenting a generic

compositional agent model. Section 3.3 discusses possible ways to model various phenomena of multi-agent systems, which should be regarded an illustration of the application of compositionality in the area of multi-agent systems. Moreover, some issues that are treated in more detail in later chapters, are also raised in this section.

3.1 Multi-Agent Systems as Compositional Systems

In Chapter 1, a multi-agent system is defined as a group of agents together with their common, shared environment. The basic assumption adopted in this thesis is that multi-agent systems are modelled as compositional systems. Thus, an application of the semantic structure has to identify entities within a multi-agent system that are represented by components in the application of the semantic structure. This section discusses a guideline that can be followed in constructing a model of a multi-agent system as a compositional system.

The starting point is to focus on a multi-agent system as a *system*. A system is often defined as a connected collection of parts (see e.g., Oxford Concise Dictionary, and Wieringa, 1995). As the semantic structure focuses on multi-agent systems *dynamics*, in this thesis, a system is seen as a coherent collection of *processes and entities that execute processes (agents and the environment)*. (This perspective, which includes explicit focus on processes, may be considered to be a dynamic variation of the definition cited above.) It is understood that any pattern of change within a system constitutes a process. In addition, the interpretation of *coherence* is deliberately left vague: it merely indicates that some relationship between the processes in a system must exist. The guideline consists of the following four steps:

- First, in a multi-agent system that has to be modelled, processes are identified. Moreover, these processes can either be classified as deliberation processes or environmental processes (Section 3.1.1);
- Then, (active) entities are identified (the agents and the environment) that execute the processes. An agent or the environment can execute more than one process simultaneously (Section 3.1.2);
- After that, a multi-agent system is modelled as a compositional system (see Chapter 1), which is achieved by representing each process as a component. This component encapsulates both the information used by the process and the process (computation) itself (Section 3.1.3);
- Finally, relations between processes and between processes and the agents or the environment have been identified (Section 3.1.4).

3.1.1 Deliberation Processes and Environmental Processes

It is assumed that processes can be classified as either deliberation processes or environmental processes. Deliberation processes, which are always internal to an

agent, comprise the main activity of an agent. Deliberation is needed (1) to acquire information by communication with other agents or by observing the environment, (2) to process information, and (3) to initiate actions or transmit information to other agents. Deliberation processes identified in the multi-agent system are represented by components in a compositional system that models the multi-agent system. As stated in Chapter 2, a component is a locus of information and computation. Deliberation processes are thus modelled as computational processes.

Environmental processes are processes that are executed by the environment. In other words, environmental processes are the processes identified in a multi-agent system that are not internal to an agent. Similar to deliberation processes, environmental processes are represented by computations in a compositional system that models a multi-agent system.

It is assumed that interaction is *not* identified as separate processes, but as "mutual or reciprocal action or influence[5]", thus as a relation between processes (relations between processes are discussed in Section 3.1.4). Moreover, interaction is not a basic concept. Three forms of interaction are distinguished: communication, action execution and observation. (Which instances of interaction are viewed as communication, action execution or observation differs from application to application and is discussed as a modelling choice in Section 3.3.3.) In addition, interaction is viewed as consisting of two (or more) instances of *unilateral* influence from which "mutual or reciprocal influence" emerges. Thus, like information exchange (see beginning of Section 2.2), interaction is viewed as inherently composed.

Exactly which processes are distinguished as deliberation processes and which as environmental processes is subject to modelling choices discussed in Section 3.3 below.

3.1.2 Agents and the Environment

It is assumed that specific entities in a multi-agent system can be identified as agents and others as the environment. How this identification is to be performed is viewed as a process akin to traditional knowledge acquisition or requirements engineering and is outside the scope of this thesis. (Compared to traditional knowledge acquisition or requirements engineering, there is much more emphasis on acquiring knowledge that supports the autonomy and social ability of an agent compared to knowledge of the agents' specific tasks. An approach to agent-based knowledge acquisition is presented in (Iglesias, Garijo, González & Velasco, 1998). See (Iglesias, Garijo & González, 1999) for a survey of such approaches.) However, it is assumed that entities identified as agents in the multi-agent system exhibit the agent characteristics presented in Chapter 1: autonomy, reactivity, pro-activeness and social ability (Wooldridge & Jennings, 1995b).

[5] Merriam-Webster WWWebster Dictionary (http://www.m-w.com/).

It is also possible that specific entities in a multi-agent system are identified as subagents of another agent. In this case, it is assumed that subagents are themselves agents: a subagent must also be autonomous, reactive, pro-active and socially able. Only subprocesses of an agent for which these properties are important should be distinguished as subagents. On the one hand, subprocesses for which these properties are not important should not be considered agents. As stated above, a subagent is assumed to be an agent itself, and therefore, such subprocesses should not be distinguished as subagents. On the other hand, processes for which the properties mentioned above are important, are assumed to be to be distinguished as agents. As these processes are subprocesses of another agent, they should be distinguished as subagents to explicitly represent the hierarchical relation between agents in the multi-agent system. Often, processes can be distinguished that co-ordinate the activities of the (autonomous) subagents. These processes themselves may be associated with some of the subagents, as is indicated by the following example. Consider a team of agents. At a specific level of analysis, the team members are abstracted from and the team itself can be viewed as an agent: it is autonomous, reactive, pro-active, socially able and can be ascribed a mental state. (E.g., one can identify the intention of the team.) At another level of abstraction, the individual team members are identified and viewed as (autonomous) agents. Team members may exercise their autonomy to leave the team or to pursue goals that are incompatible with the team's intentions. Specific team members may adopt the goal of keeping the team together and establishing the team's intention as the joint intention of each member. A more detailed investigation of the relationship between mental attitudes of a team and of its members can be found in (Singh, 1998).

3.1.3 Processes and Components

Each process distinguished in a multi-agent system is represented by a component in the compositional system that models the multi-agent system. Processes identified in the multi-agent system are classified as either deliberation processes or environmental processes. A component that collates all deliberation processes of a specific agent in fact *represents* this agent in the compositional system. However, the guideline presented in this chapter does not *define* which components are agents and which are not. Instead, the guideline assumes that, as a starting point, an analysis of the multi-agent system has already identified which entities in the system are agents, and, as explained in the previous section, it is assumed that these agents exhibit agent characteristics such as autonomy, reactivity, pro-activeness and social ability (Wooldridge & Jennings, 1995b; see also Chapter 1). As a consequence, not only the agents in the modelled system exhibit agent characteristics, also the components that represent the agents do. However, these characteristics are not explicitly represented characteristics of the component construct or any other construct provided by the semantic structure.

3.1.4 Relations between Processes and between Processes and Agents or the Environment

It is assumed that three different types of relations can be identified in a multi-agent system, each of which is represented in a different way:

- There is a hierarchical relation between processes: a process may consist of other processes (the subprocesses of the process), which themselves may consist of other processes, and so on. The level of (process) abstraction determines which processes are considered to be primitive (no subprocesses are distinguished). Process composition is studied extensively in the area of Process Algebras (Bergstra & Klop, 1985; Milner, 1980; Hoare, 1978). Many languages for the specification of dynamics distinguish composition operators taken from the area of Process Algebra (Eck, Engelfriet, Fensel, Harmelen, Venema & Willems, in press). The hierarchical subprocess relation is identified with the subcomponent relation in the compositional system that models a specific multi-agent system.
- As stated before, a multi-agent system consists of a group of agents together with their common, shared environment. Processes in a multi-agent system either take place in the environment, or in one of the agents. The second relationship associates processes with the agent or the environment in which they take place. This relation is represented in a compositional system that models a multi-agent system as follows. At the highest level of abstraction, a compositional system that models a multi-agent system consists (solely) of one component for each agent and one component for the environment. At lower levels of abstraction, all subprocesses distinguished in the multi-agent system are represented as subcomponents of either one of the agent's components or of the environment component, according to whether they are associated with that agent or with the environment in the multi-agent system. As a consequence, the structure of a multi-agent system (different agents and the environment) is thus also represented in the compositional system.
- A process consumes information provided and produces new information, which can, in turn, be consumed by other processes. By providing information to another process, a process can influence the other process. This influence establishes the third relation between processes. In a compositional model of a multi-agent system, this relation is represented by information links between components. Interaction, which is characterised as mutual influence, is an important example of this relation.

As stated in Section 3.1.1, interaction is viewed as a relation between processes. With respect to interaction, two different aspects are distinguished. First, an agent performs deliberation processes that determine when and how to communicate, initiate actions or perform observations. These deliberation processes are modelled

as discussed above. The second aspect is the interaction proper, which is modelled by information exchange between components in the compositional system that models a multi-agent system.

Three forms of interaction can be distinguished: communication (information transmission from one agent to another), action execution, and observation. Information transmission from one agent to another resembles the information transmission construct presented in Section 2.2. Consequently, information transmission from one agent to another (communication) is represented by information transmission from one component to another in a straightforward manner.

In a compositional system model of a multi-agent system, action execution and observation of the environment are also represented by the information transmission construct presented in Section 2.2. This is possible because not only processes associated with agents are represented by components, but also processes associated with the environment. According to Pednault (1987), the effect of an action is "to cause the world to jump from one state to another". Thus, actions are executed because of their effect on the state of the environment. In a compositional model of a multi-agent system, action execution is assumed to be modelled in terms of the effect on the state of the environment. Under this assumption, the information transmission construct presented in Section 2.2 is applicable, as this construct represents information transmission in terms of the effect on the states of the components involved in the transmission. In other words, action execution is viewed as a form of information transmission to a process in the environment that carries out the effects of the action in the environment. Observation is viewed as information transmission from the environment to the agents. (In some domains, also so-called active observations are distinguished. An active observation is an observation that is explicitly initiated by an agent and thus comprises information transmission from the agent to the environment as well as information transmission from the environment to the agent).

In Chapter 9, the semantic structure developed in this thesis is applied to provide a semantics for the multi-agent modelling framework DESIRE. In DESIRE, knowledge structures used in components are explicitly represented, and composition relations for knowledge structures are defined. As a result, additional relations on processes are identified, such as a relation that determines which knowledge structures are used by which components (Brazier, Jonker & Treur, 1998).

3.1.5 An Example

The guideline is illustrated in Figure 3.1. The left half of Figure 3.1 depicts two agents in their environment (the world). The agent on the left transmits information to the agent on the right by means of a telephone call. The agent on the right interacts with the world (depicted by the large arrow). The gearwheels represent processes distinguished in the multi-agent system, some of which are

associated with the agents and another with the environment. In this figure, some processes associated with the agents are recognised as subprocesses (small gearwheels inside larger gearwheels). The right half of Figure 3.1 depicts a compositional model of the multi-agent system on the left. The top component represents the process associated with the environment. The two components below the top component represent the processes of both agents and contain subcomponents that represent the subprocesses depicted on the left half. Arrows to and from (the component that represents) the environment represent interaction between one agent and the environment. The arrow between the two agents (components) represents information transmission between the two agents. This figure depicts a possible way of representing agents and the environment. A discussion of different ways to represent interaction between agents and the environment is presented in Section 3.3.1.

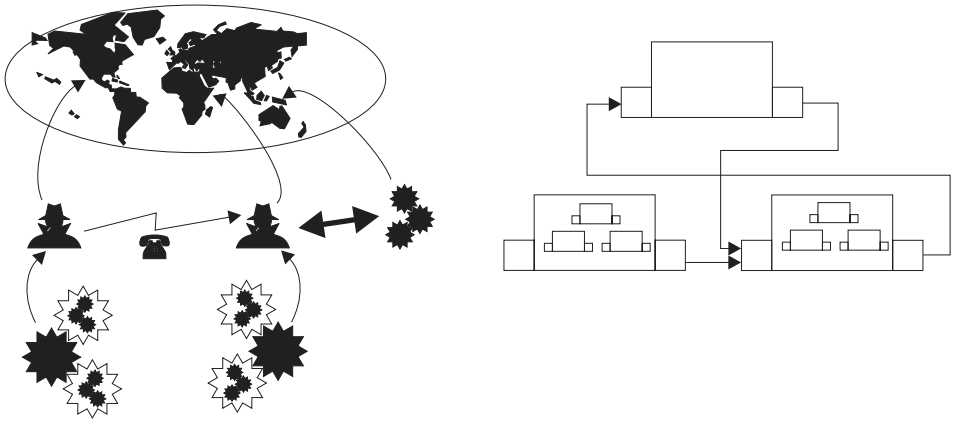

Figure 3.1: A multi-agent system modelled as a compositional system.

3.2 A Generic Compositional Agent Model

This section presents further guidelines for modelling a multi-agent system as a composition system. The guidelines are based on the concept of a generic agent model, which is introduced in Section 3.2.1. In Section 3.2.2, a specific generic agent model is described.

3.2.1 The Concept of a Generic Agent Model

As stated in the previous section, the main activity of an agent is deliberation, which is needed to acquire information by communication with other agents or by observation of the environment, to process information, and to perform actions or transmit the results to other agents. An agent's deliberation determine its autonomy, reactive and pro-active behaviour and its social abilities, the

characteristics that define an agent according to the weak notion of agents originally proposed by Wooldridge and Jennings (1995b) and discussed in Chapter 1.

This general characterisation may serve as a guideline for the identification of processes in a multi-agent system. For each agent in a multi-agent system, processes can be distinguished that perform the deliberation necessary for interaction with the environment, communication and processing information. A number of these processes are independent of the specific characteristics of a specific agent in a multi-agent system. This enables the use of a *generic agent model* as a starting point for modelling a specific multi-agent system. A generic agent model consists of components that represents top-level, generic processes that usually can be distinguished for an agent in a multi-agent system. Modelling a specific multi-agent system consists of *refining* the generic model. Refinement of a generic model involves *specialisation* and *instantiation*. Specialisation of a generic agent model entails identification of additional subprocesses of the processes identified in the generic model. These subprocesses are represented by additional subcomponents in the generic agent model. Instantiation of the generic agent model entails determination of further, specific characteristics of the processes identified in the generic model. By instantiation and specialisation, the generic agent model is transformed into a compositional model of an agent that can be used in a model of the complete multi-agent system. Results of these efforts, i.e. specific models, differ in the refinement and relative importance of the subcomponents.

3.2.2 Description of a Generic Agent Model

The current section presents a generic agent model called GAM. The generic agent model consists of processes that are not specific to an individual agent in a multi-agent system, but are, in principle, performed by each agent to manage communication, action execution and observation. In the current section, these generic agent processes are distinguished and described in more detail, leading to a compositional model of an individual agent.

The generic agent processes are related to the four characteristics required for the weak notion of agency described by Wooldridge and Jennings (1995b) and introduced in Chapter 1. In accordance with this notion, agents must (1) maintain interaction with their environment like observing and performing actions in the world: *reactivity*; (2) be able to take the initiative: *pro-activeness*; (3) be able to perform social actions like communication and co-operation: *social ability*; and (4) operate without the direct intervention of other (possibly human) agents: *autonomy*. In the generic agent model GAM depicted in Figure 3.2, these processes are each represented by a different component. Eight subcomponents are distinguished: Own Process Control (OPC), Maintenance of History (MH), Agent Specific Processes (ASP), Co-operation Management (CM), Agent Interaction Management (AIM), Maintenance of Agent Information (MAI), World Interaction

Management (WIM) and Maintain World Information (MWI). The names of the components are the same as in e.g. (Brazier, Jonker & Treur, 2000), with one exception: the component Agent Specific Processes was formerly known as Agent Specific Tasks. The term 'world' in the component names is synonymous with 'environment'. The term 'interaction' is qualified with either 'agent' or 'world' to indicate communication or action execution/observation, respectively.

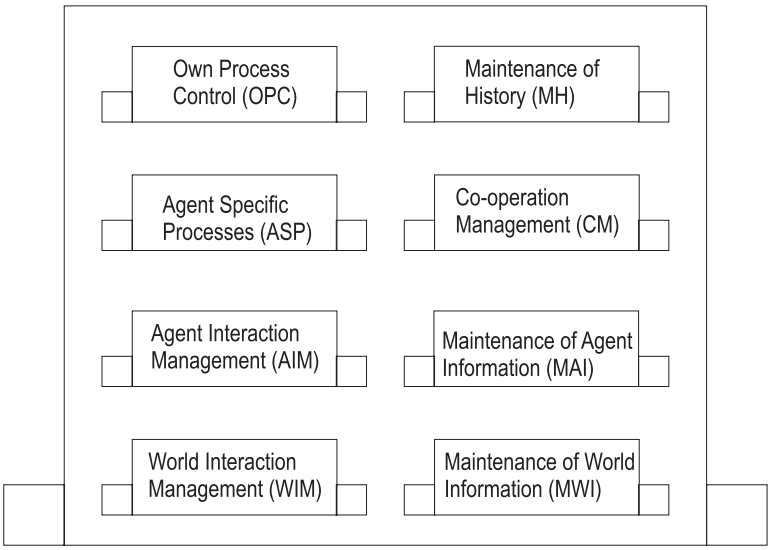

Figure 3.2: Top-level composition of the generic agent model GAM, from (Brazier, Jonker & Treur, 2000).

The following three points provide some background on the generic agent model:

- The correspondence with the four characteristics described above is as follows. Action execution and observation are performed by World Interaction Management, also using Maintain World Information. Social actions are managed by the processes Agent Interaction Management, Maintenance of Agent Information and Co-operation Management. Performing the agent's processes is co-ordinated by the component Own Process Control. This enables the agent to act autonomously and take the initiative if required. Most often, the eight subcomponents are further refined. This is illustrated in e.g. Chapter 10 and Chapter 11, in (Brazier, Dunin-Keplicz, Treur & Verbrugge, 1999), which presents a model of BDI agents designed as a refinement of GAM, and in (Brazier, Jonker & Treur, 1997), which presents a model for co-operation based on GAM. However, it is also possible that in specific agents, one or more of the generic components are not used.

- As stated in Section 3.1.2, knowledge acquisition is required to identify deliberation processes and environmental processes in a multi-agent system. The subprocesses distinguished guide agent-based knowledge acquisition, as the processes determine the various types of agent-specific knowledge required in addition to knowledge of the specific tasks an agent has been delegated. This includes (1) knowledge of an agent's priorities with respect to controlling its processes, (2) knowledge of which and how information is exchanged with other agents and the environment, (3) knowledge of how information received from the environment and other agents is to be analysed and (4) knowledge of how co-operative an agent is in given situations in relation to other agents.
- The compositional model depicted in Figure 3.2 is based on analysis (e.g., in the ARCHON project, see (Cockburn & Jennings, 1996; Brazier, Dunin-Keplicz, Jennings & Treur, 1997)) and considerations regarding the properties an agent should have as described above. This compositional model is the starting point for the models presented in Chapter 10 and Chapter 11. Moreover, for instance in (Brazier, Dunin-Keplicz, Jennings & Treur, 1997), the same model is applied to agents performing a process diagnosis and control task, with a minor shift in relative importance of the subprocesses.

3.3 Modelling Choices

As stated in the introduction of this chapter, the guideline for modelling a multi-agent system provides a considerable degree of freedom with respect to exactly how a multi-agent system is modelled. This section presents a number of issues deliberately left open by the guidelines presented in Section 3.1 and Section 3.2 and discusses various alternative ways of resolving these issues. As stated in the introduction, no commitment to specific alternatives is made. Such commitments are neither necessary for the further development of the semantic structure nor desirable, as the best alternative likely depends on the requirements imposed by a specific application.

3.3.1 The Environment and Interaction

As stated in Section 3.1.4, action execution and observation are represented by information transmission between a component that represents an agent and a component that represents the environment. This is possible because not only processes associated with agents are represented by components, but also processes associated with the environment. Section 3.1 assumes that processes identified in a multi-agent system can be classified as deliberation processes (which are internal to agents) and environmental processes. This classification is expected to be straightforward for most processes. However, depending on

properties of a specific multi-agent system and on the goal of the modelling effort, for some processes the classification may be more difficult. This holds in particular for processes that execute actions that affect other agents.

From the perspective of an agent, all other agents appear as entities in the environment it shares with the other agents. Consequently, it is, in principle, possible to observe other agents and to initiate actions that (directly) affect other agents. (E.g., hitting other agents.) In many multi-agent systems, however, this aspect of other agents can be abstracted from, because in such a multi-agent system, agents only communicate with each other and are not interested in executing actions upon one another or observing each other. In this case, agents are not represented in the environment. Instead, only non-agent entities are represented in the environment.

From the perspective of a single agent, also specific aspects of the agent itself are present in the environment, as an agent is not only a collection of mental deliberation processes (the mind), but also matter that constitutes the location of the mind. In many multi-agent systems, it is also possible to abstract from these material aspects. However, in some cases it is necessary to represent actions executed upon the agent, such as e.g. being hit by another agent (or the agent itself). In these cases, the material aspects of an agent and its mind may influence each other (e.g., brain damage and psychosomatic diseases).

Whether material aspects of agents are modelled depends on properties of a specific multi-agent system, together with the goal of the modelling effort. In this section, architectures for two alternatives are sketched. The most complex alternative presented covers mutual influence of an agent's mind and matter. This alternative is taken from (Jonker & Treur, 1997), which studies the interaction between an agent's mind and matter in great detail.

The first alternative, depicted in Figure 3.3, is adopted from Figure 10 in (Jonker & Treur, 1997). The left half of Figure 3.3 depicts an environment with one agent (Agent A) and one non-agent object (a car). Processes relating to material aspects of Agent A as well as its mental processes are distinguished. The processes relating to material aspects of Agent A are considered to be subprocesses of a process that collates all processes associated with the environment. Other subprocesses of this process are e.g. the processes executed by the car. The multi-agent system is represented by the compositional system depicted on the right half. The component labelled Agent A represents the mental processes of Agent A. The component labelled Environment represents the process that collates all processes associated with the environment. Two subcomponents are distinguished in the environment. The component labelled C represents the processes relating to the car. The component labelled A represents the processes relating to the material aspects of Agent A. The link labelled 1 is used to represent the influence of the material processes of A on its mind. The link labelled 2 transmits observation results from the environment to Agent A. The mediating link connected to this link and starting at component C can be used for observations of the car. (E.g.,

information on the speed of the car can be transmitted via this mediating link and link 2 to Agent A.) Link 3 is used by Agent A to initiate actions in the environment. The link is connected by a mediating link to component C. This mediating link can be used to execute actions upon the car (e.g., starting it). The link labelled 4 represents the influence of Agent A's mental processes on its material aspects.

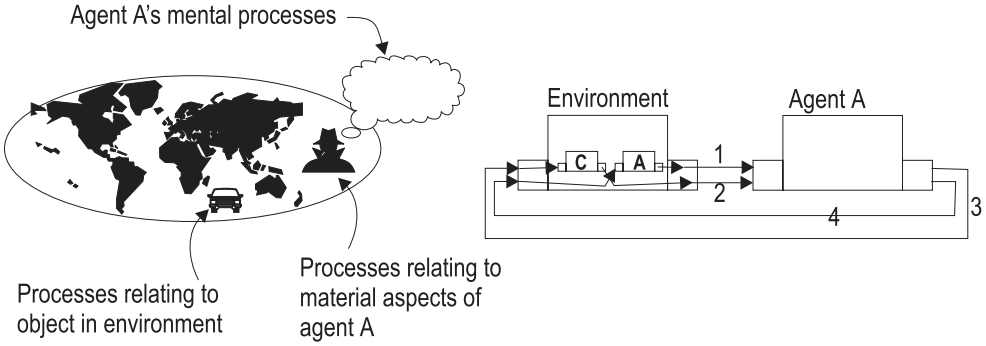

Figure 3.3: Modelling mind and matter.

The second alternative, which is more common, is a less detailed version of the model depicted in Figure 3.3. If the properties of the multi-agent system nor the goals of the modelling effort require explicit modelling of Agent A's material processes, the component labelled A as well as the links labelled 1 and 4 can be omitted from the model.

3.3.2 Observability of Actions and Processes

A common connotation of environmental processes is that these processes are, in principle, observable (and deliberation processes are not). This does not imply, however, that all processes in the environment are unconditionally observable for all agents in a multi-agent system. (As an example of a multi-agent system in observability of processes in the environment, consider the multi-agent system modelled in Chapter 11. This system consists of a society of 30 relatively simple agents that wander about in search of food, and, depending on their character, may choose to help other agents in finding food. Each agent has a limited range of vision. Processes that take place in a part of the environment too far away cannot be observed.) Whether all processes are observable for all agents in the environment is a property of a specific multi-agent system and thus varies between models.

As explained in Section 3.1.4, action execution and observation are represented in a compositional model by the construct for information transmission provided by the semantic structure. This construct only supports point-to-point information transmission (the commitment presented in Section 2.2.2). Therefore, action execution and processes are only observable for components for which there is an

information link that connects them to these processes. Thus, by carefully choosing which components in the environment to connect to which agent components, it is possible to model e.g. some processes as observable to specific agents and other components as observable to other agents. (Another possibility is to connect all agent components to the environment component, using the *state* of links to *dynamically* determine which processes are observable to specific agents.) As processes in the environment are affected by action execution, the same flexibility can be applied to choose which action executions are observable and which are not.

As a result of the flexibility with respect to whether processes are observable, there is an abundance of modelling choices. The following two alternatives might serve as general principles for the observability of action execution. The first alternative is to assume that an action always results in an observable change of the world state. This assumption is controversial. Consider for instance the situation in which two agents, A and B, both need to acquire exclusive access to a certain resource. First, A and B simultaneously observe the environment and find that the resource is still free. Then both try to take the resource. Assume that A is faster and thus acquires the resource, while B does not. However, both may observe that the resource is no longer free, so both infer that the action has been successfully performed. B's action only appeared to have resulted in an observable change of the world state, because in a sense B observed the situation before A acquired the resource and missed the fact that this has happened.

As an aside, a decentral model for mutually exclusive access to a shared resource is presented in Chapter 10.

The second alternative is to assume that an action is considered to have been successfully performed even if the world state has not changed. When the second option is used, it is not possible to determine whether the execution of an action is completed by observing the state of the world. In this case, it should be possible to observe the execution of an action itself.

3.3.3 Communication as Action Execution

As stated in Section 3.1.4, information exchange between agents (communication, one of the basic activities of an agent) is represented by information links between components that represent agents. A basic connotation of information transmission is that both the transmission and the information transmitted are, in principle, only observable by the agents involved in the transmission. The information transmission construct provided by the semantic structure guarantees that this is the case. (The commitment to point-to-point transmission presented in Section 2.2.2). However, there is an alternative way to represent communication. This alternative treats communication as an action in the environment and thus emphasises material aspects of communication. In the following circumstances, this alternative seems most suitable. First, in some multi-agent systems, there are forms of communication for which it is important to explicitly represent the way in

which the communication takes place. Consider, for example, the multi-agent system presented by Jonker, Treur and Wijngaards (2000), who study multi-modal (verbal and non-verbal) communication. In this system, all communication is modelled as action execution (by one agent), followed by observation (by the other agent). This enables representation of disturbances by environmental influence and of non-verbal communication by manipulating objects in the environment. Second, in some circumstances, the multi-agent system modelled requires that agents not involved in the communication are able to observe the communication. An example could be a model in which fraudulent agents are modelled that eavesdrop on communication. In addition, broadcast communication can be modelled as an action in the environment. As is explained in Section 2.2.2, the semantic structure does not provide constructs for broadcast communication. However, broadcast communication can be represented as an action in the environment observable for all agents.

Chapter 4

Overview and Running Example

Chapter 2 introduced the semantic structure, presenting a number of commitments that together determine which constructs are provided. After that, Chapter 3 presented a guideline for applying the semantic structure to model multi-agent systems. The next step in the development of the semantic structure, as stated in Section 1.4.2, is to formally describe the constructs and relations between constructs. Chapter 5 to Chapter 8 provide an elaborate and detailed mathematical description of the constructs that comprise the semantic structure. This chapter, Chapter 4, focuses on two topics that set the stage for this mathematical description in the next four chapters. First, Section 4.1 presents an overview of the mathematical description, which focuses on the mathematical concepts employed and on how these concepts are employed to meet the requirements put forward in Section 1.4.3. Second, Section 4.2 introduces a multi-agent system that is used as a running example in Chapter 5 to Chapter 8.

4.1 Overview of the Semantic Structure

Section 1.4.3 states a number of requirements for the semantic structure. The requirements can be summarised as follows. First, the semantic structure should support modelling the main activities of an agent, which are deliberation and interaction. Second, the semantic structure should be compositional, i.e. components are the most important construct provided by the semantic structure. Moreover, the dynamics of a compositional system (a system consisting of components) should be defined in terms of the semantics of the components that constitute the system together with a composition relation. Third, dynamics of a multi-agent system should be described in terms of the state of an agent. Fourth, the semantic structure should support a local view on dynamics. As many multi-agent systems are not centrally designed or managed, it should never be necessary to acquire a global picture of the structure or state of the system. However, it should be possible to compose a more global perspective from different local perspectives. (Thus, this requirement is also related to the compositionality requirement.) The way in which these requirements are addressed is explained in the next four subsections, which correspond to the next four chapters.

4.1.1 Constructs Provided by the Semantic Structure

Chapter 5 describes the main constructs provided by the semantic structure: components and information links. First, static aspects, and after that, dynamic aspects of these constructs are formalised.

4.1.1.1 Static Aspects

For a component, as stated in Chapter 2, three aspects are distinguished: the information state, or state, for short, of a component, its interfaces and its composition structure. The mathematical description of these three aspects together constitutes the component construct in the semantic structure.

A component is a locus of computation and information. The computations executed by a component modify the information contents of the component. The state of a component is determined by the information contents of the component. In a compositional system, components are likely to contain extensive data structures that together contain the information present in a component. These data structures enable the identification of substates, substates of substates, and so on, each of which is determined by a different data structure in the component. In fact, in many approaches, e.g., Troll (Jungclaus, Saake, Hartmann & Sernadas, 1996), the state of a component is defined in terms of the data structures it contains. However, the semantic structure developed in this thesis abstracts from the internal structure of a component's information (except for the identification of input, internal and output substates). Instead, it is assumed that at each moment in time, the state of a component can be identified and that for each component, a set of states (or, identifiers of states), is given. Moreover, in Chapter 5 to Chapter 8, in which the semantic structure is developed, no language is defined to specify sets of states. Instead, the semantic structure is developed without any reference to syntactic constructions. However, in Chapter 9, formal languages are developed that enable precise specification of sets of states and other sets that are assumed to be given, in the context of an application of the semantic structure.

As stated above, it is assumed that for *each* component, a set of states *of that component* is given. Such states are called *local states* to emphasise that these states are only determined by the information contents of a single component, and by nothing outside that component. To support locality, the semantic structure as defined in Chapter 5 does not refer to a notion a global state, that is, a state determined by the information contents of more than one component. Instead, a more global view of dynamics within a compositional system is defined in a compositional way, as compositions consisting of local states.

The other two aspects are formalised as follows. For each component, three substates of the state of the component are distinguished: the input, internal and output substates. These substates are determined by the information contents of the input and output interfaces of the component, and the component itself. The compositional structure of a component is formalised by a tuple, called a *structure*

hierarchy, consisting of a set of component identifiers, a set of link identifiers, a relation that represents the hierarchical structure of components, their subcomponents, the subcomponents of subcomponents, and so on, and two functions that map each link to the component or link connected to each of its end points.

The formalisation of information links is largely determined by the commitments presented in Section 2.2. The basic idea is that information transmission from one component to another establishes a relation between possible states of the two components. This idea appears both in the static aspects of an information link as well as the dynamic aspects. From a dynamic point of view, information transmission establishes a relation between the behaviour of both components. This is discussed in Section 4.1.2. The static aspects of an information link consist of which states of the link itself are distinguished (commitment discussed in Section 2.2.3.2), the components to which it is connected (commitment discussed in Section 2.2.3.1) and how the link is intended to relate the behaviour of the two components it connects. Thus, similar to a component, for a link a set of local states is distinguished. The state of a link is determined by the state of information transmission as a process (e.g., a link can be busy transmitting information, or waiting for information to transmit) and possibly by the contents of the link (messages in transit). However, as for component states, the semantic structure abstracts from the internal structure of states. Instead, it is assumed that for each link, a set of states is given. In addition to this set, a relation, called the *information link mapping*, is distinguished. As the information link mapping describes the intended relation established between the two components it connects, the information link mapping is defined on the state sets of these two components. States in the state set of a component are states that may or may not occur in the behaviour of a component. Thus, an information link mapping may describe the relation between two components by reference to states that do not actually occur in each behaviour of the components.

As an example, consider a compositional system with a link I which transmits information from a component D to a component C. The state set of D contains a state identified by S_1. The state set of C contains a state identified by S_2. Suppose that the following requirement is imposed on the compositional system: if D reaches state S_1, then C should reach state S_2. This requirement is part of the static aspects of I and is described by its information link mapping. The formalisation of information link mappings is presented in Section 5.1.2.

4.1.1.2 Dynamic Aspects

After the formalisation of static aspects presented in the first half of Chapter 5, the second half presents the formalisation of the dynamic aspects. First, the local behaviour of each component and link is defined. The local behaviour of a component or link consists of a set of alternative behaviours that the component or

link could exhibit if it were to exist in isolation. In other words, the local behaviour is the behaviour from a strictly local point of view, neglecting constraints imposed by a component's or link's relation with other components or links. The local behaviour of a single component or link is defined as a set of so-called local component or link traces, which are linear or branching structures of consecutive information states of the component.

Second, given the local behaviour of a set of components and links, the actual behaviour of a component or link in relation with other components and links is defined as a structure consisting of elements from the local behaviours of its constituents. The basic idea is that information transmission *constrains* the local behaviour of the components involved in the transmission, and that information transmission induces a *relation* between local traces of components that defines the constraints imposed by information transmission. Such a relation, called a *compatibility relation*, provides a more global view of co-operating components. (However, no global state and/or global language is assumed.) The basic idea is employed as follows.

For a component, three sets of traces can be distinguished, as depicted in the top half of Figure 4.1 below (in which traces are assumed to be linear). The first set is the set of traces consisting of all possible combinations of states of a component. This set includes traces that, in practice, can never be acquired, because they do not fulfil the specification of the behaviour of the component. For instance, consider a component B that is able to provide, upon request, a service 's' that, at some point in time, places new output in the output interface of B. The set of all possible traces of component B includes traces in which, after receipt of the request for a service 's' in the input interface, the result is never placed in the output interface of B.

A subset of the set of all possible traces is the set of local component traces: those traces that could, in principle, be acquired because they fulfil the specification of component behaviour. These traces are called local component traces to emphasise the fact that they are only part of a component's behaviour from a purely local point of view, in which constraints imposed by interaction with other components are not taken into account. For instance, a trace in which a service request 's' is visible in the input interface, and in a following point, the result of this service is visible in the output interface, is a local component trace. However, this trace might not be an actual behaviour of component B in a compositional system: it can only be an actual behaviour if another component, A, generates a request for service 's' and if this request is actually transferred to B. Therefore, in a structure that models the behaviour of the system consisting of component B, a component that requests B's service and interaction between A to B, such constraints should be represented.

As stated above, the behaviour of a composition of components and links is defined in terms of the local behaviour of the constituents. To represent constraints imposed by information transmission, compatibility relations are defined between traces of components that exchange information. Only local component traces and

link traces that are related by compatibility relations can constitute the dynamic structures associated with components. Thus, a local component trace of a component that transmits information to another component is part of the *actual overall behaviour* of the system only if it respects constraints imposed by information transmission. Only compatible local component and link traces are included in multitraces that model the behaviour of compositional systems. Only local component and link traces that respect information transmission are compatible.

As an example, consider a system with components A and B introduced above, and an information link between A and B. The information link automatically relays a request for B's service from A to B. In this example, a trace of component A in which event 's' is in A's output interface is compatible with traces in which event 's' is in B's input interface (and the result of service 's' is at some point in time generated at its output interface).

The bottom half of Figure 4.1 depicts similar sets of components of another component. As indicated in the picture, these sets are, in general, disjoint from the three sets of components of component A. An information link from component A to B establishes a compatibility relation between local component traces of both components. The two sets of compatible local component traces are the traces related by compatibility.

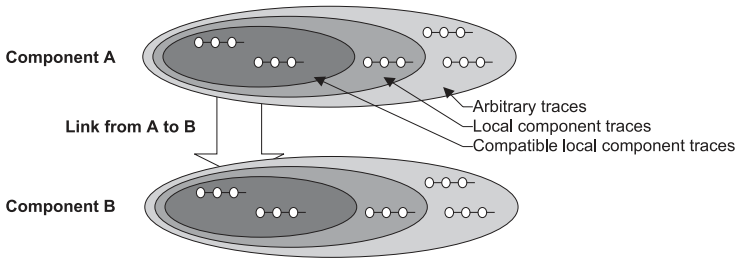

Figure 4.1: Sets of component traces.

In fact, three views on the behaviour of compositional systems are distinguished, called the black box, white box and glass box views. From a bird's-eye view, the structures that constitute these views consist of local component traces and link traces of a (possibly composed) component itself, possibly its subcomponents and its links, and possibly their subcomponents and links, and so on. All three views on behaviour developed in this section are relative for a given structure hierarchy and collection of compatibility relations. In a *black box view*, the behaviour of *C* is defined as a set of local component traces of *C* only. As a consequence, in the black box view the behaviour of subcomponents and links is not visible (although their behaviour is taken into account in the definition of the black box view to determine which local component traces of *C* constitute behaviour if information exchange is taken into account). In a *white box view*, the

behaviour of a component not only consists of local component traces of this component, but also of local component traces of its subcomponents. In a *glass box view*, the behaviour of a component consists of local component traces of this component, of local component traces of its subcomponents and of local component traces of the subcomponents of these subcomponents, and so on.To summarise, the two most important formal notions are the structure hierarchy and the glass box view on the behaviour of a component. A structure hierarchy enables defining an entire compositional system, while the glass box view defines the most complete picture of the behaviour of such a system. The definition of the glass box view, however, is relative to among others a collection of compatibility relations. Locality and compositionality play an important role in the formalisation of the dynamics of a compositional system. The starting point for describing the dynamics of a compositional system consists of structures (local component and link traces) that describe the behaviour of a component or link from a strictly local point of view. Three views on the dynamics of compositions of components and links are then defined as compositions of local component and link traces. Only local component and link traces that obey constraints enforced by information transmission can be part of these compositions. The constraints enforced by information transmission are represented by compatibility relations, which are briefly described in the next section, Section 4.1.2, and defined in Chapter 6

4.1.2 Interaction: Locality and Compatibility

The definition of properties of compatibility relations is the topic of Chapter 6. Compatibility relations are introduced in Chapter 5 as ternary relations on the set of traces of two components or links and a link. Any relation on the sets of local component traces of the domain of a link, the link and its co-domain is a possible compatibility relation. In Chapter 5, no properties of compatibility relations are defined.

In Chapter 6, specific classes of compatibility relations are defined in terms of the properties that compatibility relations in a specific class exhibit. These properties are expressed in terms of transmission octets. Transmission octets provide a bridge between, on the one hand, 8-tuples of states that occur in local component and link traces of a link I, its domain and co-domain, and, on the other hand, the (static) declaration for the information link I of how information transmission affects the states of the components and links involved. This (static) declaration itself is given by the information link mapping of I. An information link mapping does not refer to states that occur in component or link traces. Instead, as the following definition indicates, an information link mapping is directly defined in terms of the sets of all states of I, its domain and co-domain.

Figure 4.2 illustrates compatibility relations and transmission octets by zooming in on Figure 4.1. A triple consisting of a trace for a component A, for a link L from A to a component B and for B is shown. Within this triple of traces, octets of eight states (two from the trace for A, four from the trace from L, and two from the trace

for *B*) can be found that respect information transmission as depicted in Figure 2.7. Such an octet is a transmission octet, and is depicted in Figure 4.2 by black elipses. The triple of traces is part of a compatibility relation with a specific property if transmission octets can be found for a specific subset of all states in the traces of the triple. (E.g., for all states in the trace for *A*, a transmission octet can be found.)

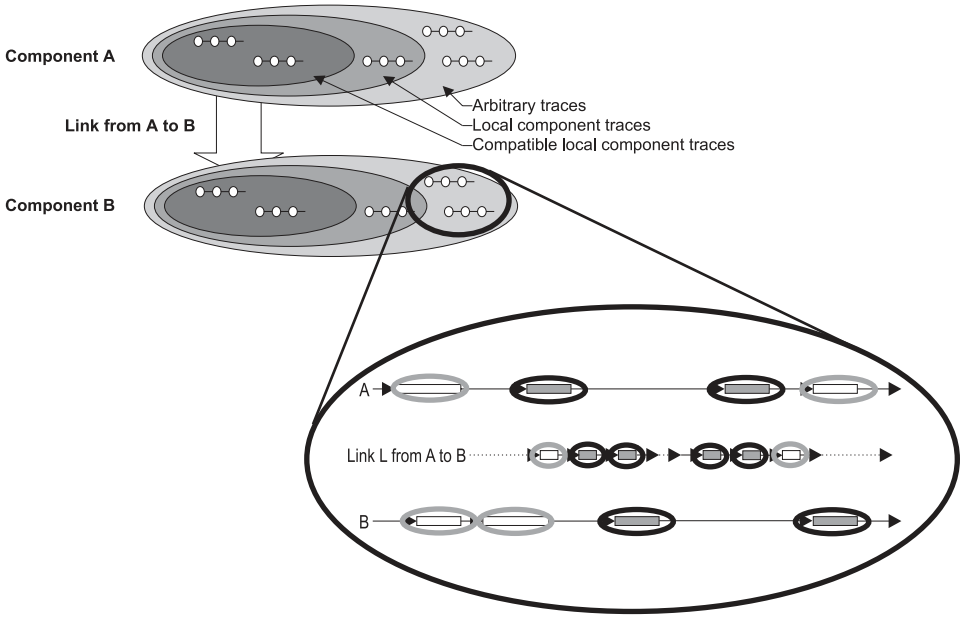

Figure 4.2: A transmission octet.

Transmission octets are employed to define properties that a compatibility relation, and thus information exchange, may exhibit. Four properties have been defined so far: the lossless transmission property, the order-preserving transmission property, the asynchronous transmission property and the logically instantaneous transmission property.

4.1.3 A Global Perspective

As explained in Section 4.1.1.2, the dynamics of a compositional system are described by compatible multitraces, which are indexed sets of local component and link traces. Three views on the dynamics of a compositional system are defined. Given a structure hierarchy that describes a compositional system, the glass box view provides the view with the most detail: the multitraces that constitute the glass box view contain local component and link traces for each component and link in the structure hierarchy. In this sense, the glass box view provides a global perspective on the behaviour of the compositional structure described by the structure hierarchy.

However, the global perspective provided by the glass box view consists of local component and link traces, each of which consists of local component and link states. Thus, this global perspective does not describe global dynamics in terms of a notion of a global state. However, in some cases, it is necessary to have available a description of the global dynamics in terms of the global state of a compositional system.

A common way to define a global state is as a composition of the local states of all components and links at the same moment. However, this definition relies on the availability of global time to determine 'the same moment' for all components and links. As stated in Chapter 1, this thesis assumes that global time is not available. In Chapter 7, a different way to define global states is developed.

In Chapter 7, a global state is defined in terms of *snapshots*. A snapshot is itself relative to a compatible multitraces for a structure hierarchy *SH* as introduced in Section 4.1.1.2. A snapshot is a function from the set of components and links in *SH* to the (disjoint union of) the sets of states of all components and links, such that the snapshot selects exactly one local component or link state from each local component or link trace in the multitrace. Figure 4.3 depicts a snapshot.

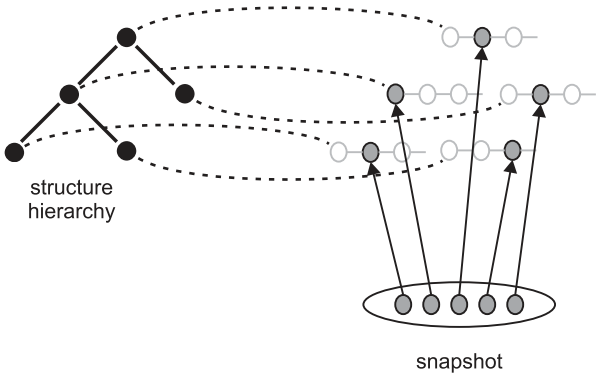

Figure 4.3: Structure hierarchy.

A snapshot is thus an arbitrary selection of states from a multitrace. Not every snapshot of a multitrace for *SH* represent a global state of the behaviour of the compositional system described by *SH*, because in an arbitrary selection of states, possibly for a component *A,* a state is selected in which information is available that is, according to the state of another component *B* in the same snapshot, not yet sent. In other words, the selected state of *A* depends on the occurrence of state *B*. Dependence was first introduced (in an event-based context) by Lamport (1978). In the semantic structure developed in this thesis, dependence is not a primitive notion. Instead, it is defined in terms of transmission octets. Loosely speaking, a state σ_A of component *A* depends on a state σ_B of component *B* if either (1) *A*=*B* and σ_B occurred at an earlier time point in the trace of *A* than σ_A, or (2) in σ_A,

information is received that is, according to a transmission octet, made available in σ_B (in other words, σ_B and σ_A are the first and eighth state of a transmission octet, respectively), or (3) there is a state σ_C of a component C such that σ_A depends on σ_C and σ_C depends on σ_B.

Two states σ_S and $\sigma_{S'}$ are independent if neither σ_S depends on $\sigma_{S'}$ nor $\sigma_{S'}$ depends on σ_S. States that are independent can possibly occur simultaneously. In Chapter 7, a global state is defined as a snapshot such that for two different components or links S and S', the states σ_S and $\sigma_{S'}$ in the snapshot are independent. In this definition, it is not assumed that global time is available. Instead, the definition only refers (indirectly) to information transmission as represented by transmission octets.

Even if the traces in a compatible multitrace for a structure hierarchy SH are linear, the set of global states as introduced above has the structure of a partial order. This partial order of global states constitutes the global perspective on the behaviour of the compositional system represented by SH. The partial order of global states can also be viewed as a transition system. Starting from the global state consisting of all initial component and link states, the compositional system proceeds as follows. Each component or link is either involved in updating its internal state, transmitting information to another component or link, or receiving information. Each of these activities results in transitions from one local state to the next local state. In the transition system, the local states in the global state are updated accordingly.

4.1.4 Control

As stated in Chapter 1, agents are socially able. Consequently, in almost every multi-agent system, agents try to influence other agents (by information transmission) to satisfy goals they cannot satisfy without help. Likewise, in a compositional system, some processes try to influence other processes. In other words, processes in a compositional system and agents in a multi-agent system try to exercise *control* over other components or agents, respectively. Chapter 8 is devoted to the representation of this phenomenon, the phenomenon of control, in the semantic structure.

Chapter 8 starts with a characterisation of the control phenomenon. Control (at least, phenomena with that name) is encountered in programming language design, but also in reactive Artificial Intelligence systems that assist in operating, for instance, a power plant. However, all these instances of the control phenomenon share a common characteristic: they repeatedly carry out a process that can be described conceptually as follows. A component that wants to exercise control over another component builds a descriptive model of the past and present behaviour of the component that is to be controlled. It does so on the basis of information obtained from the controlled component. The controlling component extends the model of the controlled component such that the extension prescribes

the future of the controlled component. This prescriptive future part of the model presumably specifies the future as the controlling component would like it to be. On the basis of this prescriptive part, the controlling component determines which actions to take, or which information to transmit, to (try to) set the actual future of the controlled component as envisioned.

Chandrasekaran (1994) argues that everything that exercises control carries out a process to do so that can be described at a conceptual level as above. (Chandrasekaran does not imply that, e.g., a thermostat, which controls a heater, actually maintains a model of the heater and its behaviour. However, the process carried out by a thermostat can be conceptualised as if it does.)

The essential element in Chandrasekaran's characterisation of control is information transmission. A component is able to control another component (at least in principle) by the virtue of information transmission. Information transmission enables trying to set the future of a controlled component as well as observing the controlled component to evaluate the exercition of control. In terms of the semantic structure developed in this thesis, Chandrasekaran's characterisation of control shows that the constructs of the semantic structure as described in Section 4.1.1 and Section 4.1.2 suffice to represent control. However, there are some additional issues that have to be addressed.

First, the constructs presented in Section 4.1.1 and Section 4.1.2 do not enable the distinction of information transmission for control as a special form of information transmision. Chapter 8 introduces a number of refinements of the constructs presented in Section 4.1.1.1 that enable the separation of control processes (these processes are represented by components as usual) and information transmission for control. These refinements are technically straightforward and therefore not discussed in this section. It is important to note that Chapter 8 does not define which information is control information. Instead, Chapter 8 introduces facilities that enable applications of the semantic structure to designate specific information as control information.

Second, Chapter 8 discusses the relation between the behaviour of a control component and controlled components. Conform the general characterisation of control, a controlling component transmits information to a controlled component to (try to) set the future behaviour of the controlled component. Control processes differ with respect to the extent to which the future behaviour of the controlled component will be as specified. For instance, compare the thermostat and the president of the national bank. On the one hand, it may be expected that if a thermostat transmits information to a heater (by closing an electric circuit) stating that the heater has to start, the heater will indeed start (not taking into account malfunctions). On the other hand, if the president of a national bank tries to influence consumers' spendings, it may well be the case that his or her information is misunderstood, neglected or nullified by other information directed at the consumers. The semantic structure, nor the refinement presented in Chapter 8, commits itself to a specific extent to which received control information determines

the future behaviour of a controlled component. Instead, applications of the semantic structure can choose a level of extent suitable for the application. The final section of Chapter 8 discusses a few possible relations.

The semantic structure is fully defined in Chapter 5, Chapter 6 and Chapter 7. Chapter 8 presents formal definitions of a number of refinements of constructs defined in Chapter 5. (Chapter 8 also presents a number of additional commitments with respect to control and discusses how control in a multi-agent systems is represented in a compositional system.) Chapter 9 also contains a number of formal definitions. However, the constructs defined in Chapter 9 are not part of the semantic structure. Instead, they constitute the semantics of the DESIRE modelling framework. The formal definitions of the constructs that constitute the semantic structure are illustrated with a running example, which is described in Section 4.2.

4.2 Running Example

An example system in the area of intelligent Internet applications is used to illustrate the semantic structure developed in this thesis. This example system is a multi-agent system in which an information brokering agent serves user agents with information from information provider agents. Information brokering is introduced in Section 4.2.1. Section 4.2.2 presents a compositional system that models information brokering in a multi-agent system. This compositional system is the actual running example.

4.2.1 Information Brokering

The multi-agent system on which the example is based consists of broker agents that act as intermediaries between agents that provide information (on arbitrary resources) and agents that use this information. (Agents that use this information are customers of the broker agent. Provider agents can, however, also be viewed as customers: the broker agent presents information on their behalf. To make the distinction between the two clear, agents that use the information provided are called 'user agents'.) Figure 4.4 depicts the three types of agents. To keep the example concise, only one broker agent is depicted (called 'Broker'), together with two agents that provide information (called 'Provider 1' and 'Provider 2', respectively) and two users of the broker agent (called 'User 1' and 'User 2'). As indicated in Figure 4.4, the user agents exchange information with both the broker agent and the provider agents. The reason for this is that the broker agent only provides information on a resource, and not the resource itself. To obtain the resource itself, user agents contact the provider agents directly.

In fact, the terms used to describe the agents in the example system indicate the *roles* of the agents distinguished, and not their types. In other words, in the domain of information brokering, one can distinguish agents that play the role of information provider and others that play the role of information users. These roles

need not be the only roles played by the agents: for example, agents may also play roles that have nothing to do with the domain of information brokering and are thus not of importance in the context of this example. Moreover, one or more agents may have more than one role: a provider may present information obtained as a user of another broker agent. In the example system, however, an agent has one specific role and is named accordingly. Thus, for example, an agent with the role of a provider is called a provider agent.

The five agents depicted in Figure 4.4 are the only agents distinguished in the running example. Figure 4.4 deliberately depicts the agents as different forms of agents: the user agents appear to be humans, while the broker appears to be an automated system. The provider agents appear to be institutions or agencies (a store and a school). The running example, however, abstracts from the form of the agents: human agents are not distinguished from software agents, and subagents that comprise the agencies (e.g., store clerks, school teachers) are not identified. Moreover, the running example also abstracts from the means of communication used by the agent. Figure 4.4 suggests that the human agents access the broker agent via a public terminal. In this case, communication between the user agents and the broker agent consists of manipulations using the (graphical) user interface of the broker agent. However, it may also be the case that the users employ their own personal computers to access the broker agent.

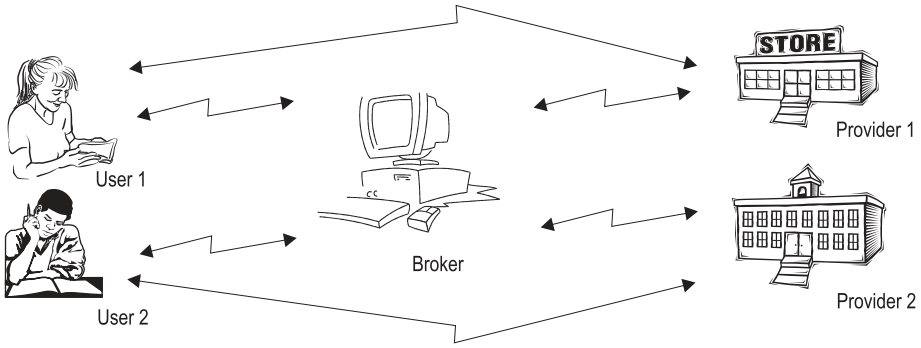

Figure 4.4: Information brokering.

The intended function of the broker agent is as follows. A user agent communicates to the broker agent that he/she is interested in information on a resource (i.e., a research paper, a WWW page, a product sold by means of e-commerce). A provider agent communicates to the broker agent descriptions of resources available to the broker agent, preferably using an internet standard such as the Resource Description Framework (RDF), see (Lasilla, 1998). The broker agent matches interests of users with information provided by the provider agents, in compliance with a number of requirements. To keep the running example concise, only one requirement is given:

- Once a broker agent receives a query from a user, information matching the query has to be communicated to the user at the next moment in time if this information was already available to the broker, or some time in the future in all other cases.

Many more requirements can be placed in the design of the multi-agent system. However, as the study of information brokering is not the topic of this thesis, and to keep the example concise, no further requirements or detail are provided. See (Jonker & Treur, 1998c), for a more complete investigation of information brokering. (In Chapter 8, an additional requirement is presented, which is used to illustrate control.)

4.2.2 Design of a Compositional System for Information Brokering

The semantic structure illustrated by the running example provides constructs for building compositional systems, such as the multi-agent system presented in Section 4.2.1. Applying the guideline put forward in the previous chapter to the example multi-agent system results in the compositional system presented in Figure 4.5. The compositional system depicted in Figure 4.5 consists of eight components:

- One component, called toplevel, of which all other components are subcomponents, or subcomponents of subcomponents, and so on. This component represents the complete multi-agent system.
- The components labelled user_1 and user_2 represent the user agents. (As stated in Section 4.2.1, the precise nature of these components is not important: user_1 and user_2 can, for example, represent human agents, their personal digital agents, or web browser applications serving as intermediaries between the user and the broker.)
- The component labelled broker represents the broker agent. For the broker agent, two subcomponents are distinguished: Agent Specific Processes (ASP) and Own Process Control (OPC). The subcomponent Own Process Control represents all subprocesses of the broker agent that control its behaviour. All other subprocesses, which are specific to the task of the broker, are subprocesses of the component Agent Specific Processes. In the next chapter, subcomponents of a subcomponent of toplevel makes the examples more interesting. In Chapter 8, OPC and ASP are used to illustrate control in a multi-agent system.
- The components labelled provider_1 and provider_2 represent the information provider agents. (Similar to user_1 and user_2, the precise nature of these components is not important.)

As stated in Section 4.2.1, the user agents, the broker and the information provider agents preferably employ an Internet standard such as the Resource Description

Format (RDF, Lassila, 1998) for communication. However, while this standard defines the format of resource descriptions, it allows different ontologies to be used for descriptions. The running example, in fact, assumes that on the one hand the broker and information providers and, on the other hand, the users use different ontologies.

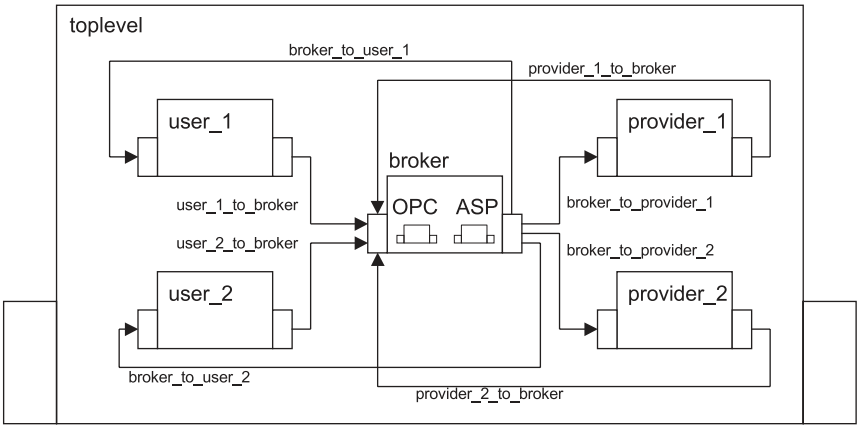

Figure 4.5: Information broker agent system.

Chapter 5

A Semantic Structure for Agent Dynamics

This chapter presents the semantic structure for agent dynamics. Section 5.1 formally defines the constructs provided by the semantic structure, focusing on static aspects of the constructs. The dynamic aspects are presented in Section 5.2. Section 5.3 summarises the semantic structure defined so far. Proofs of propositions presented in this chapter are given in Section 5.4.

5.1 The Constructs of the Semantic Structure

The main constructs provided by the semantic structure are components and information links. As stated in Section 2.1, three aspects of the first construct (components) are distinguished: the information state (i.e., the information contents of a component), the interfaces and the compositional structure in terms of subcomponents and links. The first two aspects are formally defined in Section 5.1.1. The second construct (information links) is defined in Section 5.1.2. The third aspect of a component, composition structure, is formally defined in Section 5.1.3.

5.1.1 Components, Interfaces and State

Assume that a set of components *Comp* (or, more precisely, component identifiers or names) is given. Elements of this set are typically denoted by capitals *C*, *D*, etc. As stated in Chapter 2, a dynamically changing information state is distinguished for each component. Each information state consists of input, internal and output substates. The input and output substates of an information state define the interface of a component, in the sense that these substates are accessible to other components (and the component itself) via information links. The current section assumes that for each component *C*, three non-empty sets of states $\mathcal{S}_{C,in}$, $\mathcal{S}_{C,int}$ and $\mathcal{S}_{C,out}$ are given (for the input, internal and output substates, respectively), without further commitment to the contents of these sets. (Chapter 9 provides a formal language to specify these sets.) The (overall) state of a component *C* (the first

aspect of a component distinguished in this thesis) is composed of elements of the sets $\mathcal{S}_{C,in}$, $\mathcal{S}_{C,int}$ and $\mathcal{S}_{C,out}$ as follows:

Definition 5.1. (Component information state). *Let C be a component. An* information state *of C is an element of* $\mathcal{S}_{C,in} \times \mathcal{S}_{C,int} \times \mathcal{S}_{C,out}$. *The set of all component information states of C is denoted* $\mathcal{S}_C$, *i.e.,* $\mathcal{S}_C = \mathcal{S}_{C,in} \times \mathcal{S}_{C,int} \times \mathcal{S}_{C,out}$.

Example 5.2. In the running example, the following sets are used for the user and broker agents. (As explained in Section 4.2.2, the user agents and the broker agent use different ontologies to describe resources. Therefore, two sets of ontology terms OT_1 and OT_2 are assumed to be given. (These set can, for instance, be sets of resource descriptions in the RDF (Lassila, 1998) format.) The set Q is a set of query terms, $Users = \{\text{user\_1}, \text{user\_2}\}$ and $Providers = \{\text{provider\_1}, \text{provider\_2}\}$. The user agents and the broker agent also maintain information about their own processes. For instance, the input interface of the user agents can be in a state in which it is ready to receive information. This is taken into account in the definition of $\mathcal{S}_{\text{user\_1},in}$ and $\mathcal{S}_{\text{user\_2},in}$ below. User agents and the broker agent may also internally maintain beliefs. For instance, the broker agent maintains beliefs about matches between user queries and information available via the providers.)

- $\mathcal{S}_{\text{user\_1},in} = \mathcal{S}_{\text{user\_2},in} = \{\text{communicated\_by}(t,\text{broker}) \mid t \in OT_1\} \cup \{\text{ready\_for\_information}\}$,
  $\mathcal{S}_{\text{user\_1},int} = \mathcal{S}_{\text{user\_2},int} = \{\varnothing\}$
  $\mathcal{S}_{\text{user\_1},out} = \mathcal{S}_{\text{user\_2},out} = \{\text{to\_be\_communicated\_to}(q,\text{broker}) \mid q \in Q\}$.
- $\mathcal{S}_{\text{broker},in} = \{\text{communicated\_by}(q,u) \mid q \in Q \text{ and } u \in Users\} \cup \{\text{communicated\_by}(t,p) \mid t \in OT_2 \text{ and } p \in Providers\}$,
  $\mathcal{S}_{\text{broker},int} = \{\text{belief}(\text{match}(t,q)) \mid t \in OT_2 \text{ and } q \in Q\}$
  $\mathcal{S}_{\text{broker},out} = \{\text{to\_be\_communicated\_to}(t,u) \mid t \in OT_2 \text{ and } u \in Users\} \cup \{\text{just\_communicated\_to}(t,u) \mid t \in OT_2 \text{ and } u \in Users\}$.

In this example, states are identified by elements of e.g. $\mathcal{S}_{\text{user\_1},in}$ such as $\text{communicated\_by}(t,\text{broker})$, where t is an element from the set of ontology terms OT_1. The elements of sets such as $\mathcal{S}_{\text{user\_1},in}$ resemble propositions about states. In fact, in Chapter 9, a logical language for propositions that describe states is presented. However, in this example, elements of sets such as $\mathcal{S}_{\text{user\_1},in}$ are (unique) names of states. These names have no internal structure. ■

As stated in Section 2.1, two kinds of components are distinguished: composed components and primitive components. The component information state definition given above applies to both kinds of components. In this thesis, the term 'internal (sub)state' is used in a restricted way: it refers to the state of the part of a component's information contents that is not visible to other components. In particular, the terms 'internal (sub)state' and 'internal information' do not refer to, and are thus independent of, the (input and output) states of the subcomponents and links of a composed component. In other words, the contents of a composed

component consists of subcomponents, links and the internal information contents as defined above.

5.1.2 Information Links

The second construct provided by the semantic structure is the information link. An information link transmits information from one component (called source component or *domain* of the link) to another, or possibly the same component (called the destination component or *co-domain* of the link). In this section, static aspects of this construct are formalised. As explained in Section 2.2.3.1, in the semantic structure, information links are first-class citizens; they are of the same standing as components.

It is assumed that a set of links *Lnk* (or, more precisely, link identifiers or names) is given. An element of this set is typically denoted with the capital *I*. The first static aspect identified is the *state* of the link (See Section 2.2.3.2). The state of a link is determined by (1) the state of information transmission as an activity: for instance, a link can be busy exchanging information, it can be waiting for new information to exchange, it can be enabled or disabled, and (2) the contents of the link, e.g. messages in transit. The semantic structure does not enforce a commitment with respect to the contents of link states, nor to the way in which the state of the link is described. Instead, for each information link *I*, a non-empty set of information link states $\mathcal{S}_I$ is assumed to be given.

Definition 5.3. (Link information state). *Let I be an information link. An* information state *of I is an element of a set* $\mathcal{S}_I$.

Example 5.4. In the running example, a link called broker_to_user_1 exists between broker and user_1. The set of link information states of this link is defined as follows (where *t* is a set of ontology terms):

$\mathcal{S}_{\text{broker\_to\_user\_1}} = \{\text{awake\_and\_empty}, \text{active\_and\_contents}(t) \mid t \in OT_2\}$ ■

The second static aspect of an information link is its *compositional relation* with components in a compositional system. In this section, the connection of a link with two components or links, one at each 'end point', is formalised. A more extensive definition of the compositional relation of a link is given in the next section.

Definition 5.5. (Domain and co-domain). *Let I be an information link. Two components or links, called the* domain *and* co-domain, *are relative to I. This is denoted by two functions, dom,cdom: Lnk*$\rightarrow$*Comp*$\cup$*Lnk. A link I with dom(I)=*S_1 *and cdom(I)=*S_2 *is called a link from* S_1 *to* S_2.

The third static aspect of an information link is the *information link mapping*. As explained in Section 4.1.2, an information link mapping is a relation between states from the state sets of the domain and co-domain of the link. In fact, an information link mapping is defined as an 8-ary relation to fulfil the commitments made in

Section 2.2.3.3, Section 2.2.5 and Section 2.2.6 (this is further explained below). As stated in Section 2.2.1, the semantic structure distinguishes six kinds of links. The information link mapping differs for each of the six kinds of links as follows:

Definition 5.6. (Information link mapping). *Let I be an information link. An information link mapping for I is a relation defined as follows:*

- $\lambda_I \subseteq (\mathcal{S}_{dom(I),out} \times \mathcal{S}_{dom(I),out}) \times \mathcal{S}_I^4 \times (\mathcal{S}_{cdom(I),in} \times \mathcal{S}_{cdom(I),in})$, *if I is a private link, or*
- $\lambda_I \subseteq (\mathcal{S}_{dom(I),in} \times \mathcal{S}_{dom(I),in}) \times \mathcal{S}_I^4 \times (\mathcal{S}_{cdom(I),in} \times \mathcal{S}_{cdom(I),in})$, *if I is an import mediating link, or*
- $\lambda_I \subseteq (\mathcal{S}_{dom(I),out} \times \mathcal{S}_{dom(I),out}) \times \mathcal{S}_I^4 \times (\mathcal{S}_{cdom(I),out} \times \mathcal{S}_{cdom(I),out})$, *if I is an export mediating link, or*
- $\lambda_I \subseteq (\mathcal{S}_{dom(I),in} \times \mathcal{S}_{dom(I),in}) \times \mathcal{S}_I^4 \times (\mathcal{S}_{cdom(I),out} \times \mathcal{S}_{cdom(I),out})$, *if I is a cross-mediating link, or*
- $\lambda_I \subseteq (\mathcal{S}_{dom(I),out} \times \mathcal{S}_{dom(I),out}) \times \mathcal{S}_I^4 \times (\mathcal{S}_{cdom(I)} \times \mathcal{S}_{cdom(I)})$, *if I is a link modifier link, or*
- $\lambda_I \subseteq (\mathcal{S}_{dom(I)} \times \mathcal{S}_{dom(I)}) \times \mathcal{S}_I^4 \times (\mathcal{S}_{cdom(I),in} \times \mathcal{S}_{cdom(I),in})$, *if I is a link monitoring link.*

The intended meaning of an information link mapping is explained with reference to Figure 2.7. In Figure 2.7, information transmission by a link L from a component A to a component B is depicted by eight states. (Thus, $dom(L)=A$ and $cdom(L)=B$.) These eight states correspond to an element $\langle\langle \nu_{A,i}; \nu_{A,j}\rangle; \langle \nu_{L,i''}; \nu_{L,j''}; \nu_{L,k}; \nu_{L,l}\rangle; \langle \nu_{B,i'}; \nu_{B,j'}\rangle\rangle \in \lambda_L$. (In this thesis, tuples are delimited by angular brackets and their elements are separated by semicolons.) This element states that (the numbers between parentheses refer to the explanation below):

If component A reaches state $\nu_{A,i}$ (1), *and* link L is in state $\nu_{L,i''}$ (2), *and* component B is in state $\nu_{B,i'}$ (3),

then component B should reach state $\nu_{B,j'}$ (4) as one of the successors of $\nu_{B,i'}$ (5), and one of the following states of A should be $\nu_{A,j}$ (6),

and state $\nu_{L,j''}$ should be the first state of L in which the information is in transit, *and* state $\nu_{L,k}$ should be the last state of L in which the information is in transit, *and* state $\nu_{L,l}$ should be the first state of L in which the information is no longer in transit (7).

Parts (1) and (4) form the most important part of this expression and as such are part of the informal explanation in Section 4.1.2. Parts (1) and (4) show that in the semantic structure, information transmission is characterised in terms of states of the components that exchange information. Condition (2) is related to the commitment presented in Section 2.2.3.2: state $\nu_{L,i}$ enables the expression of a requirement on the state of the information link to enable transmission, e.g., the expression that the link must be in an enabled state. Condition (3) is related to the

commitment presented in Section 2.2.6 and enables expression of an enabling condition for receipt imposed by the destination component. Parts (5) and (6) reflect the commitments to non-blocking receive (Section 2.2.6) and send (Section 2.2.5), respectively. Part (7) enables expression of the dynamics of transmission as a process as explained in Section 2.2.3.3. The sequence of states can, for instance, be used to describe that a message is put in the link (change from $v_{L,i''}$ to $v_{L,j''}$) and later taken from it (change from $v_{L,k}$ to $v_{L,l}$). An information link mapping does not, in general, need to be functional in all of its arguments, and is therefore formalised using a relation.

Example 5.7. The information link mapping of the link broker_to_user_1 is defined as follows (where *trans* is a function from OT_2 to OT_1 that translates ontology terms in OT_2 to OT_1, which is assumed to be given):

$\lambda_{\text{broker\_to\_user\_1}}$ = {⟨⟨to_be_communicated_to(*t*,user_1);just_communicated_to(*t*,user_1)⟩;
⟨awake_and_empty;active_and_contents(*t*);active_and_contents(*t*);
awake_and_empty⟩;⟨ready_for_information;
communicated_by(*t'*,broker)⟩⟩ | $t \in OT_2$, $t' \in OT_1$ *and* $t'=trans(t)$}.

This information mapping specifies that if in broker's behaviour, there is a state to_be_communicated_to(*t*,user_1), and the state of link broker_to_user_1 is awake_and_empty, and in user_1's behaviour, there is a state ready_for_information, then a transition of user_1's input interface state to the state communicated_by(*t'*,broker) exists, for resource description terms such that $t'=trans(t)$. Moreover, in broker's behaviour, one of the successor states is the state just_communicated_to(*t*,user_1), and the state of link broker_to_user_1 changes to active_and_contents(*t*) and then back to awake_and_empty. (To keep the example simple, it is assumed that link broker_to_user_1 can only transmit a message if no other messages are in transit.) ■

An information link mapping provides a static description of the relation between two components and a link as a result of information transmission. The (informal) meaning of an information link description is made more precise in Chapter 6. In Chapter 6, the actual behaviour of two components and a link is related to the intended behaviour as described by the information link mapping. In other words, Chapter 6 discusses the relation between states that *actually occur* in the behaviour of the destination and source components of a link and the link itself, and the relation between *possible states* of these components described by an information link mapping. In Chapter 5, information link mappings are only used in examples of compatibility relations.

Information links provide flexibility in information transmission in three ways. In the first place, it is possible to specify how states of the two components and the link are connected. As the state of components and links is determined by their information contents, expressed in their 'own' terms, information transmission enables translation of the terms used for different components. (In the running example, this facility is used to translate the ontology used by the broker agent to

the ontology used by the user agents, and vice versa.) In the second place, using the link state, the process of information transmission can be modelled in great detail. In the third place, it is possible to specify results of information transmission (in the example: state just_communicated_to(t,user_1)), and enabling conditions in the destination component of a link (in the example: state ready_for_information) and in the link itself.

5.1.3 Compositional Structures

Section 2.1 names the composition structure of a component as its third aspect. This aspect describes compositions of components in terms of the subcomponents and information links that constitute the components. The structural aspect is very general: starting from a set of components *Comp* (or, more precisely, component names) and a set of links *Lnk* (or, link names), arbitrary recursive composition structures, called *structure hierarchies,* can be described. It is not assumed that for a given set of components and a given set of links, there is only one structure hierarchy, nor is there any commitment with respect to whether components are composed or primitive. Thus, different perspectives on a set of components and a set of links can be described. For instance, in a certain stage in the analysis or development of a multi-agent system, certain components may be considered primitive. (They do not have subcomponents). In a later stage, these subcomponents may become composed. The qualifiers 'composed' and 'primitive' are thus related to a given structure hierarchy, as is the notion of being a subcomponent.

While the semantic structure developed in this thesis does not assume that the structure hierarchy of a component is unique, it is assumed that many applications of the semantic structure have a certain unique structure for each component. For instance, a specification framework for multi-agent systems may provide facilities to describe the compositional structure of a component. Such a description may induce a unique structure hierarchy for the component. Even if this is the case, different refinements of specifications most often result in different structure hierarchies for the same component. The formal definition of a structure hierarchy is as follows:

Definition 5.8. (Structure hierarchy). *A* structure hierarchy *SH is a tuple* $\langle Comp;Lnk;\prec;dom;cdom\rangle$*, where:*

- *Comp is a finite set of component names;*
- Lnk *is a finite set of information link names such that* $Comp \cap Lnk = \emptyset$*;*
- $\prec \subseteq (Comp \cup Lnk) \times Comp$ *(the hierarchy relation) such that* $\prec$ *defines a forest: a finite, non-empty collection of trees. For all pairs* $\langle I;C\rangle \in \prec$ *such that* $I \in Lnk$*, I must be a leaf. The reflexive closure of* $\prec$ *is denoted* $\preceq$*;*

- *dom,cdom: Lnk→Comp∪Lnk are total functions (the domain and co-domain functions) such that for all I∈Lnk, if dom(I)=S_1 and cdom(I)=S_2, then either*
  - *S_1,S_2∈Comp and there is a P∈Comp such that S_1,S_2≺P (private link), or*
  - *S_1,S_2∈Comp and S_2≺S_1 (import mediating link), or*
  - *S_1,S_2∈Comp and S_1≺S_2 (export mediating link), or*
  - *S_1,S_2∈Comp and there is a P∈Comp such that I,S_1,S_2≺P and S_1=S_2 (cross-mediating link), or*
  - *S_1∈Comp and S_2∈Lnk and there are S_3,S_4,P∈Comp such that S_3≠S_1, S_4≠S_1, P≠S_i and S_i≺P for i=1,…,4 and dom(S_2)=S_3 and cdom(S_2)=S_4, (link modifier link), or.*
  - *S_1∈Lnk and S_2∈Comp and there are S_3,S_4,P∈Comp such that S_3≠S_2, S_4≠S_2, P≠S_i and S_i≺P for i=1,..,4 and dom(S_1)=S_3 and cdom(S_1)=S_4, (link monitoring link).*

A structure hierarchy is called a structure hierarchy for a component C∈Comp iff ≺ defines a collection of exactly one tree (i.e., ≺ is connected, formally ∀C_1,C_2∈Comp: C_1≺C_2 ∨ C_2≺C_1) and C is the root of the tree defined by ≺, thus ¬∃C′∈Comp: C≺C′. If Comp=∅, then the structure hierarchy is called empty.

In the subcomponent relation, C_1≺C_2 denotes that C_1 is a subcomponent of C_2 in *SH*. A component *C* is called primitive in *SH*=⟨Comp;Lnk;≺;dom;cdom⟩ iff there is no *C′*∈*Comp* such that *C′*≺*C*. Otherwise, it is called composed in *SH*. Components which are leaves in a structure hierarchy (that is, components *C* in the structure hierarchy for which there is no *C′* such that *C′*≺*C)* are by definition primitive in *SH*.

In general, a structure hierarchy for a component *C* not only contains subcomponents of *C*, but also subcomponents of these subcomponents, and so on. Therefore, a structure hierarchy itself comprises more than the compositional structure (the third aspect of a component distinguished by the semantic structure) of a component *C*. A structure hierarchy consisting of *C*, its subcomponents and links is called the *composition structure* of *C*. Formally:

Definition 5.9. (Composition structure).

- *A* composition structure *for a component C is a structure hierarchy CS=⟨Comp;Lnk;≺;dom;cdom⟩ for C such that for all S∈Comp∪Lnk, S≺C.*
- *Let SH=⟨Comp;Lnk;≺;dom;cdom⟩ be a structure hierarchy and let C∈Comp be a component. The* composition structure *CS(C,SH)* of *C* with respect to *SH is the structure hierarchy ⟨Comp′;Lnk′;≺′;dom′;cdom′⟩ where:*
  - *Comp′={ C′∈Comp | C′⪯C };*
  - *Lnk′={ I∈Lnk | I≺C };*
  - *S≺′C ⇔ S≺C, S∈Comp′∪Lnk′ and C∈Comp′;*
  - *For all I∈Lnk′, dom′(I)=dom(I);*
  - *For all I∈Lnk′, cdom′(I)=cdom(I).*

If, for a structure hierarchy SH for C it holds that $SH=CS(C,SH)$, then SH itself is a composition structure for C. For a given component C and structure hierarchy SH, the composition structure $CS(C,SH)=\langle Comp';Lnk';\prec';dom';cdom'\rangle$ is unique. The set of subcomponents of C with respect to $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ is denoted $Subc(C,SH)=\{C' \in Comp \mid C' \prec C\}$. The set of links of C with respect to SH is denoted $Lnk(C,SH)=\{l \in Lnk \mid l \prec C\}$. The set of subcomponents of C with respect to SH and links '*inside*' C is denoted $SLC(C,SH)$: $SLC(C,SH)=\{C\} \cup Subc(C,SH) \cup Lnk(C,SH)$. (The abbreviation 'SLC' stands for '*S*ubc, *L*nk and the *C*omponent itself', but also for 'slice'.) If a component C is a primitive component according to a structure hierarchy SH for C, then, according to the definition, $Subc(C,SH)=\varnothing$.

Example 5.10. To illustrate structure hierarchies, a structure hierarchy for the compositional system depicted in Figure 4.5 is given. For conciseness, the structure hierarchy only contains the components toplevel, user_1, broker, ASP and OPC. The following structure hierarchy can be used to analyse user_1 together with agent broker: $sh=\langle Comp;Lnk;\prec;dom;cdom\rangle$, with:

- $Comp=\{$toplevel,user_1,broker,ASP,OPC$\}$;
- $Lnk=\{$user_1_to_broker,broker_to_user_1$\}$;
- $\prec=\{\langle$ASP;broker$\rangle,\langle$OPC;broker$\rangle,\langle$user_1;toplevel$\rangle,\langle$broker;toplevel$\rangle\}$;
- $dom=\{\langle$user_1_to_broker;user_1$\rangle,\langle$broker_to_user_1;broker$\rangle\}$;
- $cdom=\{\langle$user_1_to_broker;broker$\rangle,\langle$broker_to_user_1;user_1$\rangle\}$. ■

5.2 Dynamics

This section focuses on how the dynamics of a compositional system is described using the constructs provided by the semantic structure. First, Section 5.2.1 discusses the description of the local dynamics of components. Section 5.2.2 discusses the description of the dynamics of composition structures.

5.2.1 Local Dynamics

The information state of a component changes over time, which is modelled using sequences of information states called traces. Separate local traces of the input, internal and output substates are distinguished, making it possible for applications of the semantic structure to focus on one or more of the interfaces of a component.

Definition 5.11. (Time frame). *A* time frame *is a pair $TF=\langle T;<\rangle$, where T is a set of time points and $<$ is a strict partial order on T. Moreover, $<$ is connected, i.e. $\forall t \in T: \exists t' \in T: t<t' \vee t'<t$. There is one element, $\perp \in T$, for which there is no $t \in T$ such that $t<\perp$.*

This definition enables various types of time frames to be used in the semantic structure, such as, for instance, branching time frames and dense time frames. In the remainder of this thesis, linear time frames are assumed, unless explicitly states

otherwise. (For example, the discussion of dense time frames in Chapter 7). It is, however, possible to use different time frames with different properties for different components in a compositional system.

Definition 5.12. (Local trace). *Local traces are defined as follows:*

- *A* local input trace *of a component C for a time frame* $TF=\langle T;<\rangle$ *is a pair* $LT_{C,in}=\langle TF;V\rangle$*, where V is a total function* $V: T\rightarrow\mathcal{S}_{C,in}$*. The* internal and output traces *are defined analogously*.
- *A* local component trace *of a component C for a time frame* $TF=\langle T;<\rangle$ *is a pair* $LT_C=\langle TF;V\rangle$*, where V is a function* $V: T\rightarrow\mathcal{S}_C$.
- *The set of all local input traces of a component C is denoted* $\mathcal{LT}_{C,in}$*. The sets of all internal and output traces of a component are denoted analogously. The set of all local component traces of a component C is denoted* $\mathcal{LT}_C$.

In the previous definitions, four different notions of traces are distinguished (input, internal, output and local component traces). In the semantic structure developed in this thesis, all four traces can be associated with each component. These four traces are not independent: a local component trace consists of local component states, each of which itself consists of the input, internal and output traces of the same component. The dependencies between traces can be of several types. For instance, assume that for the input, internal and output trace of a component a discrete totally ordered time frame is used. In this case, at least three choices for the local component traces can be distinguished:

- For a local component trace, a discrete totally ordered time frame is chosen. Consider a point in time t and its immediate successor t' (which is assumed to exist here). The state at time point t' differs from the state at time point t for *all three substates* input, internal and output;
- For a local component trace, a discrete totally ordered time frame is chosen. Consider a point in time t and its immediate successor t' (which is assumed to exist here). The state at time point t' differs from the state at time point t for *two of the three substates* input, internal and output: the input and output substate differ, while the internal state remains the same;
- For a local component trace, a discrete totally ordered time frame is chosen. Consider a point in time t and its immediate successor t' (which is assumed to exist here). The state at time point t' differs from the state at time point t for *one of the three substates* input, internal and output;

If the restriction to totally ordered time frames is dropped, and instead, a local component trace with a partially ordered time frame is chosen, another alternative is possible. Consider a point in time t and *one* of its immediate successors t' (which is assumed to exist here). The state at time point t' differs from the state at time point t for *one of the three substates* input, internal and output.

These different possibilities for local component traces model different choices with respect to synchronisation of the different (input, internal and output) substates of a component. In an application of the semantic structure, a time frame and a dependence relation between the traces is chosen. (It is possible to vary these choices for different components.) The flexibility with respect to these choices is a feature of the semantic structure.

Local component traces are used to model the behaviour of components. For each component C, a subset of $\mathcal{LT}_C$ is distinguished which contains all local component traces that are possible behaviours of C from a local point of view. (It is assumed that the possible behaviours of C from a local point of view are given, for instance using the specification language presented in Chapter 9.) Formally:

Definition 5.13. (Local component behaviour). *Let C be a component. A* local component behaviour *$Beh_{loc}(C)$ of C is a set of local component traces of C.*

To clarify the use of the qualification 'local' in the name of the notion defined above, a distinction is made between two views on the concept of location. On the one hand, a component has a specific location in a structure hierarchy. This concept of location refers to the structure of a compositional system in terms of components and subcomponents. This structure is determined by how processes relate to one another in terms of the function of the subprocesses, as explained in Chapter 3. In this conception of the location of a component, a subcomponent is 'close', or 'local' to its parent component. On the other hand, there is also a 'physical' distribution of components (and the processes they represent) over processes, which are spatially divided. It is *not* assumed that the subcomponents of a component all exist at the same physical location. Thus, subcomponents of a component are not necessarily local to their parent component with respect to the 'physical' location of a compositional system. In this thesis, the qualifications 'local' and 'global' always refer to the physical distribution of a compositional system.

The notion of local component behaviour is called 'local' because the set $Beh_{loc}(C)$ is not constrained by non-local phenomena such as information transmission (not even with its subcomponents). A set $Beh_{loc}(C)$ is independent of any structure hierarchy in which C occurs. In other words, the set $Beh_{loc}(C)$ can contain local component traces that are not possible behaviour when information exchange is taken into account, for instance because these local component traces depend on information residing in other components. This is made more precise in Section 5.2.2.

Example 5.14. In the running example, the set of natural numbers together with the usual ordering is used as a time frame. A local component trace is represented as a sequence of component states with the sets of input, internal and output propositions separated by bars. To present some example traces, concrete elements of $\mathcal{S}_{\text{broker}}$ have to be given. (In a previous example, elements of $\mathcal{S}_{\text{broker}}$ were specified in an abstract way, because the sets of ontology terms OT_1 and OT_2 were not

specified.) In this example, the sets OT_1, OT_2 and Q are partially specified by a few sample elements. First, the set of ontology terms OT_2 used by the information providers and the broker agent contains elements res_1 and res_2, which stand for specific resource descriptions. Thus, $\{$res_1,res_2$\} \subseteq OT_2$. Second, the set OT_1 of ontology terms used by the broker agent to transmit resource descriptions to user agents contains elements res_3 and location_of_holy_grail. Thus, $\{$res_3,location_of_holy_grail$\} \subseteq OT_1$. Third, the set of query terms used by user agents contains query terms query_1 and where_is_holy_grail. Formally: $\{$query_1,where_is_holy_grail$\} \subset Q$. The query about the location of the Holy Grail and the ontology term describing a resource that reveals the location of the Holy Grail serve to illustrate incorrect or formally correct, but impossible traces.

First, two example traces for broker are presented. Both traces only contain states of $\mathcal{S}_{\text{broker}}$, so both traces are elements of $\mathcal{LT}_{\text{broker}}$. Local traces of broker, or, in other words, elements of Beh_{loc}(broker), should satisfy the requirement mentioned in Section 4.2. The first trace presented below satisfies the requirement presented in Section 4.2 and is therefore an element of Beh_{loc}(broker). The second trace does not satisfy the requirement, as, after receipt of a query for which it knows a matching resource, it does not transmit information on this resource to user_1. Instead, it transmits information on another resource, res_2. Therefore, this trace is not an element of Beh_{loc}(broker).

$lt_{\text{broker},1}$ = ∅ | belief(match(res_1,query_1)) | ∅ →
communicated_by(query_1,user_1) | belief(match(res_1,query_1)) | ∅ →
communicated_by(res_1,provider_1) | belief(match(res_1,query_1)) | ∅ →
∅ | belief(match(res_1,query_1)) | to_be_communicated_to(res_1,user_1) →
∅ | belief(match(res_1,query_1)) | just_communicated_to(res_1,user_1)

$lt_{\text{broker},2}$ = ∅ | belief(match(res_1,query_1)) | ∅ →
communicated_by(query_1,user_1) | belief(match(res_1,query_1)) | ∅ →
∅ | belief(match(res_1,query_1)) | to_be_communicated_to(res_2,user_1) →
∅ | belief(match(res_1,query_1)) | just_communicated_to(res_2,user_1)

Second, two traces for user_1 are presented. Both traces only contain states of $\mathcal{S}_{\text{user\_1}}$, so both traces are elements of $\mathcal{LT}_{\text{user\_1}}$. Moreover, both traces are elements of Beh_{loc}(user_1), because both comply with the intended functionality of a user agent in the running example. However, the second trace cannot be an actual trace for behaviour of user_1, because in the example, there is no information provider that provides information on the location of the Holy Grail.

$lt_{\text{user\_1},1}$ = ∅ | ∅ | to_be_communicated_to(query_1,broker) →
ready_for_information | ∅ | ∅ →
communicated_by(res_3,broker) | ∅ | ∅

$lt_{\text{user\_1},2}$ = ∅ | ∅ | to_be_communicated_to(query_where_is_holy_grail,broker) →
ready_for_information | ∅ | ∅ →
communicated_by(location_of_holy_grail,broker) | ∅ | ∅ ■

These example traces are used in several examples below. The information state of a link also changes over time, and this is likewise modelled by traces of link states as follows:

Definition 5.15. (Information link trace). *An* information link trace *of an information link I for a time frame TF=⟨T;<⟩ is a pair LT_I=⟨TF;V⟩, where V is a total function V: T→S_I. The set of all link traces of a link from D to C is denoted $\mathcal{LT}_I$.*

The way in which the behaviour of a link is described by a set of information link traces, as witnessed by the following definition, is similar to the way in which local component behaviour is defined by a set of local component traces:

Definition 5.16. (Local link behaviour). *Let I be an information link. The* local link behaviour *of I is a set $Beh_{loc}(I)$ of information link traces.*

Example 5.17. A possible information link trace for the link broker_to_user_1 is:

$lt_{\text{broker\_to\_user\_1}}$ = awake_and_empty → active_and_contents(res_1) → awake_and_empty →

In this trace, the link is first awake (ready to transmit information) and there are no messages in transit. At the second time point, the link is busy (actively transmitting information, and a message t is in transit. At the third point in time, the message is delivered. The link is empty again and ready to transmit information. ■

5.2.2 Compositional Dynamics

In Section 5.2.1, local component traces are defined as a means to model the behaviour of a single component. In Section 5.1.3, the notion of a structure hierarchy is introduced, which enables the description of compositions consisting of components and links. This section defines structures for describing the behaviour of such compositions, in terms of the local component and link traces defined in Section 5.2.1.

Local component and link traces, the elements of the set $Beh_{loc}(S)$ given for *S*, as defined in Section 5.2.1, are the basic notions for describing the behaviour of a single component or link *S*. A local component trace of a composed component *C* is, by itself, not not sufficient to describe the behaviour of the subcomponents and links of *C*, because local component traces only record local information. However, the behaviour of a composed component *C* and its subcomponents and links can be described by a *structure* consisting of *a number of* local component and link traces: a local component trace of *C*, a local component trace of each of its subcomponents and a local link trace of each link in *C*. Indeed, so-called *multitraces*, which are defined in Section 5.2.2.1 to describe the behaviour of (composed) components, are structures of local component and link traces.

Multitraces cannot be *arbitrary* combinations of elements of the sets $Beh_{loc}(S)$, where $S \in \{C\} \cup Subc(C)$. The set $Beh_{loc}(C)$ as defined in Section 5.2.1 is intended, as stated, to describe the *local* behaviour of *C*. In other words, the behaviour of *C* as described by the set $Beh_{loc}(C)$ is independent of the composition structure in which

C is a part and the information exchange with other components in such a structure. However, to describe the behaviour of a composed component, dependencies imposed by information exchange between subcomponents, and between subcomponents and the composed component, have to be taken into account. For example, assume that no information provider provides information on a resource describing the location of the holy grail. If the behaviour of the running example compositional system were to be described by arbitrary combinations of local component and link traces, the example trace in which user_1 receives information on a resource describing the location of the holy grail (given in Example 5.14) would be included in at least some of the combinations of local component traces. The intended meaning of this fact is that, at least in some actual behaviours of the example compositional system, component user_1 receives information on a resource describing the location of the holy grail. However, such behaviour is impossible because user_1 can only receive such information if one of the information providers can provide such information. By assumption, this is not possible in the running example.

5.2.2.1 Compatibility and Multitraces

Dependencies between components imposed by information transmission are represented by compatibility relations. The principle underlying compatibility relations is as follows. For each link, a compatibility relation for that link describes which local component or link traces of the domain of the information link are compatible with which local component or link traces of the co-domain of the link. A local component or link trace of the domain is compatible with a local component or link trace of the co-domain if the information transmission as defined by the information link mapping, and specific general properties of information transmission, are taken into account (this is defined formally and in detail in Section 6.1). The behaviour of a component not only depends on the behaviour of other components with which it exchanges information, but also on the state of the links used for this information exchange (commitment presented in Section 2.2.3.2). For instance, in the running example, if the information link from broker to user_1 is continuously disabled, then the traces given for broker and user_1 cannot be considered to be compatible. Therefore, a compatibility relation for an information link *I* is defined as a *ternary* relation on the set of local component or link traces of the domain of *I*, the set of link traces of *I*, and the set of local component or link traces of the co-domain of *I*.

The focus of this section is on how compatibility relations can be employed to determine which combinations of local component traces of a component, its subcomponents and its links, constitute its actual behaviour. As a consequence, although compatibility relations are defined in this section, the discussion of various properties of compatibility relations is postponed until Chapter 6.

Definition 5.18. (Compatibility relation). *A* compatibility relation *for a link I is a relation* $c\mathcal{R}_I \subseteq \mathcal{LT}_{dom(I)} \times \mathcal{LT}_I \times \mathcal{LT}_{cdom(I)}$.

Example 5.19. Compatibility relations vary widely in their appearance due to the flexibility offered by the semantic structure. For instance, properties of the local component or link traces of the components and links related by a compatibility relation influence how a concrete compatibility relation in an application of the semantic structure is defined. In this example, a compatibility relation for a link *I* is defined, based on the following assumptions:

- Local component and link traces of *I*, its domain and co-domain are discrete, totally ordered and infinite (thus, the traces are isomorphic to the natural numbers);
- The link *I* is a private link;
- For link *I*, the facilities offered by the semantic structure for a detailed representation of information transmission as a process and enabling conditions on the state of *I* itself are not used.

Moreover, the compatibility relation defined in this example has the input persistence and lossless transmission properties (which are introduced in Chapter 6). In this example, the following notation is used. Let $LT_S = \langle\langle T, <\rangle; V\rangle \in \mathcal{LT}_S$ be a local component or link trace. The state $V(t)$ in LT_S at time point $t \in T$ is denoted $v_{S,t}$. The compatibility relation, $c\mathcal{R}_I$, is defined as follows: $c\mathcal{R}_I$ is the set of tuples $\langle LT_{dom(I)}; LT_I; LT_{cdom(I)}\rangle$ such that:

- $LT_{dom(I)} = \langle\langle T_{dom(I)}; <_{dom(I)}\rangle; V_{dom(I)}\rangle \in \mathcal{LT}_{dom(I)}$;
- $LT_I = \langle\langle T_I; <_I\rangle; V_I\rangle \in \mathcal{LT}_I$;
- $LT_{cdom(I)} = \langle\langle T_{cdom(I)}; <_{cdom(I)}\rangle; V_{cdom(I)}\rangle \in \mathcal{LT}_{cdom(I)}$;
- For all $i \in T_{dom(I)}$, either:
  - There are $j \in T_{dom(I)}$, $i'' <_I j'' <_I k <_I l \in T_I$ and $i' <_{cdom(I)} j' \in T_{cdom(I)}$ such that $\langle\langle out(v_{dom(I),i}); out(v_{dom(I),j})\rangle; \langle v_{I,i''}; v_{I,j''}; v_{I,k}; v_{I,l}\rangle; \langle in(v_{cdom(I),i'}); in(v_{cdom(I),j'})\rangle\rangle \in \lambda_I$,
  - Or there is no $\langle\langle out(v_{dom(I),i}); \sigma_2\rangle; \langle \sigma_3; \sigma_4; \sigma_5; \sigma_6\rangle; \langle \sigma_7; \sigma_8\rangle\rangle \in \lambda_I$ for any $\sigma_2, \ldots, \sigma_8$.

This definition shows the role of information link mappings: only those traces in which states occur in specific sequences as specified by the information link mapping, are compatible. ■

Example 5.20. In the example, a possible compatibility relation for the link broker_to_user_1 is given. A compatibility relation for this link consists of triples of local component and link traces of broker_to_user_1, *dom*(broker_to_user_1)=broker and *cdom*(broker_to_user_1)=user_1. Traces for broker and user_1 are given in Example 5.14. A trace for the link broker_to_user_1 is given in Example 5.17. A compatibility relation for broker_to_user_1 relates traces of broker, user_1 and broker_to_user_1 according to the information link mapping of broker_to_user_1 given in Example 5.7,

together with general properties of information transmission presented in Chapter 6. (In this example, such properties are briefly sketched where necessary.) The triple $\langle lt_{broker,1};lt_{broker\_to\_user\_1};lt_{user\_1,1}\rangle$ is an element of a compatibility relation for broker_to_user_1, for the following reasons. First, the information link mapping presented in Example 5.7 states that if a state to_be_communicated_to(t,user_1) occurs, for t an ontology term, and the state of link broker_to_user_1 is awake_and_empty, then user_1 should reach the state communicated_by(t',broker), for $t'=trans(t)$ another ontology term that is the translation of t. Moreover, before user_1 can reach this state, it must have reached the state ready_for_information, and after state to_be_communicated_to(t,user_1) occurs in broker, the state just_communicated_to(t,user_1) should have occurred. Furthermore, the link broker_to_user_1 should reach the state active_and_contents(t) and after that, awake_and_empty. A compatibility relation for broker_to_user_1 relates local component and link traces in which the situation called for by the information link mapping actually occurs. The traces in the triple $\langle lt_{broker,1};lt_{broker\_to\_user\_1};lt_{user\_1,1}\rangle$ are traces that agree with the requirement represented by the information link mapping. At the end of trace $lt_{broker\_1,1}$, the output substate first is to_be_communicated_to(t,user_1) and after that, just_communicated_to(t,user_1) as required. At the end of trace $lt_{user\_1,1}$, first the state ready_for_information occurs, followed by the state communicated_by(t',broker). Trace $lt_{broker\_to\_user\_1}$ also fulfils the requirement: its first state is awake_and_empty, then the state active_and_contents(term_1) occurs, followed by the state awake_and_empty. Second, the traces in the triple $\langle lt_{broker,1};lt_{broker\_to\_user\_1};lt_{user\_1,1}\rangle$ fulfil general properties of information transmission. In fact, the three traces fulfil the following properties: information transmitted by broker arrives at user_1, and it is not garbled. The three traces also comply with the most basic property of information transmission (information does not arrive before it is sent). However, the example is too simple to prove or disprove this claim. ■

As stated in Section 4.1.1.2, in fact three views on the behaviour of a compositional system are defined: the glass box view, the white box view and the black box view. In all three views, the behaviour of a component is defined in terms of *multitraces*. A multitrace is a collection of local component traces and link traces of a set of components and links, indexed by a structure hierarchy as defined in Section 5.1.3. With compatibility relations imposing additional structure and subject to certain other requirements, these collections of local component traces and link traces model behaviour as defined by the three views. An indexed set can also be seen as a function, which is the view taken in the formal definition below:

Definition 5.21. (Multitrace). *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy. A* multitrace *$(mt_S)_{S\in Comp\cup Lnk}$ for SH is a total function mt: $Comp\cup Lnk \rightarrow \bigcup_{S\in Comp\cup Lnk} \mathcal{LT}_S$ such that for all $S\in Comp\cup Lnk$, $mt(S)\in\mathcal{LT}_S$. The set of all multitraces for SH is denoted MT_{SH}. A typical element of MT_{SH} is denoted μ. The element of a multitrace μ with index P is denoted μ_P (or sometimes $\mu(P)$ if this is more convenient).*

The hierarchy relation $\prec$ on the index set of a multitrace induces a hierarchical structure on the collection of local component and link traces that is indexed by this set, as depicted in Figure 5.1, which refers to the running example. In this figure, the left hand sides depicts the hierarchy relation of the structure hierarchy *sh* given in Example 5.10. The dashed lines show the multitrace as a mapping that maps to each component a local component trace depicted at the right hand side.

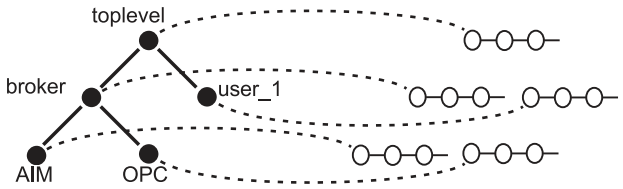

Figure 5.1: Hierarchical structure of a multitrace.

In the semantic structure developed in this thesis, multitraces are used to model possible behaviours of composed components. To model a possible behaviour, a multitrace, which is a collection of local component traces without additional structure apart from the indexing, has to take constraints imposed by non-local phenomena into account. Such constraints are represented by compatibility relations. Therefore, in the following definition, *compatible multitraces* are defined as multitraces in which the local component traces and link traces in the multitrace are related by compatibility relations.

Definition 5.22. (Compatible multitrace). *Let $SH=(Comp;Lnk;\prec;dom;cdom)$ be a structure hierarchy and let $\gamma=(\gamma_I)_{I\in Lnk}$ be a collection of compatibility relations. A multitrace μ for SH is* compatible for γ *iff the following property holds:*

$$\forall I\in Lnk: \langle \mu_{dom(I)};\mu_I;\mu_{cdom(I)}\rangle\in\gamma_I.$$

This definition shows why multitraces are indexed by a set that not only includes components, but also links: compatibility requires that the middle element of a triple in the compatibility relation is present in the multitrace (denoted above by μ_I).

If a component *C* is primitive, according to a structure hierarchy *SH* for *C*, and there are no links from *C* to itself, (thus $SH=(\{C\};\emptyset;\emptyset;\emptyset;\emptyset)$), then every multitrace for *SH* is compatible.

Before the definitions of the three views on behaviour can be given, two additional notions derived from the notion of a structure hierarchy are defined. In some definitions, only a subtree of (one of the trees occurring in) a structure hierarchy is used. Such a subtree is unique for a given structure hierarchy and is defined as follows:

Definition 5.23. (Substructure). *Let $SH=(Comp;Lnk;\prec;dom;cdom)$ be a structure hierarchy and let $S\in Comp\cup Lnk$ be a component or link. If $S\in Comp$, then the*

substructure *SS(S,SH) of SH for S is defined as the structure hierarchy* $\langle Comp';Lnk';\prec';dom';cdom'\rangle$ *where:*

- $Comp'=\{ C' \in Comp \mid C' \prec^* S \} \cup \{S\}$, *where is* $\prec^*$ *the transitive closure of* $\prec$;
- $Lnk'=\{ I \in Lnk \mid dom(I),cdom(I) \in Comp' \}$;
- $S \prec' C \Leftrightarrow S \prec C$, $S \in Comp' \cup Lnk'$ *and* $C \in Comp'$;
- *For all* $I \in Lnk'$, $dom'(I)=dom(I)$;
- *For all* $I \in Lnk'$, $cdom'(I)=cdom(I)$.

If $S \in Lnk$, *then* $SS(S,SH)=\langle \emptyset;\{S\};\emptyset;\emptyset;\emptyset\rangle$.

It is obvious that for arbitrary $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ and C in $Comp$, $SS(C,SH)$ is a structure hierarchy for C (that is, it is a forest of exactly one tree with C as root). The notion of a substructure of SH for S with S a link is used in other definitions where a variable S' ranges over all components and links in $SLC(C,SH)$, such as the following definition:

Definition 5.24. (Primitive components). *Let C be a component and let SH be a structure hierarchy for C. The set of* primitive components *in SH is defined as follows:*

$Prim(SH) = \bigcup_{C' \in Subc(C,SH)} Prim(SS(C',SH))$, *if* $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ *such that* $Comp \supset^{6} \{C\}$.

$Prim(SH) = \{C\}$, *if* $SH=\langle \{C\};Lnk;\emptyset;dom;cdom\rangle$.

This inductive definition of the set of primitive components in a structure hierarchy is referenced in several definitions and propositions below. Moreover, it serves as a means for proofs by induction of these propositions. The stage is now set to define the three views on the behaviour of a component.

5.2.2.2 The White Box View

The first of the three views presented is the white box view. The three views on the behaviour of a component are defined relative to a structure hierarchy and the behaviour of specific other components in the structure hierarchy. Thus, the three views can be seen as composition operators that define behaviour of a composition of components in terms of the behaviour of the constituents of the composition. Consequentially, if the behaviour of the components in the structure hierarchy to which each view is relative, is not correct, then the behaviour defined by each of the views is also incorrect. The behaviour of a component S of C to which a view on the behaviour of C is relative need not be $Beh_{loc}(S)$. However, it is required that the behaviour of such a set S is a subset of $Beh_{loc}(S)$.

The white box view on the behaviour of a component differs from the other two views with respect to the kind of structure hierarchy considered. On the one hand,

[6] In this thesis, $\subset$ and $\supset$ denote *proper* subsets.

the black box and glass box views are both relative to an arbitrary structure hierarchy. The definitions of these two views refer to compatible multitraces for the relevant structure hierarchy. Consequently, the black box and glass box views may take the behaviour of arbitrary components into account by choosing an appropriate structure hierarchy. On the other hand, the white box view is relative to a composition structure *CS*, which is not an arbitrary structure hierarchy. As a consequence, the definition of the white box view on the behaviour of a composed component *C* can only refer to the behaviour of the subcomponents of *C*. (As stated in Chapter 2, in this thesis the term 'subcomponent' always refers to *direct* subcomponent of a component *C*.)

In addition to a composition structure *CS*, the white box view is relative to a collection of compatibility relations γ and to the behaviour of the subcomponents and links (the elements of $SLC(C,CS)\backslash\{C\}$). The white box view on the behaviour of a component *C* consists of a set of structures (multitraces) each of which consists of local component traces of *C*, its subcomponents and links. Because of nondeterminism, which gives rise to different alternative behaviours, a component can have more than one multitrace. (Each multitrace contains one behaviour alternative of a specific component.) Therefore, the white box view consists of a set of multitraces.

Definition 5.25. (Component behaviour, white box view). *Let C be a component, let* $CS=\langle Comp;Lnk;\prec;dom;cdom\rangle$ *be a non-empty composition structure for C, let* $\gamma=(\gamma_l)_{l\in Lnk}$ *be a collection of compatibility relations and let* $\Gamma=(\Gamma_S)_{S\in SLC(C,CS)}$ *be a collection of sets of traces such that for all* $S\in SLC(C,CS)$, $\Gamma_S\subseteq Beh_{loc}(S)$. *The* white box view *on the behaviour of C,* $Beh_{WB}(C,CS,\gamma,\Gamma)$, *with respect to CS,* γ *and* Γ *is the set of compatible multitraces* $\mu\in MT_{CS}$ *of C such that for each subcomponent or link S of C it holds that the local component trace of S in* μ *is an element of* Γ_S. *Formally:*

$$Beh_{WB}(C,CS,\gamma,\Gamma) = \{\, \mu \mid \mu\in MT_{CS} \text{ is compatible for } \gamma \text{ and } \forall S\in SLC(C,CS): \mu_S\in\Gamma_S \,\}.$$

This definition of component behaviour provides a white box view on the behaviour of a component in the sense that each multitrace in $Beh_{WB}(C,CS,\gamma,\Gamma)$ not only contains a local component trace of *C*, but also local component and link traces of the subcomponents and links of *C*. However, these local component and link traces themselves do not contain information of their subcomponents and so on, recursively. Nevertheless, as is the case with the black box view on the behaviour of *C*, the requirement on the local component traces that constitute $Beh_{WB}(C,CS,\gamma,\Gamma)$ ensures that the combinations of local component traces and link traces that constitute the elements of $Beh_{WB}(C,CS,\gamma,\Gamma)$ take constraints imposed by information transmission into account.

If a component *C* is primitive according to a composition structure *CS* for *C*, (thus $CS=\langle\{C\};\emptyset;\emptyset;\emptyset;\emptyset\rangle$ and $Subc(C,CS)=Lnk(C,CS)=\emptyset$), then for every collection of

compatibility relations γ, it holds that $Beh_{WB}(C,CS,\gamma,\Gamma)=\{ \mu \in MT_{CS} \mid \mu_C \in Beh_{loc}(C) \}$ (with $\Gamma=\varnothing$).

5.2.2.3 The Black Box View

The second of the three views presented is the black box view. Similar to the other two views, the black box view is defined relative to a structure hierarchy, a collection of compatibility relations and the behaviour of specific other components in the structure hierarchy. In contrast to the white box view, but similar to the glass box view, the black box view is relative to an arbitrary structure hierarchy. (The white box view is relative to a specific type of structure hierarchy: a composition structure.)

In addition to an arbitrary structure hierarchy SH, the black box view on the behaviour of a component C is defined relative to a collection of compatibility relations γ and the behaviour of its subcomponents and links (the elements of $SLC(C,SH)\setminus\{C\}$).

Definition 5.26. (Component behaviour, black box view). *Let C be a component, let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a non-empty structure hierarchy for C, let $\gamma=(\gamma_I)_{I\in Lnk}$ be a collection of compatibility relations and let $\Gamma=(\Gamma_S)_{S\in SLC(C,SH)}$ be a collection of sets of traces such that for all $S\in SLC(C,SH)$, $\Gamma_S\subseteq Beh_{loc}(S)$. The* black box view *$Beh_{BB}(C,SH,\gamma,\Gamma)$ for C with respect to SH, γ and Γ_S on the behaviour of C is the subset of $Beh_{loc}(C)$ such that each local component trace in this subset is part of a compatible multitrace for SH that is based on the given traces of the subcomponents and links of C. Formally:*

$$\begin{aligned} Beh_{BB}(C,SH,\gamma,\Gamma) = \{\ & \mu_C \mid \mu\in MT_{SH} \text{ is compatible for } \gamma, \\ & \forall S\in SLC(C,SH): \mu_S\in\Gamma_S \text{ and} \\ & \forall I\in Lnk \text{ such that } dom(I)=cdom(I)=C: \mu_I\in Beh_{loc}(I)\ \}. \end{aligned}$$

The definition given above provides a black box view in the sense that the behaviour of a component C is a set consisting of local component traces of C itself only, and not of its subcomponents and links. As the definition of local component traces given in Section 5.2.1 indicates, only local information is recorded in a local component trace of a component. In particular, information of subcomponents is *not* recorded in the local component trace of the encompassing component and is thus not visible in the black box view on component behaviour as defined above.

As stated in the beginning of Section 5.2.2, the behaviour of a composed component is, in general, not a set of *arbitrary* combinations of local component traces of its subcomponents and link traces, because only combinations that take constraints imposed by information transmission into account can be considered to represent behaviour of the component. The requirement on the local component traces that constitute $Beh_{BB}(C,SH,\gamma,\Gamma)$ ensures that the combinations of local component traces from which the elements of $Beh_{BB}(C,SH,\gamma,\Gamma)$ are taken, take constraints imposed by information transmission into account. This is ensured

because $Beh_{BB}(C,SH,\gamma,\Gamma)$ depends on the existence of a compatible multitrace, and the elements of this compatible multitrace are themselves part of the behaviour of the subcomponents, to which the definition of the black box view is relative.

To determine the black box view on the behaviour of a component C, only the behaviour of the subcomponents and links of C needs to be given. This also holds if, according to the related structure hierarchy, these subcomponents are composed. It suffices to only take the behaviour of the subcomponents into account, (and not of their subcomponents), because it is assumed that the behaviour given for these subcomponents takes constraints imposed by their subcomponents into account. However, if this is not the case, the behaviour as defined by the black box view is incorrect.

If a component C is primitive, according to a structure hierarchy SH for C, and there are no links from C to itself, (thus $SH=\langle\{C\};\emptyset;\emptyset;\emptyset;\emptyset\rangle$, and $Subc(C,SH)=Lnk(C,SH)=\emptyset$), then for every collection of compatibility relations γ, it holds that $Beh_{BB}(C,SH,\gamma,\Gamma)=Beh_{loc}(C)$ (with $\Gamma=\emptyset$). However, $Beh_{BB}(C,SH,\gamma,\Gamma)$ is *not* equal to $Beh_{WB}(C,SH,\gamma,\Gamma)$ because $Beh_{WB}(C,SH,\gamma,\Gamma)$ is a set of multitraces, while $Beh_{BB}(C,SH,\gamma,\Gamma)$ is a set of local component traces.

5.2.2.4 The Glass Box View

As is the case for the black box view, the glass box view is relative to a structure hierarchy SH, a collection of compatibility relations γ and the behaviour specific subcomponents (in this case, the primitive subcomponents). The glass box view on the behaviour of a component C is defined by imposing three requirements on the set of multitraces μ for a structure hierarchy $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ for C: (i) they must be compatible, (ii) for all components C' in $Comp$ that are primitive in SH, the local component trace $\mu_{C'}$ must be an element of the given sets of traces and (iii) for all components and links I in Lnk, the link trace μ_I must be an element of $Beh_{loc}(I)$. Formally:

Definition 5.27. (Component behaviour, glass box view). *Let C be a component, let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a non-empty structure hierarchy for C, let $\gamma=(\gamma_I)_{I\in Lnk}$ be a collection of compatibility relations and let $\Gamma=(\Gamma_S)_{S\in Prim(SH)}$ be a collection of sets of traces for the primitive components in SH. The* glass box view *$Beh_{GB}(C,SH,\gamma,\Gamma)$ for C with respect to SH, γ and Γ on the behaviour of C is the subset of the set MT_{SH} of multitraces for SH such that for all $\mu\in Beh_{GB}(C,SH,\gamma,\Gamma)$ it holds that:*

- *μ is compatible for γ;*
- *$\forall C'\in Prim(SH)\setminus\{C\}$: $\mu_{C'}\in\Gamma_{C'}$ and*
- *$\forall S\in Comp\cup Lnk$: $\mu_S\in Beh_{loc}(S)$.*

The glass box view is the most complete view on the behaviour of a component, as it consists of multitraces for a structure hierarchy SH. At first sight, the white box

view may look like a special case of the glass box view. This is indeed *almost* the case for a structure hierarchy $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ for a component C if this structure hierarchy consists of only two levels. In this case $SH=SS(C,SH)=CS(C,SH)$ and $Prim(SH)\cup\{C\}=SLC(C,SH)=Comp\cup Lnk$, so the three requirements that a multitrace μ must meet to be an element of $Beh_{GB}(C,SH,\gamma,(\Gamma_S)_{S\in Prim(SH)})$ can be written as:

- μ is compatible for γ,
- $\forall C'\in SLC(C,SH)\setminus\{C\}$: $\mu_{C'}\in\Gamma_{C'}$ and
- $\forall S\in SLC(C,SH)$: $\mu_S\in Beh_{loc}(S)$.

The only difference, in this case, between the white box and the glass box view is that the white box view requires that $\mu_C\in\Gamma_S$, while the glass box views requires that $\mu_C\in Beh_{loc}(S)$. An alternative definition of the white box view can thus be given:

Definition (Component behaviour, white box view, alternative definition I). *Let C be a component, let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy for C such that $SH=SS(C,SH)$, let $CS=SS(C,SH)$, let $\gamma=(\gamma_I)_{I\in Lnk}$ be a collection of compatibility relations and let $\Gamma=(\Gamma_S)_{S\in SLC(C,SH)}$ be a collection of sets of traces such that for all $S\in SLC(C,SH)$, $\Gamma_S\subseteq Beh_{loc}(S)$. The* white box view *$Beh_{WB}(C,SH,\gamma,\Gamma)$ for C with respect to CS, γ and Γ on the behaviour of C is the set of compatible multitraces in $Beh_{GB}(C,SH,\gamma,(\Delta_S)_{S\in Prim(SH)})$ (with $\Delta_S=\Gamma_S$ for all $S\in Prim(SH)$) such that $\mu_C\in\Gamma_S$. Formally:*

$$Beh_{WB}(C,SH,\gamma,(\Gamma_S)_{S\in SLC(C,SH)}) = \{\mu\in Beh_{GB}(C,SH,\gamma,(\Delta_S)_{S\in Prim(SH)}) \mid \mu_C\in\Gamma_S\}.$$

In this case, for structure hierarchies with more than two levels, the white box view is not defined. (Note that Definition 5.25 of the white box view is defined for composition structures, which are two-level structure hierarchies, so the alternative definition and Definition 5.25 are similar in this respect.)

A completely different definition of the white box view is to define the white box view for arbitrary structure hierarchies, but only focus on the two highest levels:

Definition (Component behaviour, white box view, alternative definition II). *Let C be a component, let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy for C, let Lnk' be the set of links in $SLC(C,SH)$, let $\gamma=(\gamma_I)_{I\in Lnk'}$ be a collection of compatibility relations for the links in $SLC(C,SH)$ and let $\Gamma=(\Gamma_S)_{S\in SLC(C,SH)}$ be a collection of sets of traces such that for all $S\in SLC(C,SH)$, $\Gamma_S\subseteq Beh_{loc}(S)$. The* white box view *$Beh_{WB}(C,SH,\gamma,\Gamma)$ for C with respect to CS, γ and Γ on the behaviour of C is the set of multitraces $\mu\in MT_{CS(C,SH)}$ of C such that:*

- $\forall I\in Lnk'$: $\langle\mu_{dom(I)};\mu_I;\mu_{cdom(I)}\rangle\in\gamma_I$, *and*
- $\forall S\in SLC(C,SH)$: $\mu_S\in\Gamma_S$.

Note that in this second alternative definition, it is *not* required that μ is a compatible multitrace, because the definition of compatible multitraces requires that $\langle \mu_{dom(I)}; \mu_I; \mu_{cdom(I)} \rangle \in \gamma_I$ for all $I \in Lnk$, not only for all I in the subset Lnk' of Lnk.

Of the three definitions of the white box view on the behaviour of a component C, Definition 5.25 is chosen because this definition puts most emphasis on the compositional nature of the semantic structure. A composition structure CS describes the structure of a component as an entity independent of other components with which it is related. The white box view relative to a composition structure CS describes the behaviour of such an independent component. The other two views on the behaviour of a component can then be employed to describe the behaviour of the component in a larger context. In the rest of this section, a proposition is presented that enables the definition of the behaviour of such a larger context as a composition of the white box views on the behaviour of its constituents.

In fact, in the rest of this section, three propositions are presented that relate the three views on the behaviour of a component. These propositions show how the white box view and the black box view can be expressed in terms of the glass box view by *restricting* the multitraces (in the usual sense of restricting the domain of a function) in the glass box view to a subset of the components and links in a structure hierarchy of C. A restriction of a multitrace to a set S of components and links is denoted $\mu_{|S}$ and is called a *restricted multitrace*. The set of all multitraces restricted by a set S is denoted $MT_{|S}$.

The first proposition shows how the white box view and the glass box view are related. On the one hand, the proposition shows how the white box view can be expressed in terms of the glass box view by restricting multitraces. On the other hand, the proposition shows how the glass box view on the behaviour of a composed component can be constructed from the white box views on the behaviour of this component, its subcomponents and their subcomponents, and so on. The proposition assumes that a structure hierarchy SH for a component C is given and states that, if a multitrace for this structure hierarchy satisfies a specific requirement for the restriction of this multitrace to the subcomponents of C, and their subcomponents, and so on, then this multitrace is an element of the glass box view on the behaviour of C. The definition of the proposition involves the following technical issues:

- Figure 5.2 shows a structure hierarchy for a composed component C consisting of six composed components (grey ovals), thirteen primitive components (white ovals) and four links (small white boxes). For each composed component C', the composition structure $CS(C',SH)$ is enclosed in a solid line. Within component S_1, as an example also $SLC(S_1,SH)\backslash\{S_1\}$ is depicted by a dashed line. The '=' between $CS(S_1,SH)$ and $SLC(S_1,SS(S_1,SH))$ is put between quotes because $CS(S_1,SH)$ is a *structure hierarchy* and $SLC(S_1,SS(S_1,SH))$ is a *set*, and therefore, they cannot be directly compared.

The proposition below expresses the white box view in terms of the glass box view on the behaviour of C' by restricting multitraces for SH to $SLC(C',CS(C',SH))$. To construct the glass box view from the white box views on the behaviour of each C', the proposition assumes that for each component C' in SH (composed and primitive), the restriction of a multitrace for SH to $SLC(C',SS(C',SH))$ is an element of the white box view on the behaviour of C'.

- The glass box view constructed from the white box views is relative to a collection of sets of local component and link traces of the primitive components in SH, as indicated by the definition of the glass box view on the behaviour of a composed component. Such a collection of sets $(\Delta_S)_{S \in Prim(SH)}$ is assumed to be given.

- Each of the white box views from which the glass box view is constructed, is itself relative to a collection of sets of local component and link traces, as indicated by the definition of the white box view on the behaviour of a component. These collections are taken from multitraces for SH as follows. Let μ be a multitrace for SH. For each component C' in SH, a collection of sets of local component and link traces $(\Gamma_S)_{S \in SLC(C',SS(C',SH))}$ is defined as follows: $\Gamma_S=\Delta_S$ if S is a primitive component in SH, or $\Gamma_S=\{\mu_S\}$ otherwise. The collections of sets of traces for each composed component in SH are themselves grouped as a collection of collections $\Gamma\Gamma$ indexed by the components in SH. (Figure 5.2 might help to obtain an overview of the index sets involved.)

- The collection of collections $\Gamma\Gamma$ is well-defined, although for each *composed* component S in SH that is a subcomponent of a component C', $(\Gamma\Gamma_{C'})_S$ is defined twice: once because $S \in SLC(C',SS(C',SH))$ and once because $S \in SLC(S,SS(S,SH))$. In both cases, $(\Gamma\Gamma_{C'})_S=\{\mu_S\}$ because S is composed.

- Each of the white box views from which the glass box views is constructed, is itself also relative to a collection of compatibility relations indexed by the set of links in the composition structure to which the glass box view is relative. A collection of collection of compatibility relations $\gamma\gamma$, indexed by the set $Comp$, is defined in terms of the collection of compatibility relations γ to which the glass box view is relative and which is assumed to be given, as follows: $\gamma\gamma=((\gamma_I)_{I \in Lnk'})_{C' \in Comp}$, with Lnk' the set of links in $CS(C',SH)=\langle Comp';Lnk';\prec;dom';cdom'\rangle$, such that for all $C' \in Comp$ and for all $I' \in Lnk$, $(\gamma\gamma_{C'})_I=\gamma_I$.

- It is straightforward to construct the glass box view on the behaviour of a primitive component from the white box view on its behaviour, as shown by Proposition 5.30 below. To avoid unnecessary extra cases in the proof of the

proposition below, the proposition is only applicable to composed components.

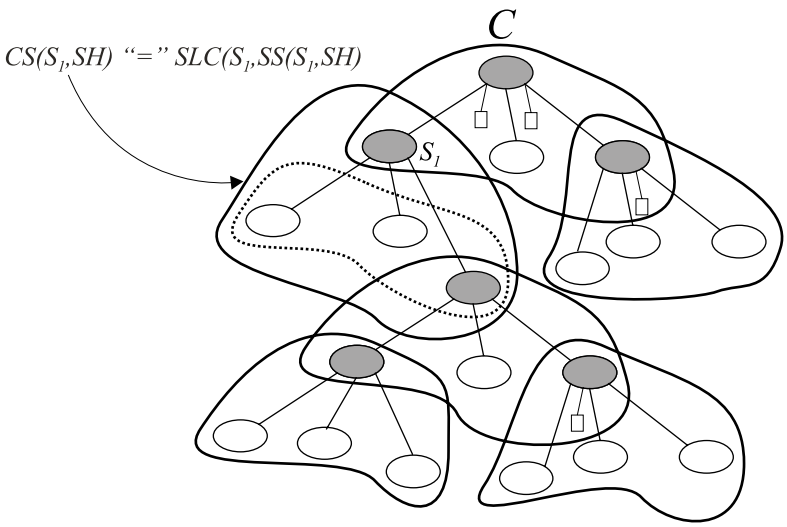

Figure 5.2: Composing the glass box view.

Proposition 5.28. *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy for a composed component C. Let $\mu\in MT_{SH}$ be a multitrace for SH such that for all $S\in Comp\cup Lnk$, $\mu_S\in Beh_{loc}(S)$. Let $\gamma=(\gamma_l)_{l\in Lnk}$ be a collection of compatibility relations and let $(\Delta_S)_{S\in Prim(SH)}$ be a collection of sets of local component traces for the primitive components in SH such that for all $S\in Prim(SH)$: $\Delta_S\subseteq Beh_{loc}(S)$. Define $\gamma\gamma=((\gamma_l)_{l\in Lnk'})_{C'\in Comp}$, with Lnk' the set of links in $CS(C',SH)=\langle Comp';Lnk';\prec';dom';cdom'\rangle$, such that for all $C'\in Comp$ and for all $l'\in Lnk$, $(\gamma\gamma_{C'})_{l'}=\gamma_{l'}$. Define $\Gamma\Gamma=((\Gamma_S)_{S\in SLC(C',CS(C',SH))})_{C'\in Comp}$ such that for all $C'\in Comp$ and for all $S\in SLC(C',CS(C',SH))$:*

$$(\Gamma\Gamma_{C'})_S=\begin{cases}\Delta_S & \text{if } S\in Prim(SH),\\ \{\mu_S\} & \text{otherwise.}\end{cases}$$

Then the following equivalence holds:

For all $C'\in Comp$: $\mu_{|SLC(C',CS(C',SH))}\in Beh_{WB}(C',CS(C',SH),\gamma\gamma_{C'},\Gamma\Gamma_{C'})$,

$\Leftrightarrow$ *$\mu\in Beh_{GB}(C,SH,\gamma,(\Delta_S)_{S\in Prim(SH)})$.*

Proposition 5.28 is important for compositional verification of compositional systems (Engelfriet, Jonker & Treur, 1999), because this proposition shows that the semantic structure supports proving global properties of a system from local properties. This topic is not further investigated in this thesis.

The glass box view on the behaviour of a component C can be compared to the parallel composition operator $\|$ in process algebra with communication (Bergstra & Klop, 1985) in the following way. In Process Algebra, the parallel composition operator takes a number of processes and returns a composed process. The behaviour of the composed process is the same as the concurrent execution of its constituent processes, taking non-local phenomena (information transmission) into account. Likewise, as indicated by Proposition 5.28, the glass box view on behaviour can be used to determine the behaviour of the concurrent execution of a set *Comp* of components given the white box views on the behaviour of each of the components in *Comp* individually. The semantic structure developed in this thesis has a very rich mechanism for composing components to form compositional structures. Therefore, this construction viewed as a composition operator is parameterised by a set of links with which the components are connected in *SH*.

The next proposition shows how the black box view can be expressed in terms of the glass box view. As stated before, all three views are, among others, relative to sets of traces of specific subcomponents. In the context of the propositions presented below, these sets of traces to which the black box view is relative, are defined in terms of the glass box view from which the black box view is generated. This is done by defining a collection of sets $(\Delta_S)_{S \in SLC(C,SH)\setminus\{C\}}$ such that for each $S \in SLC(C,SH)\setminus\{C\}$, $\Delta_S=\{\mu_S \mid \mu \in Beh_{GB}(C,SH,\gamma,\Gamma)\}$. In this proposition, the black box view is taken relative to (Δ_S), which (only) consists of sets of traces for the subcomponents of C in SH taken from the glass box view. The glass box view itself is defined relative to a collection of sets $(\Gamma_S)_{S \in Prim(SH)\setminus\{C\}}$ of traces of *the primitive components in SH*. In general, the primitive components in a structure hierarchy SH for C and the subcomponents of C are distinct.

Proposition 5.29. *Let C be a composed component, let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a non-empty structure hierarchy for C, let $\gamma=(\gamma_I)_{I \in Lnk}$ be a collection of compatibility relations and let $\Gamma=(\Gamma_S)_{S \in Prim(SH)}$ be a collection of sets of traces for the primitive components in SH. Define $\Delta=(\Delta_S)_{S \in SLC(C,SH)}$ such that for all $S \in SLC(C,SH)$, $\Delta_S=\{\mu_S \mid \mu \in Beh_{GB}(C,SH,\gamma,(\Gamma_S))\}$. Then:*

$$Beh_{BB}(C,SH,\gamma,\Delta)=\{\, \mu_C \in Beh_{loc}(C) \mid \mu \in Beh_{GB}(C,SH,\gamma,\Gamma) \,\}.$$

The last proposition presented in this section shows how the three views relate to each other in the case of primitive components:

Proposition 5.30. *Let C be a primitive component, let $SH=\langle\{C\};Lnk;\varnothing;dom;cdom\rangle$ be a structure hierarchy for C, let $\gamma=(\gamma_I)_{I \in Lnk}$ be a collection of compatibility relations, let $\Gamma=(\Gamma_S)_{S \in \{C\}}$ be a collection consisting of one set of traces such that $\Gamma_C \subseteq Beh_{loc}(C)$ and let μ be a multitrace for SH such that for all $I \in Lnk$, $\mu_I \in Beh_{loc}(I)$. Then:*

$$\mu \in Beh_{GB}(C,SH,\gamma,\Gamma) \Leftrightarrow \mu \in Beh_{WB}(C,CS(C,SH),\gamma,\Gamma) \Leftrightarrow \mu_C \in Beh_{BB}(C,SH,\gamma,\Gamma).$$

5.2.2.5 *Example*

In the next example, the behaviour of the example compositional system is described to illustrate how the behaviour of components is modelled.

Example 5.31. In this example, three views on the behaviour of the compositional system in the running example are presented. This example is based on the following assumptions:

- The three views are developed relative to the structure hierarchy *SH* presented in Example 5.10;
- A collection of compatibility relations γ is given. In conformance with Example 5.20, γ is assumed to be defined such that traces $lt_{\text{broker},1}$, $lt_{\text{broker\_to\_user\_1}}$ and $lt_{\text{user\_1},1}$ are compatible;
- Component toplevel only serves as a demarcation component for the example compositional system. Its set of information states is defined as $\mathcal{S}_{\text{toplevel}}=\{(\varnothing;\varnothing;\varnothing)\}$;
- The behaviour of OPC and ASP, the subcomponents of broker according to *SH* and Figure 4.5, is such that their collective behaviour fulfils the requirements put forward in Section 4.2.1. Moreover, sets of local component traces are given that reflect this behaviour.

As there is a component, toplevel, that represents the entire running example compositional system, the behaviour of the example compositional system is the behaviour of toplevel. So, in this example, three views on the behaviour of toplevel are developed. The first view on the behaviour of toplevel, the black box view, is obtained as follows. As for toplevel, only one state is distinguished (state $(\varnothing;\varnothing;\varnothing)$), the only possible local component trace of toplevel is the trace $lt_{\text{toplevel}}=\varnothing|\varnothing|\varnothing\rightarrow\varnothing|\varnothing|\varnothing\rightarrow\ldots$ This trace is compatible because there are no links from toplevel to any of its subcomponents or vice versa, nor are there any links from toplevel to its parent (which does not exist in *SH*) or components at the same level (which do not exist in *SH* either). Thus, $Beh_{BB}(\text{toplevel},SH,\gamma,\Gamma)=\{\varnothing|\varnothing|\varnothing\rightarrow\varnothing|\varnothing|\varnothing\rightarrow\ldots\}$ for any collection Γ of component and link traces.

The second view on the behaviour of toplevel, the white box view, is not relative to *SH*, but, according to Definition 5.25, to a composition structure. In this example, a composition structure *CS* is defined as follows: $CS=CS(\text{toplevel},SH)$. Moreover, the white box view on the behaviour of toplevel is defined relative to a collection $\Gamma=(\Gamma_S)_{S\in SLC(\text{toplevel},CS)\setminus\{\text{toplevel}\}}$ of sets of traces of the components and links in $SLC(\text{toplevel},CS)\setminus\{\text{toplevel}\}=\{\text{user\_1},\text{user\_2},\text{broker},\text{provider\_1},\text{provider\_2},\text{broker\_to\_user\_1}\}$.
Thus, before the white box view on the behaviour of toplevel can be defined, a relevant collection of component and link traces has to be defined. According to *SH*, only broker is a composed component. Therefore, a relevant collection of component and link traces can be defined by taking $Beh_{loc}(S)$ for the links and

primitive components in $SLC(\text{toplevel},SH)\backslash\{\text{toplevel}\}$ and the black box view on the behaviour of broker relative to its subcomponents. This is motivated as follows:

- Sets $Beh_{loc}(S)$ are assumed not to take any non-local constraint into account. Thus, these sets are, in general, too large: they contain traces that are not possible if non-local constraints were taken into account. However, requirements on the multitraces from which the black box and white box views on the behaviour of toplevel are taken, prevent local component and link traces in $Beh_{loc}(S)$ that do not take information transmission with components at the same level and with the parent component from appearing in these multitraces. Moreover, according to SH, there are no subcomponents of the primitive components that further constrain which local component or link traces from $Beh_{loc}(S)$ can appear in multitraces. Therefore, it is reasonable to take $Beh_{loc}(S)$ for the links and primitive components in SH. As an aside, if specific components or links are omitted from SH, and subcomponents of components that are primitive in SH can be distinguished, taking $Beh_{loc}(S)$ is not a good choice: constraints imposed by the omitted components and links are not taken into account.
- According to SH, broker is a composed component. In this case, it is not a good choice to take $Beh_{loc}(\text{broker})$. Instead, the black box view on the behaviour of broker relative to sets of local component traces of its subcomponents in SH is taken. This is a better choice for the following reason. The black box view, which is a set of component traces as required, is a subset of $Beh_{loc}(\text{broker})$ in which constraints imposed by information exchange with the subcomponents of broker is taken into account. (As an aside, it would have been possible to take the black box view not only for broker, but also for the other components in $SLC(\text{toplevel},SH)\backslash\{\text{toplevel}\}$. However, as these other components are primitive according to SH, it holds that $Beh_{BB}(S,SS(S,SH),\gamma,\varnothing)=Beh_{loc}(S)$, so formally there is no difference.)

Thus, the next step is to develop the black box view $Beh_{BB}(\text{broker},SH,\gamma,(\Delta_{S'})_{S'\in SLC(\text{broker},SH)\backslash\{\text{broker}\}})$ on the behaviour of broker. The black box view is defined relative to the behaviour of the subcomponents of broker according to SH, which are OPC and ASP. (Thus, $SLC(\text{broker},SH)\backslash\{\text{broker}\}=\{\text{APC},\text{OPC}\}$.) As OPC and ASP are primitive according to SH, $(\Delta_{S'})_{S'\in\{\text{OPC},\text{ASP}\}}$ is defined as $\Delta_{S'}=Beh_{loc}(S')$ for all $S'\in\{\text{OPC},\text{ASP}\}$. (The same motivation as for the primitive subcomponents of toplevel applies.) As stated above, it is assumed that the behaviour of OPC and ASP is such that their collective behaviour fulfils the requirements put forward in Section 4.2.1. Therefore, trace $lt_{\text{broker},1}$ presented in Example 5.14, is an example element of $Beh_{BB}(\text{broker},SH,\gamma,(\Delta_{S'})_{S'\in\{\text{OPC},\text{ASP}\}})$, which is a set of local component traces generated from compatible multitraces in $MT_{|\,SLC(\text{broker},SH)}$.

The white box view on the behaviour of toplevel can now be defined relative to the collection of sets $(\Gamma_S)_{S\in SLC(\text{toplevel},SH)}$ defined such that

$\Gamma_{\text{broker}}=Beh_{BB}(\text{broker},SH,\gamma,(\Delta_{S'})_{S'\in\{\text{OPC,ASP}\}})$ and $\Gamma_S=Beh_{loc}(S)$ for $S\in(SLC(\text{toplevel},SH)\setminus\{\text{toplevel}\})\setminus\{\text{broker}\}$. The white box view is a set of compatible multitraces μ from MT_{CS} such that (among other requirements) for each $S\in SLC(\text{toplevel},CS)\setminus\{\text{toplevel}\}$, $\mu_S\in\Gamma_S$. This requirement holds for the following multitrace, as the traces $lt_{\text{user\_1},1}$, $lt_{\text{broker\_to\_user\_1}}$ and $lt_{\text{broker},1}$ are compatible:

$$\begin{aligned} mt_1 \;=\; \{&\langle\text{toplevel};lt_{\text{toplevel}}\rangle,\langle\text{user\_1};lt_{\text{user\_1},1}\rangle,\langle\text{broker\_to\_user\_1};lt_{\text{broker\_to\_user\_1}}\rangle,\\ &\langle\text{broker};lt_{\text{toplevel}}\rangle,\langle\text{user\_2};lt_{\text{user\_2}}\rangle,\langle\text{provider\_1};lt_{\text{provider\_1}}\rangle,\\ &\langle\text{provider\_2};lt_{\text{provider\_1}}\rangle\}, \end{aligned}$$

where $lt_{\text{user\_2}}\in Beh_{loc}(\text{user\_2})$, $lt_{\text{provider\_1}}\in Beh_{loc}(\text{provider\_1})$ and $lt_{\text{provider\_2}}\in Beh_{loc}(\text{provider\_2})$. (The other traces referred to in mt_1 have been introduced before.) Other elements of $Beh_{WB}(\text{toplevel},SH,\gamma,\Gamma)$ can be found in a similar way.

To finalise the example, also the glass box view on the behaviour of toplevel is given. The glass box view is defined relative to a collection of sets of traces of the primitive components in SH, the elements of the set $Prim(SH)=\{\text{user\_1},\text{user\_2},\text{provider\_1},\text{provider\_2},\text{OPC},\text{ASP}\}$. This collection $\Delta'=(\Delta'_S)_{S\in Prim(SH)}$ is defined such that for all $S\in Prim(SH)$, $\Delta'_S=Beh_{loc}(S)$. (The same motivation as for the primitive subcomponents of toplevel applies.) An example element of $Beh_{GB}(\text{toplevel},SH,\gamma,\Delta')$ is given by the following multitrace to show how the behaviour of the example multi-agent system is modelled in the glass box view. However, it is left to the reader to check that the hierarchical multitrace given is indeed an element of $Beh_{GB}(\text{toplevel},SH,\gamma,\Delta')$. (This is quite straightforward.)

$$\begin{aligned} mt_2 \;=\; \{&\langle\text{toplevel};lt_{\text{toplevel}}\rangle,\langle\text{user\_1};lt_{\text{user\_1},1}\rangle,\langle\text{broker\_to\_user\_1};lt_{\text{broker\_to\_user\_1}}\rangle,\\ &\langle\text{broker};lt_{\text{toplevel}}\rangle,\langle\text{OPC};lt_{\text{OPC}}\rangle,\langle\text{ASP};lt_{\text{ASP}}\rangle,\langle\text{user\_2};lt_{\text{user\_2},1}\rangle,\\ &\langle\text{provider\_1};lt_{\text{provider\_1}}\rangle,\langle\text{provider\_2};lt_{\text{provider\_1}}\rangle\}, \end{aligned}$$

This multitrace is an element of MT_{SH}. The hierarchical structure on mt_2 is imposed by the ordering $\prec$ of SH. This ordering is depicted in Figure 5.3 as a tree. ■

The example presented in this section exhibits two aspects of compositionality. In the first place, the example compositional system consists of five components, one of which itself consists of two subcomponents. The composition of the system is reflected in the formal description of its structure by the structure hierarchy presented in Example 5.10. In the second place, the composition of the example system is also reflected in the behaviour of the system, in particular in the glass box view on its behaviour: the compositional structure of the example system induces the structure of the multitraces that comprises the glass box view on the behaviour of the example system. Moreover, a specific feature of the semantic structure developed in this thesis is that the behaviour of a composed component is not only determined by the behaviour of its subcomponents but also by the local behaviour of the composed component itself. (In the previous examples, $Beh_{loc}(\text{broker})$, which was assumed to be given, represented this factor in the total behaviour of the composed component broker.)

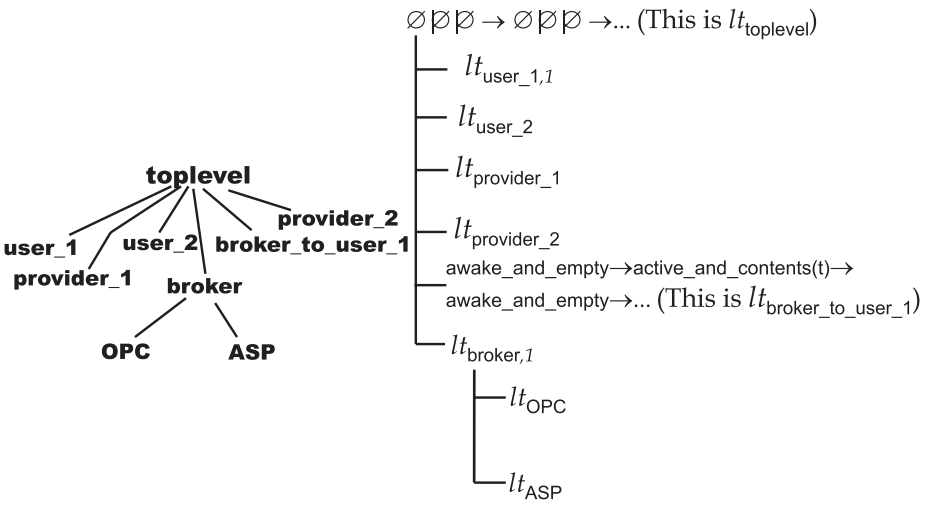

Figure 5.3: A structure hierarchy SH and an element of Beh_{GB}(toplevel,SH,γ,Δ').

5.3 Summary and Outlook

As explained in Chapter 1, this thesis aims at developing a formal, compositional semantic structure for multi-agent system dynamics. The semantic structure, which consists of constructs for building compositional systems, is presented in this chapter. The two most important definitions are the definition of a structure hierarchy and the definition of the glass box view on the behaviour of a component. A structure hierarchy enables an entire compositional system to be defined, while the glass box view defines the most complete picture of the behaviour of such a system. The definition of the glass box view, however, is relative to, among others, a collection of compatibility relations, which are discussed further in Chapter 6. Chapter 7 defines a more global notion of state as an extension of the semantic structure. After further development of the semantic structure in Chapter 6 and Chapter 7, facilities for separated, domain independent control are added in Chapter 8. Finally, in Chapter 9, the semantic structure is applied in the development of a semantics for DESIRE, an extensive compositional modelling framework for multi-agent systems.

5.4 Proofs

This thesis employs a notation for structuring proofs proposed by Lamport (1995). In Section 5.4.1, this notation is introduced. Proofs of the propositions and theorems presented in this chapter are provided in Section 5.4.2.

5.4.1 A Note on Proof Notation

Lamport (1995) advocates the use of a hierarchically structured proof style as a replacement for unstructured proofs presented in the form of (English) prose. The proof style he advocates is a refinement of natural deduction, in which proof steps are numbered according to a specific scheme, and the structure of the proof is represented by the indentation in its textual form. The following principles are applied:

- A proof has a hierarchical structure, which is represented in its textual form by indentation. Each level consists of proof steps, which are themselves proven at the next level. The proof steps at a specific level together prove the statement at the next higher level. A proof of a very simple statement can be given directly at the level of that statement. A proof can be read level by level.
- Proof steps are numbered by a decimal numbering system similar to the numbering of the section headings in this thesis. As step numbers at deeper levels of the proof can become quite long, and as it is easy to confuse e.g. 3.1.1.1.2 with 3.1.1.2, an abbreviation is used: a number such as 3.1.1.2 is written as ⟨4⟩2 (a four-part number ending in 2), while 3.1.1.1.2 is written as ⟨5⟩2 (a five-part number ending in 2). The laws of natural deduction guarantee that the abbreviated form suffices: a step such as 3.1.1.1.2 can only be used after it is proven, but because it is proven under the assumption of step 3.1.1.1, it can only be referred to in the proof of its parent. In the proof of its parent, step 3.1.1.1.2 is the only five-part number ending in 2. Parts of a proof step are referred to by appending the part number to the step number, separated by a semicolon e.g., ⟨3⟩:1. An assumption at the highest level is referred to by ⟨0⟩.
- A proof by cases is introduced by the word 'Case', followed by an expression characterising the case, which is an assumption in the proof of the case. A proof by cases ends with a proof that shows that all cases are covered.

According to Lamport, this proof style results in more rigorous and less error prone proofs that are easier to read, especially in the area of correctness proofs for algorithms, where proofs are seldom deep, but have considerable detail.

5.4.2 Proofs of Propositions and Theorems

In Section 5.2.2, three propositions are presented that relate the three views on behaviour developed in that section.

Proposition 5.28. *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy for a composed component C. Let $\mu \in MT_{SH}$ be a multitrace for SH such that for all*

$S \in Comp \cup Lnk$, $\mu_S \in Beh_{loc}(S)$. Let $\gamma=(\gamma_I)_{I \in Lnk}$ be a collection of compatibility relations and let $(\Delta_S)_{S \in Prim(SH)}$ be a collection of sets of local component traces for the primitive components in SH such that for all $S \in Prim(SH)$: $\Delta_S \subseteq Beh_{loc}(S)$. Define $\gamma\gamma=((\gamma_I)_{I \in Lnk'})_{C' \in Comp}$, with Lnk' the set of links in $CS(C',SH)=\langle Comp';Lnk';\prec;dom';cdom'\rangle$, such that for all $C' \in Comp$ and for all $I' \in Lnk$, $(\gamma\gamma_{C'})_I=\gamma_I$. Define $\Gamma\Gamma=((\Gamma_S)_{S \in SLC(C',CS(C',SH))})_{C' \in Comp}$ such that for all $C' \in Comp$ and for all $S \in SLC(C',CS(C',SH))$:

$$(\Gamma\Gamma_{C'})_S=\begin{cases}\Delta_S & \text{if } S \in Prim(SH),\\ \{\mu_S\} & \text{otherwise.}\end{cases}$$

Then the following equivalence holds:

For all $C' \in Comp$: $\mu_{|\,SLC(C',CS(C',SH))} \in Beh_{WB}(C',CS(C',SH),\gamma\gamma_{C'},\Gamma\Gamma_{C'})$,

$\Leftrightarrow$ $\mu \in Beh_{GB}(C,SH,\gamma,(\Delta_S)_{S \in Prim(SH)})$.

Proof. Proof sketch: the proof of the first implication (the implication from left to right) is as follows. For a multitrace μ that complies with a number of assumptions, it is proven that $\mu \in Beh_{GB}(C,SH,\gamma,(\Delta_S)_{S \in Prim(SH)})$. Thus, three requirements for elements of $Beh_{GB}(C,SH,\gamma,(\Delta_S)_{S \in Prim(SH)})$ are proven. (Actually, one of the three, the requirement that for all $S \in Comp \cup Lnk$, $\mu_S \in Beh_{loc}(S)$, is an assumption, so only two requirements are proven.) The requirements are checked by induction to the structure of SH. The base case for induction is identified as $SH=CS(C,SH)$. Thus, the base case is formed of structure hierarchies consisting of precisely two levels. (It is assumed that C is composed.) Then, induction can be applied based on Definition 5.24 (the definition of $Prim(SH)$), which guarantees that for each primitive component C' in SH, there is a subcomponent C'' of C such that C' is a primitive component in $SS(C'',SH)$. Technically, the induction steps in the proof are identified as $\exists C' \in Subc(C,SH): C' \notin Prim(SH)$. The second implication (the implication from right to left) consists of straightforward expansions of the definitions.

Assume:
1. $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ is a structure hierarchy for a composed component C.
2. μ is a multitrace for SH such that for all $S \in Comp \cup Lnk$, $\mu_S \in Beh_{loc}(S)$.
3. $\Gamma\Gamma=((\Gamma_S)_{S \in SLC(C',CS(C',SH))})_{C' \in Comp}$ such that for all $C' \in Comp$ and for all $S \in SLC(C',CS(C',SH))$:

$$(\Gamma\Gamma_{C'})_S=\begin{cases}\Delta_S & \text{if } S \in Prim(SH),\\ \{\mu_S\} & \text{otherwise.}\end{cases}$$

4. $\gamma\gamma=((\gamma_I)_{I \in Lnk'})_{C' \in Comp}$ such that for all $C' \in Comp$ and for all $I' \in Lnk$, $(\gamma\gamma_{C'})_I=\gamma_I$.

Prove: for all $C' \in Comp$: $\mu_{|\,SLC(C',CS(C',SH))} \in Beh_{WB}(C',CS(C',SH),\gamma\gamma_{C'},\Gamma\Gamma_{C'}) \Leftrightarrow$ $\mu \in Beh_{GB}(C,SH,\gamma,(\Delta_S)_{S \in Prim(SH)})$.

⟨1⟩1. $\forall C' \in Comp: \mu_{|SLC(C',CS(C',SH))} \in Beh_{WB}(C',CS(C',SH),\gamma\gamma_{C'},\Gamma\Gamma_{C'}) \Rightarrow \mu \in Beh_{GB}(C,SH,\gamma,(\Delta_S)_{S \in Prim(SH)})$.

Assume: $\forall C' \in Comp: \mu_{|SLC(C',CS(C',SH))} \in Beh_{WB}(C',CS(C',SH),\gamma\gamma_{C'},\Gamma\Gamma_{C'})$.

Prove: $\mu \in Beh_{GB}(C,SH,\gamma,(\Delta_S)_{S \in Prim(SH)})$.

⟨2⟩1. For each $C' \in Prim(SH)$, $\exists C'' \in Subc(C,SH): C' \in Prim(SS(C'',SH))$.

Proof: by assumption ⟨0⟩:1, C is a composed component. By Definition 5.24, $Prim(SH)=\bigcup_{C' \in Subc(C,SH)} Prim(SS(C',SH))$.

⟨2⟩2. μ is compatible for γ.

⟨3⟩1. Let $I \in Lnk$. There is a $C'' \in Comp$ such that $I \in SLC(C'',CS(C'',SH))$.

Proof: by the definition of a structure hierarchy.

⟨3⟩2. $\mu_{|SLC(C'',CS(C'',SH))} \in Beh_{WB}(C'',CS(C'',SH),\gamma\gamma_{C''},\Gamma\Gamma_{C''})$.

Proof: by assumption ⟨1⟩.

⟨3⟩3. $\mu_{|SLC(C'',CS(C'',SH))}$ is compatible for $\gamma\gamma_{C''}$.

Proof: by step ⟨3⟩2 and the definition of $Beh_{WB}(C'',CS(C'',SH),\gamma\gamma_{C''},\Gamma\Gamma_{C''})$.

⟨3⟩4. $\langle \mu_{dom(I)};\mu_I;\mu_{cdom(I)} \rangle \in (\gamma\gamma_{C''})_I$.

Proof: by step ⟨3⟩3 and the definition of compatible multitraces, for each link $I' \in Lnk'$ where Lnk' is the set of links in $CS(C'',SH)$, $\langle \mu_{dom(I')};\mu_{I'};\mu_{cdom(I')} \rangle \in (\gamma\gamma_{C''})_{I'}$. By step ⟨3⟩1, $I \in Lnk'$.

⟨3⟩5. For all $I \in Lnk$, $\langle \mu_{dom(I)};\mu_I;\mu_{cdom(I)} \rangle \in \gamma_I$.

Proof: by steps ⟨3⟩1 and ⟨3⟩4 and assumption ⟨0⟩:4.

⟨3⟩6. Q.E.D.

Proof: by step ⟨3⟩5 and the definition of compatible multitraces.

⟨2⟩3. $\forall C' \in Prim(SH)\backslash\{C\}: \mu_{C'} \in \Delta_{C'}$.

⟨3⟩1. Case: $SH=CS(C,SH)$.

⟨4⟩1. $Prim(SH)\backslash\{C\} \subseteq SLC(C,CS(C,SH))\backslash\{C\}$.

Proof: by assumption ⟨3⟩.

⟨4⟩2. $\forall S \in SLC(C,SH)\backslash\{C\}: (\mu_{|SLC(C,CS(C,SH))})_S \in (\Gamma\Gamma_C)_S$.

Proof: by assumption ⟨1⟩, $\mu_{|SLC(C,CS(C,SH))} \in Beh_{WB}(C,CS(C,SH),\gamma\gamma_C,\Gamma\Gamma_C)$, and by the definition of $Beh_{WB}(C,CS(C,SH),\gamma\gamma_C,\Gamma\Gamma_C)$, $(\mu_{|SLC(C,CS(C,SH))})_S \in (\Gamma\Gamma_C)_S$.

⟨4⟩3. $\forall C' \in Prim(SH)\backslash\{C\}: \mu_{C'} \in (\Gamma\Gamma_C)_{C'}$.

Proof: by assumption ⟨3⟩, $\mu_{|SLC(C,CS(C,SH))}=\mu$, and by steps ⟨4⟩1 and ⟨4⟩2, $\mu_{C'} \in (\Gamma\Gamma_C)_{C'}$.

⟨4⟩4. $\forall C' \in Prim(SH)\backslash\{C\}: (\Gamma\Gamma_C)_{C'}=\Delta_{C'}$.

Proof: by assumption ⟨0⟩:3 (definition of $\Gamma\Gamma_{C'}$).

⟨4⟩5. Q.E.D.

Proof: by steps ⟨4⟩3 and ⟨4⟩4.

⟨3⟩2. Case: 1. $\exists C' \in Subc(C,SH): C' \notin Prim(SH)$.

2. $\forall S \in Subc(C,SH): \forall C' \in Prim(SS(S,SH))\backslash\{C\}: \mu_{C'} \in \Delta_{C'}$.

⟨4⟩1. Let $C' \in Prim(SH) \setminus \{C\}$. $\exists S \in Subc(C,SH): C' \in Prim(SS(S,SH))$.
Proof: by assumption ⟨3⟩:1 and step ⟨2⟩1.

⟨4⟩2. Q.E.D.
Proof: by step ⟨4⟩1 and assumption ⟨3⟩:2.

⟨3⟩3. Q.E.D.
Proof: by steps ⟨3⟩1, ⟨3⟩2 and induction to the structure of SH.

⟨2⟩4. Q.E.D.
Proof: by steps ⟨2⟩2 and ⟨2⟩3, assumption ⟨0⟩:2 and the definition of $Beh_{GB}(C,SH,\gamma,(\Delta_S)_{S \in Prim(SH)})$.

⟨1⟩2. $\mu \in Beh_{GB}(C,SH,\gamma,(\Delta_S)_{S \in Prim(SH)}) \Rightarrow$
$\forall C' \in Comp: \mu_{| SLC(C',CS(C',SH))} \in Beh_{WB}(C',CS(C',SH),\gamma\gamma_{C'},\Gamma\Gamma_{C'})$.

Assume: 1. μ is a multitrace for SH such that $\mu \in Beh_{GB}(C,SH,\gamma,(\Delta_S)_{S \in Prim(SH)})$.
2. $C' \in Comp$ and μ' is a multitrace for $CS(C',SH)$ such that $\mu' = \mu_{| SLC(C',CS(C',SH))}$.

Prove: for all $C' \in Comp: \mu_{| SLC(C',CS(C',SH))} \in Beh_{WB}(C',CS(C',SH),\gamma\gamma_{C'},\Gamma\Gamma_{C'})$.

⟨2⟩1. $\mu' \in MT_{CS(C',SH)}$ is compatible for $\gamma\gamma_{C'}$.

⟨3⟩1. μ is compatible for γ.
Proof: by definition of $Beh_{GB}(C,SH,\gamma,(\Delta_S)_{S \in Prim(SH)})$.

⟨3⟩2. For all $I \in Lnk$: $\langle \mu_{dom(I)};\mu_I;\mu_{cdom(I)} \rangle \in \gamma_I$.
Proof: by step ⟨3⟩1 and the definition of compatible multitraces.

⟨3⟩3. Let Lnk' be the set of links in $SLC(C',CS(C',SH))$. For all $I' \in Lnk'$: $\langle \mu'_{dom(I')};\mu'_{I'};\mu'_{cdom(I')} \rangle \in \gamma_{I'}$.
Proof: by step ⟨3⟩2 and $Lnk' \subseteq Lnk$, and by assumption ⟨1⟩:2, $\mu' = \mu_{| SLC(C',CS(C',SH))}$.

⟨3⟩4. $\forall I' \in Lnk': \langle \mu'_{dom(I')};\mu'_{I'};\mu'_{cdom(I')} \rangle \in (\gamma\gamma_{C'})_{I'}$.
Proof: by step ⟨3⟩3 and assumption ⟨0⟩:4.

⟨3⟩5. Q.E.D.
Proof: by step ⟨3⟩4 and the definition of compatible multitraces.

⟨2⟩2. For all $S \in SLC(C',CS(C',SH)): \mu'_S \in (\Gamma\Gamma_{C'})_S$.

⟨3⟩1. Case: $C' \in Prim(SH)$.
Proof: by assumption ⟨2⟩, $SLC(C',CS(C',SH))=\{C\}$, thus $S=C'$. By assumption ⟨0⟩:3, $(\Gamma\Gamma_{C'})_S=\Delta_S$. By the definition of $Beh_{GB}(C,SH,\gamma,(\Delta_S)_{S \in Prim(SH)})$, $\mu_S \in \Delta_S$ and by assumption ⟨1⟩:2, $\mu'_S=\mu_S \in \Delta_S=(\Gamma\Gamma_{C'})_S$.

⟨3⟩2. Case: $C' \notin Prim(SH)$.
Proof: by assumption ⟨0⟩:3, $(\Gamma\Gamma_{C'})_S=\{\mu_S\}$. By assumption ⟨1⟩:2, $\mu'_S=\mu_S$. Therefore, $\mu'_S=\mu_S \in \{\mu_S\}=(\Gamma\Gamma_{C'})_S$.

⟨3⟩3. Q.E.D.
Proof: steps ⟨3⟩1 and ⟨3⟩2 list all cases.

⟨2⟩3. Q.E.D.

Proof: by steps ⟨2⟩1 and ⟨2⟩2 and the definition of $Beh_{WB}(C',CS(C',SH),\gamma\gamma_{C'},\Gamma\Gamma_{C'})$.

⟨1⟩3. Q.E.D.
Proof: by steps ⟨1⟩2 and ⟨1⟩2.

Proposition 5.29. *Let C be a composed component, let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a non-empty structure hierarchy for C, let $\gamma=(\gamma_I)_{I\in Lnk}$ be a collection of compatibility relations and let $\Gamma=(\Gamma_S)_{S\in Prim(SH)}$ be a collection of sets of traces for the primitive components in SH. Define $\Delta=(\Delta_S)_{S\in SLC(C,SH)}$ such that for all $S\in SLC(C,SH)$, $\Delta_S=\{\mu_S \mid \mu\in Beh_{GB}(C,SH,\gamma,(\Gamma_S))\}$. Then:*

$$Beh_{BB}(C,SH,\gamma,\Delta)=\{\ \mu_C\in Beh_{loc}(C) \mid \mu\in Beh_{GB}(C,SH,\gamma,\Gamma)\ \}.$$

Proof. Proof sketch: at the highest level, the proof consists of two inclusions. The proof of the first inclusion is complicated. For an arbitrary element of $Beh_{BB}(C,SH,\gamma,\Delta)$ (a local component trace), it is proven that the multitrace from which this element is taken is an element of $Beh_{GB}(C,SH,\gamma,\Gamma)$. Induction on the structure of SH is applied twice to prove two requirements on elements of $Beh_{GB}(C,SH,\gamma,\Gamma)$. Intuitively, induction is needed to ensure that local component or link traces that cannot occur in elements of $Beh_{GB}(C,SH,\gamma,\Gamma)$, do not occur in the multitraces from which $Beh_{BB}(C,SH,\gamma,\Delta)$ is built. The proof of the second inclusion consists of straightforward expansions of definitions.

Assume: 1. C is a composed component,
2. $\Delta=(\Delta_S)_{S\in SLC(C,SH)}$ such that for all $S\in SLC(C,SH)$, $\Delta_S=\{\mu_S \mid \mu\in Beh_{GB}(C,SH,\gamma,\Gamma)\}$.

Prove: $Beh_{BB}(C,SH,\gamma,\Delta)=\{\ \mu_C\in Beh_{loc}(C) \mid \mu\in Beh_{GB}(C,SH,\gamma,\Gamma)\ \}$.

⟨1⟩1. $Beh_{BB}(C,SH,\gamma,\Delta)\subseteq\{\ \mu_C\in Beh_{loc}(C) \mid \mu\in Beh_{GB}(C,SH,\gamma,\Gamma)\ \}$.
Assume: $LT\in Beh_{BB}(C,SH,\gamma,\Delta)$.
Prove: $LT\in\{\ \mu_C\in Beh_{loc}(C) \mid \mu\in Beh_{GB}(C,SH,\gamma,\Gamma)\ \}$.
⟨2⟩1. There is a $\mu\in MT_{SH}$ compatible for γ such that $\mu_C=LT\in\Delta_S\subseteq Beh_{loc}(C)$ and $\forall S\in SLC(C,SH)$, $\mu_S\in\Delta_S$.
Proof: by definition of $Beh_{BB}(C,SH,\gamma,\Delta)$.
⟨2⟩2. $\forall C'\in Prim(SH)\setminus\{C\}$: $\mu_{C'}\in\Gamma_{C'}$.
⟨3⟩1. Case: $SH=CS(C,SH)$.
⟨4⟩1. $Prim(SH)=Comp\setminus\{C\}\subseteq SLC(C,SH)$.
Proof: by assumptions ⟨3⟩ and ⟨0⟩:1, all subcomponents of C are primitive.
⟨4⟩2. $\forall C'\in Prim(SH)$: $\mu_{C'}\in\Delta_{C'}$.
Proof: by definition of $Beh_{BB}(C,SH,\gamma,\Delta)$, $\forall S\in SLC(C,SH)$: $\mu_S\in\Delta_S$ and by step ⟨4⟩1, $Prim(SH)\subseteq SLC(C,SH)$.
⟨4⟩3. $\forall C'\in Prim(SH)$: $\exists\mu'\in MT_{SH}$ such that $\mu'\in Beh_{GB}(C,SH,\gamma,\Gamma)$ and $\mu_{C'}=\mu'_{C'}$.

Proof: by step ⟨4⟩2 and assumption ⟨0⟩:2 (definition of Δ_S).

⟨4⟩4. $\forall C' \in Prim(SH): \exists \mu' \in MT_{SH}: \mu'_{C'} \in \Gamma_{C'}$ and $\mu_{C'}=\mu'_{C'}$.
Proof: by step ⟨4⟩3 and definition of $Beh_{GB}(C,SH,\gamma,\Gamma)$.

⟨4⟩5. Q.E.D.
Proof: by step ⟨4⟩4.

⟨3⟩2. Case: 1. $\exists C' \in Subc(C,SH): C' \notin Prim(SH)$.
2. $\forall S \in SLC(C,SH)\backslash\{C\}: \forall C' \in Prim(SS(S,SH)): \mu_{C'} \in \Gamma_{C'}$.

⟨4⟩1. $\forall C' \in Prim(SH)\backslash\{C\}: \exists S \in SLC(C,SH)\backslash\{C\}: C' \in Prim(SS(S,SH))$.
Proof: by assumption ⟨3⟩:1 and the definition of $Prim(SH)$.

⟨4⟩2. Q.E.D.
Proof: by step ⟨4⟩1 and assumption ⟨3⟩:2.

⟨3⟩3. Q.E.D.
Proof: by step ⟨3⟩1, ⟨3⟩2 and induction to structure of SH.

⟨2⟩3. $\forall S \in Comp \cup Lnk: \mu_S \in Beh_{loc}(S)$.

⟨3⟩1. Case: $SH=CS(C,SH)$.

⟨4⟩1. Case: $S=C$.
Proof: $\mu_C \in \Delta_S \subseteq Beh_{loc}(C)$ by definition of $Beh_{BB}(C,CS(C,SH),\gamma,\Delta)$

⟨4⟩2. Case: $S \in (Comp\backslash\{C\}) \cup Lnk$.
Proof: $S \in SLC(C,SH)\backslash\{C\}$ because $\forall I \in Lnk, I \prec^* C$. Therefore, $\mu_S \in \Delta_S$. Thus, $\exists \mu' \in Beh_{GB}(C,SH,\gamma,\Gamma): \mu_S=\mu'_S$ and $\forall S \in (Comp\backslash\{C\}) \cup Lnk: \mu'_S \in Beh_{loc}(S)$ by the definition of $Beh_{GB}(C,SH,\gamma,\Gamma)$.

⟨4⟩3. Q.E.D.
Proof: steps ⟨4⟩1 and ⟨4⟩2 list all cases.

⟨3⟩2. Case: 1. $\exists C' \in Subc(C,SH): C' \notin Prim(SH)$.
2. $\forall S' \in SLC(C,SH)\backslash\{C\}: \forall S'' \in Comp' \cup Lnk': \mu_{S''} \in Beh_{loc}(S'')$ with $SS(S,SH)=\langle Comp';Lnk';\prec;dom';cdom'\rangle$,
3. $S \in Comp \cup Lnk$.

⟨4⟩1. Case: $S=C$
Proof: by definition of $Beh_{BB}(C,SH,\gamma,\Delta)$, $\mu_C \in \Delta_C \subseteq Beh_{loc}(C)$.

⟨4⟩2. Case: $S \in Lnk$ such that $S \prec C''$ for $C'' \in Subc(C,SH)$, or $S \in Comp\backslash\{C\}$.
Proof: $\exists S' \in SLC(C,SH)\backslash\{C\}: S \in Comp' \cup Lnk'$ for $SS(S',SH)=\langle Comp';Lnk';\prec;dom';cdom'\rangle$. By assumption ⟨3⟩:2, $\mu_S \in Beh_{loc}(S)$ for $S \in Comp' \cup Lnk'$.

⟨4⟩3. Case: $S \in Lnk$ such that $S \prec C$.
Proof: $S \in SLC(C,SH)$. Therefore, $\mu_S \in \Delta_S$. Thus, $\exists \mu' \in Beh_{GB}(C,SH,\gamma,\Gamma): \mu_S=\mu'_S$ and $\forall S \in (Comp\backslash\{C\}) \cup Lnk: \mu'_S \in Beh_{loc}(S)$ by the definition of $Beh_{GB}(C,SH,\gamma,\Gamma)$.

⟨4⟩4. Q.E.D.
Proof: step ⟨4⟩1, ⟨4⟩2 and ⟨4⟩3 list all cases.
⟨3⟩3. Q.E.D.
Proof: by step ⟨3⟩1, ⟨3⟩2 and induction to structure of *SH*.
⟨2⟩4. Q.E.D.
Proof: by step ⟨2⟩1, ⟨2⟩2 and ⟨2⟩3, $\mu \in Beh_{GB}(C,SH,\gamma,\Gamma)$. $\mu_C = LT \in Beh_{loc}(C)$, thus, $LT \in \{\mu_C \in Beh_{loc}(C) \mid \mu \in Beh_{GB}(C,SH,\gamma,\Gamma)\}$.

⟨1⟩2. $\{\mu_C \in Beh_{loc}(C) \mid \mu \in Beh_{GB}(C,SH,\gamma,\Gamma)\} \subseteq Beh_{BB}(C,SH,\gamma,\Delta)$.
Assume: $LT \in \{\mu_C \in Beh_{loc}(C) \mid \mu \in Beh_{GB}(C,SH,\gamma,\Gamma)\}$.
Prove: $LT \in Beh_{BB}(C,SH,\gamma,\Delta)$.
⟨2⟩1. $\exists \mu \in Beh_{GB}(C,SH,\gamma,\Gamma)$: $LT = \mu_C \in Beh_{loc}(C)$.
Proof: by assumption ⟨1⟩.
⟨2⟩2. μ is compatible for γ, $\forall C' \in Prim(SH) \setminus \{C\}$: $\mu_{C'} \in \Gamma_{C'}$ and $\forall S \in Lnk \cup Comp$: $\mu_S \in Beh_{loc}(S)$.
Proof: by step ⟨2⟩1 and definition of $Beh_{GB}(C,SH,\gamma,\Gamma)$.
⟨2⟩3. $\forall S \in SLC(C,SH)$: $\mu_S \in \Delta_S$.
Proof: by step ⟨2⟩2 and assumption ⟨0⟩ (definition of Δ).
⟨2⟩4. $\forall I \in Lnk$ such that $dom(I)=cdom(I)=\{C\}$: $\mu_I \in Beh_{loc}(I)$.
Proof: by step ⟨2⟩2.
⟨2⟩5. Q.E.D.
Proof: by step ⟨2⟩3, ⟨2⟩4, compatibility of μ and definition of $Beh_{BB}(C,SH,\gamma,\Delta)$.

⟨1⟩3. *Q.E.D*.
Proof: by steps ⟨1⟩1 and ⟨1⟩2.

Proposition 5.30. *Let C be a primitive component, let SH=⟨{C};Lnk;∅;dom;cdom⟩ be a structure hierarchy for C, let* $\gamma=(\gamma_I)_{I \in Lnk}$ *be a collection of compatibility relations, let* $\Gamma=(\Gamma_S)_{S \in \{C\}}$ *be a collection consisting of one set of traces such that* $\Gamma_C \subseteq Beh_{loc}(C)$ *and let* μ *be a multitrace for SH such that for all* $I \in Lnk$, $\mu_I \in Beh_{loc}(I)$. *Then:*

$$\mu \in Beh_{GB}(C,SH,\gamma,\Gamma) \Leftrightarrow \mu \in Beh_{WB}(C,CS(C,SH),\gamma,\Gamma) \Leftrightarrow \mu_C \in Beh_{BB}(C,SH,\gamma,\Gamma).$$

Proof. Proof sketch: the proof consists of straightforward expansions of the definitions. Note that, because *C* is primitive, *Lnk*=*Lnk'*, where *Lnk'* is the set of links in *CS(C,SH)*. Therefore, the collection of sets of link traces γ conforms to the requirements of the white box view as well as of the glass box and black box views.
Assume: 1. *C* is a primitive component,
2. *SH*=⟨{*C*};*Lnk*;∅;*dom*;*cdom*⟩ is a structure hierarchy for *C*,
3. $\Gamma=(\Gamma_S)_{S \in \{C\}}$ is a collection consisting of one set of traces such that $\Gamma_C \subseteq Beh_{loc}(C)$,
4. μ is a multitrace for *SH* such that for all $I \in Lnk$, $\mu_I \in Beh_{loc}(I)$.

Prove: $\mu \in Beh_{GB}(C,SH,\gamma,\Gamma) \Leftrightarrow \mu \in Beh_{WB}(C,CS(C,SH),\gamma,\Gamma) \Leftrightarrow \mu_C \in Beh_{BB}(C,SH,\gamma,\Gamma)$.
⟨1⟩1. $\mu \in Beh_{GB}(C,SH,\gamma,\Gamma) \Rightarrow \mu \in Beh_{WB}(C,CS(C,SH),\gamma,\Gamma)$.

Assume: $\mu \in Beh_{GB}(C,SH,\gamma,\Gamma)$.

Prove: $\mu \in Beh_{WB}(C,CS(C,SH),\gamma,\Gamma)$.

⟨2⟩1. $\mu \in MT_{SH}=MT_{|\{C\}}=MT_{|SLC(C,SH)}$ is compatible for γ, and $\mu_C \in \Gamma_C$.
Proof: by definition of $Beh_{GB}(C,SH,\gamma,\Gamma)$.

⟨2⟩2. $\forall S \in SLC(C,CS(C,SH))$: $\mu_S \in \Gamma_S$.
Proof: by assumption ⟨0⟩:1, $SLC(C,CS(C,SH))=\{C\}$. By step ⟨2⟩1, $\mu_C \in \Gamma_S$.

⟨2⟩3. Q.E.D.
Proof: by steps ⟨2⟩1 and ⟨2⟩2, and the definition of $Beh_{WB}(C,CS(C,SH),\gamma,\Gamma)$.

⟨1⟩2. $\mu \in Beh_{WB}(C,CS(C,SH),\gamma,\Gamma) \Rightarrow \mu_C \in Beh_{BB}(C,SH,\gamma,\Gamma)$.

Assume: $\mu \in Beh_{WB}(C,CS(C,SH),\gamma,\Gamma)$.

Prove: $\mu_C \in Beh_{BB}(C,SH,\gamma,\Gamma)$.

⟨2⟩1. $\mu \in MT_{CS(C,SH)}=MT_{|\{C\}}=MT_{SH}$ is compatible for γ and $\forall S \in SLC(C,SH)\setminus\{C\}$: $\mu_S \in \Gamma_S$.
Proof: by definition of $Beh_{WB}(C,CS(C,SH),\gamma,\Gamma)$ and by assumption ⟨0⟩:1, $CS(C,SH)=SH$.

⟨2⟩2. $\forall I \in Lnk$ such that $dom(I)=cdom(I)=\{C\}$, $\mu_I \in Beh_{loc}(I)$.
Proof: by assumption ⟨0⟩:4.

⟨2⟩3. Q.E.D.
Proof: by steps ⟨2⟩1 and ⟨2⟩2 and the definition of $Beh_{BB}(C,SH,\gamma,\Gamma)$.

⟨1⟩3. $\mu_C \in Beh_{BB}(C,SH,\gamma,\Gamma) \Rightarrow Beh_{GB}(C,SH,\gamma,\Gamma)$.

Assume: $\mu_C \in Beh_{BB}(C,SH,\gamma,\Gamma)$.

Prove: $\mu \in Beh_{GB}(C,SH,\gamma,\Gamma)$.

⟨2⟩1. $\exists \mu' \in MT_{SH}$ such that $\mu_C=\mu'_C \in \Gamma_C \subseteq Beh_{loc}(C)$, μ' is compatible for γ, and $\forall I \in Lnk$ such that $dom(I)=cdom(I)=\{C\}$, $\mu'_I \in Beh_{loc}(I)$.
Proof: by definition of $Beh_{BB}(C,SH,\gamma,\Gamma)$.

⟨2⟩2. $\forall C' \in Prim(SH)\setminus\{C\}$: $\mu_S \in \Gamma_S$.
Proof: by assumption ⟨0⟩:1, $SLC(C,SH)=\{C\}$, and $Prim(SH)\setminus\{C\}=\varnothing$.

⟨2⟩3. $\forall S \in Comp \cup Lnk$: $\mu_S \in Beh_{loc}(S)$
Proof: assumption ⟨0⟩:3 covers $S \in Lnk$ and by step ⟨2⟩1, $\mu_C \in Beh_{loc}(C)$.

⟨2⟩4. Q.E.D.
Proof: by steps ⟨2⟩1, ⟨2⟩2 and ⟨2⟩3 and definition of $Beh_{GB}(C,SH,\gamma,\Gamma)$.

⟨1⟩4. Q.E.D.
Proof: by steps ⟨1⟩1, ⟨1⟩2, and ⟨1⟩3 and transitivity of $\Rightarrow$.

Chapter 6

Properties of Information Transmission

This chapter presents properties of information transmission. First, in Section 6.1, properties of component interfaces, and properties of local component and link traces related to information transmission are presented. In Section 6.2, properties of compatibility relations are defined. Some of these properties are related to properties of component interfaces defined in Section 6.1. Finally, in Section 6.3, a discussion of the notion of a compatibility relation is presented.

6.1 Properties of Interfaces and Traces

In this section, two properties of (component) interfaces and local component and link traces are discussed. The first property is properness of traces. The properness property ensures that for each state in a trace, an immediate successor state can be distinguished. Properness is presented in Section 6.1.1. The second property, input persistence, is a property of the input interface of a component or of a link. This property states that input information can only change as a result of information transmission. The input persistence property is defined in terms of an important notion, transmission octets, which is defined in Section 6.1.2 below. The input persistence property itself is discussed in Section 6.1.3.

6.1.1 Properness and Finite Variability

Some properties of compatibility relations refer to immediate successor states, or *next states*. Time frames as defined in Section 5.2.1 are not necessarily discrete. Therefore, a next state of a state $v_{A,i}$ is defined as the first state after $v_{A,i}$ that differs from $v_{A,i}$. Formally:

Definition 6.1. (Next and previous state). *Let LT=⟨TF;V⟩ be a local component or link trace with time frame TF=⟨T;<⟩.*

- *A* next state *of a state V(t) is a state V(t') such that t<t' and V(t')≠V(t) and for all t'' with t<t''<t' it holds that V(t)=V(t''). The set of all next states of a state V(t) is denoted* $next_{LT}(V(t))$.
- *A* previous state *of a state V(t) is a state V(t') such that t'<t and V(t')≠V(t) and for all t'' with t'<t''<t it holds that V(t)=V(t''). The set of all previous states of a state V(t) is denoted* $prev_{LT}(V(t))$.

Note that, for a discrete time frame, it is possible that there are two time points t and t' such that there is no t'' with $t<t''<t'$, while $V(t')=V(t)$. For a dense time frame, there are usually intervals of time for which the state remains constant.

Traces in which up to a specific point in time t, each state $V(t')$ with $t'<t$ has a next state and a previous state (unless $t'=\perp$), are called *proper traces*.

Definition 6.2. (Proper trace). *A local component or link trace* $LT=\langle TF;V\rangle$ *with time frame* $TF=\langle T;<\rangle$ *is a* proper trace *iff for all* $t\in T$, *V(t) has a next state and a previous state unless* $t=\perp$, *or there is a* $t'\in T$ *such that for all* $t\in T$ *with* $t'<t$, *V(t) has a next state and a previous state unless* $t=\perp$, *and for all* $t''\geq t'$, $V(t'')=V(t')$.

It is not the case that for a proper trace, the order of the time frame is necessarily discrete. However, the order of the *states* induced by the next state relation, i.e. the order $\{\langle v_{S,i};v_{S,j}\rangle \mid v_{s,j}\in next_{LT}(v_{S,i})\}$, is discrete.

For a proper trace with a dense order, another property can be defined: the *finite variability* or *non-Zenoness* property (Barringer, Kuiper & Pnueli, 1986). This property asserts that in a finite amount of time, only a finite number of next states can be distinguished. This property is not formally defined in this thesis. Instead, a trace that does not have the finite variability property is depicted (Figure 6.1).

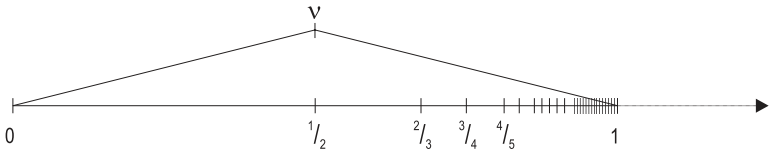

Figure 6.1: A trace that does not have the finite variability property.

In Figure 6.1, the horizontal line represents a dense set of time points of the time frame, i.e. the set of reals. The small vertical lines represent time points at which there is a state change. As can be seen from the figure, on the horizontal line, state changes occur at *t=1/2, t=2/3, 3/4, t=4/5,* and so on. There are an infinite number of state changes before *t=1*, but nevertheless, for every $t\in[0,1)$, the state at t has a next state. Thus, the trace depicted in the figure is a trace that does not have the finite variability property. (The trace depicted in Figure 6.1 is a proper trace, because all states except the state at *t=0* have a previous state. A previous state of the state at *t=1* is state *v*.)

The semantic structure presented in this thesis is not committed to a specific choice with respect to properness and finite variability of traces. However, as indicated in Chapter 1, the semantic structure is based on the assumption that global time does not exist. This assumption is related to the properties presented in this section. First, two different conceptions of the notion of time are presented, which are called *observer time* and *implied time*:

- A local trace describes the behaviour of a component or link. A possible way to interpret such a trace is to view the trace as a record of observations made by a dedicated observer of the component or link. An observer has access to a device that generates a sequence of time points. (E.g., a wall clock, or a stopwatch. Whether these ticks are generated at regular intervals cannot be determined by the observer.) The observer is able to observe the state of the component or link. (As the observer is dedicated to the component or link it observes, the observer does not observe any other component or link.) At each time tick, the observer makes note of the state of the component or link, which results in a local component or link trace. Thus, this trace is based on *observer time*: the (notion of) time of an observer. The observer's time device can be assumed to generate a continuous sequence of time points (this is probably the normal conception of real-world time). This case naturally leads to a dense time frame for the local component trace, although this is not necessary. With observer time, whether dense or discrete, it is frequently the case that for two consecutive time points or an interval of time, the state of the component or link remains the same.
- A second way to interpret a local trace is to view the component or link itself as the source of the time frame of the trace. No observer and no external time device is assumed. Instead, each change of the (externally visible part) of the component or link defines a new point in time. In other words, (a notion of) time is *implied* by the activity of the component or link. If the behaviour of the component or link is continuous, the implied time is dense. If the behaviour of the component or link is discrete, the implied time is discrete. A consequence of this interpretation is that, in a discrete time frame, for any pair of time points t,t' such that there is no t'' with $t<t''<t'$, $V(t)\neq V(t')$. (If this were not the case, then t' would not be distinguished as a new point in time). In a dense time frame, a similar property holds: for any two points in time t and t', no matter how close to one another, there is always a time point t'' such that $t<t''<t'$, $V(t'')\neq V(t)$ and $V(t'')\neq V(t')$.

The semantic structure presented in this thesis does not assume that local traces are interpreted as observer time or as implied time. However, as the semantic structure is based on the assumption that global time does not exist, neither the observer time interpretation nor the implied time interpretation for a specific component or link can be used as a reference time for another component or link. As an example, if the observer time interpretation is adopted, then it is assumed

that no two components or link share an observer or its time device. (This is the result of the assumption that observers are dedicated.)

6.1.2 Transmission Octets

Information transmission establishes a relation between two components and a link, or between two links and a component. More specifically, each occurrence of an information transmission establishes a relation between states of the domain and co-domain of a specific link: a state of the domain contains information, which is transmitted to the co-domain, in which a new state results from this transmission. This correspondence is the basis of interaction.

In fact, in the semantic structure presented in this thesis, the relation can be extended to a correspondence between eight states, as explained in this section. Tuples of eight states that correspond as a result of information transmission are called *transmission octets*. The relation between the eight states of a transmission octet is related to an information link mapping: tuples composed of eight states that occur in local component and link traces of a link, its domain and co-domain, form a transmission octet if they comply with the information link mapping of the link. This relation is formally defined in this section.

A transmission octet for a link I is an octet consisting of two states from a local component or link trace of the domain of I, four states from a local link trace of I and two states from a local component or link trace of the co-domain of I. A transmission octet thus consists of states taken from the behaviour of a link, its domain and its co-domain as represented by local component and link traces. The connection with the intended information transmission functionality of the link is made as follows: an octet of states is a transmission octet if the states in the octet equal an element of the information link mapping of I and have a specific order in the local traces from which they are taken. (The information link mapping of a link consists of substates taken from the set of all possible states of a link, its domain and co-domain—the sets $\mathcal{S}_I$, $\mathcal{S}_{dom(I)}$ and $\mathcal{S}_{cdom(I)}$, respectively.

The definition of a transmission octet only enforces that the two states of the domain have a specific order in the local trace of the domain, and likewise for the four states of the link itself and the two states of the co-domain. The definition of a transmission octet does not enforce any other relation between the two states of the domain of the link, or between the two states of the co-domain of the link, or between the four states of the link itself. Thus, almost any two states of the domain of a link, together with four states of the link itself and two states of the co-domain, can potentially form a transmission octet, regardless of the number of states in-between the states in the domain, link or co-domain,. The definitions of properties of compatibility relations in Section 6.2, however, do enforce relations between the two states of the domain of a link, between the four states of the link, and between the two states of the co-domain. E.g., a specific property holds for a compatibility relation for a link if a state of the domain of the link and its immediate successor

state, together with four states of the link in a specific order and a state of the co-domain with its immediate successor, form a transmission octet.

The notion of a transmission octet is close to the notion of an information link mapping. The difference between, on the one hand, a transmission octet, and, on the other hand, an information link mapping is as follows. An information link mapping consists, as indicated by Definition 5.6, of octets of *substates* taken from the sets of possible states of a link, its domain and co-domain. (The exact choice of substates varies for the six types of information links distinguished in the semantic structure.) However, local component and link traces consist of states. For a specific property of a compatibility relation, only specific substates of these states are relevant, depending on the type of the link with which the compatibility relation is associated (see below). Contrary to an information link mapping, a transmission octet consists of *states*. A transmission octets matches *states* from local component or link traces with (input and output) *substates* from an information link mapping.

As stated before, properties of compatibility relations are expressed in terms of transmission octets. The relationship between compatibility relations and transmission octets can be illustrated as follows. Loosely speaking, properties of compatibility relations (which are relations defined on *traces*), require that specific combinations of eight states—two from a local trace of the domain of a link, four from a local trace of the link itself and two of a local trace of the co-domain—form transmission octets. Again loosely speaking, if for all states in a trace of the domain or co-domain of a link, such combinations can be found, then a compatibility relation for that link satisfies a specific property.

To summarise, information link mappings, transmission octets and compatibility relations are compared as follows. An information link mapping relates *eight substates* from the *sets of all states* of two components and a link or a component and two links (Thus, an information link mapping does not refer to traces). A transmission octet relates *eight states* taken from *traces* of two components and a link or one component and two links, such that these eight states constitute an instance of transmission as described by an information link mapping. A *property* of a compatibility relation requires, loosely speaking, that specific octets of states taken from triples of traces in the compatibility relation form transmission octets. (Different properties require different specific octets to form transmission octets.)

For ease of comprehension, the definition of a transmission octet below deliberately uses the names of the link, components and states depicted in Figure 2.7 as the names of the variables in the definition. In the definition, the following notation is used: let $LT_S=\langle T;V\rangle\in\mathcal{LT}_S$ be a local component or link trace. The state $V(t)$ in LT_S at time point $t\in T$ is denoted $v_{S,t}$ or sometimes $LT_S(t)$.

Definition 6.3. (Transmission octet). *Let LT_A, LT_B and LT_L be three local traces of components A and B and an information link L with information link mapping λ_L, such*

that A=dom(L) and B=cdom(L). Let $v_{A,i}$ *and* $v_{A,j}$ *be two states in* LT_A*, let* $v_{L,i''}$*,* $v_{L,j''}$*,* $v_{L,k}$ *and* $v_{L,l}$ *be four states in* LT_L *and let* $v_{B,i'}$ *and* $v_{B,j'}$ *be two states in* LT_B*. The octet* $\langle\langle v_{A,i}; v_{A,j}\rangle;\langle v_{L,i''}; v_{L,j''}; v_{L,k}; v_{L,l}\rangle;\langle v_{B,i'}; v_{B,j'}\rangle\rangle$ *is a* transmission octet *with respect to* LT_A*,* LT_L *and* LT_B *iff* $i<_A j$*,* $i''<_L j''<_L k<_L l$*,* $i'<_B j'$ *and:*

- *L is a private link and* $\langle\langle out(v_{A,i}); out(v_{A,j})\rangle;\langle v_{L,i''}; v_{L,j''}; v_{L,k}; v_{L,l}\rangle;\langle in(v_{B,i'}); in(v_{B,j'})\rangle\rangle \in \lambda_L$*, or*
- *L is an import mediating link and* $\langle\langle in(v_{A,i}); in(v_{A,j})\rangle;\langle v_{L,i''}; v_{L,j''}; v_{L,k}; v_{L,l}\rangle;\langle in(v_{B,i'}); in(v_{B,j'})\rangle\rangle \in \lambda_L$*, or*
- *L is an export mediating link and* $\langle\langle out(v_{A,i}); out(v_{A,j})\rangle;\langle v_{L,i''}; v_{L,j''}; v_{L,k}; v_{L,l}\rangle;\langle out(v_{B,i'}); out(v_{B,j'})\rangle\rangle \in \lambda_L$*, or*
- *L is a cross-mediating link and* $\langle\langle in(v_{A,i}); in(v_{A,j})\rangle;\langle v_{L,i''}; v_{L,j''}; v_{L,k}; v_{L,l}\rangle;\langle out(v_{B,i'}); out(v_{B,j'})\rangle\rangle \in \lambda_L$*, or*
- *L is a link modifier link and* $\langle\langle out(v_{A,i}); out(v_{A,j})\rangle;\langle v_{L,i''}; v_{L,j''}; v_{L,k}; v_{L,l}\rangle;\langle v_{B,i'}; v_{B,j'}\rangle\rangle \in \lambda_L$*, or*
- *L is a link monitoring link and* $\langle\langle v_{A,i}; v_{A,j}\rangle;\langle v_{L,i''}; v_{L,j''}; v_{L,k}; v_{L,l}\rangle;\langle in(v_{B,i'}); in(v_{B,j'})\rangle\rangle \in \lambda_L$*.*

Example 6.4. In Example 5.19 a compatibility relation for the traces $lt_{\text{broker},1}$, $lt_{\text{broker\_to\_user\_1}}$ and $lt_{\text{user\_1},1}$ is presented. The current example presents two transmission octets for these traces, to provide a more detailed view on the compatibility relation. The transmission octets presented in the current example are used in another example in Section 6.2.1 to illustrate the definition of a property of a compatibility relation in terms of transmission octets. For ease of reference, the information link mapping for broker_to_user_1, given in Example 5.7, is repeated here:

$\lambda_{\text{broker_to_user_1}} = \{\langle\langle$to_be_communicated_to($t$,user_1)$;$just_communicated_to($t$,user_1)$\rangle;$
$\langle$awake_and_empty;active_and_contents(t);active_and_contents(t);
awake_and_empty$\rangle;\langle$ready_for_information;
communicated_by(t',broker)$\rangle\rangle \mid t \in OT_2, t' \in OT_1$ *and* $t'=trans(t)\}$.

In this example, assume that *trans*(res_1)=res_3. First, consider the following octet of states, taken from the traces $lt_{\text{broker},1}$ and $lt_{\text{user\_1},1}$ given in Example 5.14, and $lt_{\text{broker\_to\_user\_1}}$, given in Example 5.17:

⟨⟨∅ | belief(match(res_1,query_1)) | to_be_communicated_to(res_1,user_1); *(4th state in* $lt_{\text{broker},1}$*)*
∅ | belief(match(res_1,query_1)) | just_communicated_to(res_1,user_1)⟩; *(5th state in* $lt_{\text{broker},1}$*)*
⟨awake_and_empty; *(1st state in* $lt_{\text{broker\_to\_user\_1}}$*)*
active_and_contents(res_1); *(2nd state in* $lt_{\text{broker\_to\_user\_1}}$*)*
active_and_contents(res_1); *(2nd state in* $lt_{\text{broker\_to\_user\_1}}$*)*
awake_and_empty⟩; *(3rd state in* $lt_{\text{broker\_to\_user\_1}}$*)*

⟨ready_for_information | ∅ | ∅; (2nd state in $lt_{user\_1,1}$)
communicated_by(res_3,broker) | ∅ | ∅⟩ (3rd state in $lt_{user\_1,1}$)

The first element in this octet is referred to as $v_{broker\_1,4}$, the second as $v_{broker\_1,5}$ the third to the sixth as $v_{broker\_to\_user\_1,i}$ for $i=1,\ldots4$, respectively, the seventh as $v_{user\_1,2}$ and the eight as $v_{user\_1,3}$. This octet is a transmission octet for the following reason: broker_to_user_1 is a private link, $out(v_{broker\_1,4})$= to_be_communicated_to(res_1,user_1), $out(v_{broker\_1,5})$=just_communicated_to(res_1,user_1), $in(v_{user\_1,2})$=ready_for_information, and $in(v_{user\_1,3})$=communicated_by(res_3,broker). Thus:

$$
\begin{aligned}
&\langle\langle out(v_{broker\_1,4});out(v_{broker\_1,5})\rangle;\\
&\quad\langle v_{broker\_to\_user\_1,1}; v_{broker\_to\_user\_1,2}; v_{broker\_to\_user\_1,2}; v_{broker\_to\_user\_1,3}\rangle;\\
&\quad\langle in(v_{user\_1,2});in(v_{user\_1,3})\rangle\rangle\\
=\ &\langle\langle \text{to\_be\_communicated\_to(res\_1,user\_1)};\text{just\_communicated\_to(res\_1,user\_1)}\rangle;\\
&\quad\langle \text{awake\_and\_empty; active\_and\_contents(res\_1);active\_and\_contents(res\_1)}\\
&\quad\ \text{awake\_and\_empty}\rangle;\langle \text{ready\_for\_information;}\\
&\quad\ \text{communicated\_by(res\_3,broker)}\rangle\rangle\\
\in\ &\lambda_{broker\_to\_user\_1}
\end{aligned}
$$

This is exactly as required for a private link by Definition 6.3.

Second, consider an octet of states similar to the octet given in the first part of this example, but with the first element replaced by ∅|belief(match(res_1,query_1))|∅, named $v_{broker\_1,1}$, which is the first state in $lt_{broker\_1,1}$. The octet with $v_{broker\_1,1}$ is *not* a transmission octet for broker_to_user_1, for the following reason: $out(v_{broker\_1,1})=\varnothing$[7], thus $\langle\langle out(v_{broker\_1,4});out(v_{broker\_1,5})\rangle;\langle v_{broker\_to\_user\_1,1}; v_{broker\_to\_user\_1,2}; v_{broker\_to\_user\_1,2}; v_{broker\_to\_user\_1,3}\rangle;\langle in(v_{user\_1,2});in(v_{user\_1,3})\rangle\rangle \notin \lambda_{broker\_to\_user\_1}$, which is contrary to the requirement for a private link as given by Definition 6.3. ■

6.1.3 Input Persistence

The first property of a component defined in this section, the input persistence property, specifies that information that is input to the component cannot change spontaneously:

Definition 6.5. (Input persistence property). *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy, let μ be a multitrace for SH and let $C\in Comp$ be a component with $\mu_C=\langle\langle T_C;<_C\rangle;V_C\rangle$. Component C has the* input persistence property *if for each $j'\in T_C\setminus\{\perp\}$, either:*

- *There is an $I\in Lnk$ with $cdom(I)=C$, $\mu_I=\langle\langle T_I;<_I\rangle;V_I\rangle$ and $\mu_{dom(I)}=\langle\langle T_{dom(I)};<_{dom(I)}\rangle;V_{dom(I)}\rangle$ such that there exist $i'\in T_C$, $i''<_I j''<_I k<_I l\in T_I$ and $i<_{dom(I)} j\in T_{dom(I)}$ such that $\langle\langle v_{dom(I),i};v_{dom(I),j}\rangle;\langle v_{I,i''};v_{I,j''};v_{I,k};v_{I,l}\rangle;\langle v_{C,i'};v_{C,j'}\rangle\rangle$ is a transmission octet for $LT_{dom(I)}$, LT_I and LT_C,*

[7] ∅ is the name of a state (see Example 5.2), not the empty set.

- *Or, there is an* $i' \in T_C$ *with* $i' <_C j'$ *such that for all* $m \in T_C$ *with* $i' <_C m \leq_C j'$, $in(v_{C,m}) = in(v_{C,j'})$.

This property specifies that for each state $v_{S,j'}$, one of the following requirements holds:

- There is another component or link in *SH* that provides new input to *C*. Formally, it is required that there is a link in *SH* with *C* as co-domain, such that a state $v_{C,i'}$ exists that is a predecessor (not necessarily immediate) of $v_{C,j'}$, four states $v_{I,i''}$, $v_{I,j''}$, $v_{I,k}$, and $v_{I,l}$ and two states $v_{dom(I),i}$ and $v_{dom(I),j}$ such that all eight states together form a transmission octet. This implies that the input substate of state $v_{C,j'}$ contains information obtained from the domain of link *I*, where it was available in state $v_{dom(I),i}$. Moreover, the transmission is carried out in conformance with the information link mapping. Two enabling conditions (represented by the states $v_{I,i''}$ and $v_{C,i'}$, respectively) are also fulfilled.
- Or, the input substate of state $v_{C,j'}$ has not changed, i.e., a time point i' can be found such that for all time points *m* between i' and j', the input substate of *m* equals the input substate of $v_{C,j'}$.

This property models input persistence in the sense that if the input substate changes, then the change is caused by information transmission. Otherwise, the input substate is persistent: it is the same as in the previous state.

6.2 Properties of Compatibility Relations

Section 5.2.2 defined a compatibility relation for a link *I* as a ternary relation on the set of local component or link traces of the domain of *I*, the set of link traces of *I* and the set of local component or link traces of the co-domain of *I*. The formal definition is repeated here for ease of reference:

Definition 5.18. (Compatibility relation). *A* compatibility relation *for a link I is a relation* $cR_I \subseteq \mathcal{LT}_{dom(I)} \times \mathcal{LT}_I \times \mathcal{LT}_{cdom(I)}$.

As stated in Section 5.2.2, compatibility relations are used to model constraints imposed by information transmission. However, Section 5.2.2 does not define how compatibility relations model different properties of information transmission, such as reliability (whether information can be lost). This chapter defines such properties of compatibility relations. Applications of the semantic structure may or may not choose to adopt these properties.

As stated earlier, a compatibility relation for a link relates states that occur in the actual behaviour of the link, its domain and co-domain, in conformance with the information link mapping given for the link. In fact, the connection between, on the one hand, an information link mapping and, on the other hand, states that

actually occur in local component and link traces is formalised by the notion of *transmission octets*, introduced in Section 6.1.1. The various properties of information transmission are defined in this section in terms of transmission octets. In sections 6.2.1 to 6.2.4, four properties are presented.

6.2.1 Lossless Transmission Property

The first possible property of a compatibility relation defined in this section, the lossless transmission property, specifies that no data is lost during information transmission. Formally:

Definition 6.6. (Lossless transmission property). *Let cR_I be a compatibility relation for a link I. For this compatibility relation the* lossless transmission property *holds iff for each $\langle LT_{dom(I)};LT_I;LT_{cdom(I)}\rangle \in cR_I$: with $LT_I=\langle\langle T_I;<_I\rangle;V_I\rangle$, $LT_{dom(I)}=\langle\langle T_{dom(I)};<_{dom(I)}\rangle;V_{dom(I)}\rangle$, and $LT_{cdom(I)}=\langle\langle T_{cdom(I)};<_{cdom(I)}\rangle;V_{cdom(I)}\rangle$: for all $i \in T_{dom(I)}$ it holds that either:*

- *There exist $j \in T_{dom(I)}$, $i'' <_I j'' <_I k <_I l \in T_I$ and $i' <_{cdom(I)} j' \in T_{cdom(I)}$ such that $\langle\langle v_{dom(I),i}; v_{dom(I),j}\rangle;\langle v_{I,i''}; v_{I,j''}; v_{I,k}; v_{I,l}\rangle;\langle v_{cdom(I),i'}; v_{cdom(I),j'}\rangle\rangle$ is a transmission octet,*
- *Or, one of the following holds:*
  - *There is no $\langle\langle out(v_{dom(I),i});\sigma_2\rangle;\langle\sigma_3;\sigma_4;\sigma_5;\sigma_6\rangle;\langle\sigma_7;\sigma_8\rangle\rangle \in \lambda_I$ for any $\sigma_2,\ldots,\sigma_8$, if I is a private link, an export mediating link or a link modifier link, and*
  - *There is no $\langle\langle in(v_{dom(I),i});\sigma_2\rangle;\langle\sigma_3;\sigma_4;\sigma_5;\sigma_6\rangle;\langle\sigma_7;\sigma_8\rangle\rangle \in \lambda_I$ for any $\sigma_2,\ldots,\sigma_8$, if I is an import mediating link or a cross-mediating link, and*
  - *There is no $\langle\langle v_{dom(I),i};\sigma_2\rangle;\langle\sigma_3;\sigma_4;\sigma_5;\sigma_6\rangle;\langle\sigma_7;\sigma_8\rangle\rangle \in \lambda_I$ for any $\sigma_2,\ldots,\sigma_8$, if I is a link monitoring link.*

This property specifies that for each state $v_{dom(I),i}$, one of the following requirements must hold:

- There exists a state $v_{dom(I),j}$ that is a successor (not necessarily immediate) of $v_{dom(I),i}$, four states $v_{I,i''}$, $v_{I,j''}$, $v_{I,k}$, and $v_{I,l}$, two states $v_{cdom(I),i'}$ and $v_{cdom(I),j'}$ such that all eight states together form a transmission octet. For a private link, this implies that for each state $v_{dom(I),i}$, if the output substate of $v_{dom(I),i}$ should, according to the information link mapping of *I*, be transmitted, and two enabling conditions (represented by the states $v_{I,i''}$ and $v_{cdom(I),i'}$, respectively) are fulfilled, then there should be a state $v_{cdom(I),j'}$ in which the effect of the transmission as specified by the information link mapping is present.
- Or, state $v_{dom(I),j}$ is not applicable for transmission via link *I*, which is indicated by the absence of an octet of states in the information link mapping of *I* in which (a substate of) $v_{dom(I),j}$ is the first element.

This property models lossless transmission in the sense that if, according to the information link mapping, transmission is required, then there is a state in the

trace of the co-domain in which the effect of the transmission is present. Thus, the information is not lost in transit.

From the definition of the lossless transmission property, a number of interesting characteristics of compatibility relations can be observed:

- Like an information link mapping, a compatibility relation does not refer to the actual behaviour of components: a compatibility relation only refers to arbitrary local component and link traces. Instead, a compatibility relation in a sense lifts intended information transmission as defined by information link mappings, from possible states to states that occur in local component and link traces. These local component and link traces are possible behaviours and not necessarily actual traces of a component. However, as explained in Section 5.2.2, compatibility relations are used to define actual behaviour of composed components. In this case, compatibility relations impose a structure on multitraces consisting of actual component behaviour.
- The lossless transmission property does *not* specify that the *moment in time* at which state $v_{dom(I),i}$ in $LT_{dom(I)}$ or state $v_{I,k}$ in LT_I occurs must precede the moment in time at which state $v_{cdom(I),j}$ in $LT_{cdom(I)}$ occurs. However, it *does* specify that the input substate of the co-domain of I depends on output substate of the domain of I and the state of I. With the definition presented, it is not even possible to express that moments in time for the domain of I have any relation with moments in time of the co-domain of I, because no relationship between time in the domain and co-domain (represented by their traces) is given. In the next chapter, the relationship between dependence and global temporal precedence is explored in more detail.
- The lossless transmission property is related to the commitments put forward in Chapter 2. In particular, the commitment to non-blocking send and receive operations presented in Section 2.2.5 and Section 2.2.6, respectively, are represented in the definition of the lossless transmission property. These commitments are represented by the (relatively weak) requirements that state $v_{dom(I),j}$ is a successor, but not an immediate successor, of $v_{dom(I),i}$ (non-blocking send) and that state $v_{cdom(I),j'}$ is a successor, but not an immediate successor, of $v_{cdom(I),i'}$ (non-blocking receive).

Example 6.7. Example 5.20 in Chapter 5 presents a compatibility relation for the link broker_to_user_1. Example 5.20 claims that one of the elements of this compatibility relation is the triple $\langle lt_{\text{broker},1};lt_{\text{broker\_to\_user\_1}};lt_{\text{user\_1},1}\rangle$, with the local component traces $lt_{\text{broker},1}$ and $lt_{\text{user\_1},1}$ as presented in Example 5.14 and the local link trace $lt_{\text{broker\_to\_user\_1}}$ as presented in Example 5.17. Example 6.4 presented two transmission octets for these traces.

The current example shows that the triple $\langle lt_{\text{broker},1};lt_{\text{broker\_to\_user\_1}};lt_{\text{user\_1},1}\rangle$ satisfies the lossless transmission property. Thus, for each state $v_{\text{broker},i}$ it has to be shown

that either there are states in $lt_{\text{broker\_to\_user\_1}}$ and $lt_{\text{user\_1},1}$ such that these states form a transmission octet, or $v_{\text{broker},i}$ is not applicable to transmission via the (private) link broker_to_user_1:

- $v_{\text{broker},i}$ for $i=1,2,3$: $out(v_{\text{broker},i})=\emptyset$, and there is no element $\langle(\emptyset;\sigma_2);(\sigma_3;\sigma_4;\sigma_5;\sigma_6);(\sigma_7;\sigma_8)\rangle \in \lambda_{\text{broker\_to\_user\_1}}$ for any $\sigma_2,\ldots,\sigma_8$, so the second clause of Definition 6.6 holds.
- $v_{\text{broker},4}$: According to Example 6.4, the following octet is a transmission octet for the traces $lt_{\text{broker},1}$, $lt_{\text{broker\_to\_user\_1}}$ and $lt_{\text{user\_1},1}$:

  $\langle(\emptyset$ | belief(match(res_1,query_1)) | to_be_communicated_to(res_1,user_1);
  $\emptyset$ | belief(match(res_1,query_1)) | just_communicated_to(res_1,user_1)$)$;
  $\langle$awake_and_empty;
  active_and_contents(res_1);
  active_and_contents(res_1);
  awake_and_empty$)$;
  $\langle$ready_for_information | $\emptyset$ | $\emptyset$;
  communicated_by(res_3,broker) | $\emptyset$ | $\emptyset\rangle\rangle$
  $=\langle(v_{\text{broker},4};v_{\text{broker},5})$;
  $\langle v_{\text{broker\_to\_user\_1},1};v_{\text{broker\_to\_user\_1},2};v_{\text{broker\_to\_user\_1},3})$;
  $\langle v_{\text{user\_1},2};v_{\text{user\_1},2}\rangle\rangle$.

  Thus, there exist a j in T_{broker}, an i'', j'', k and l in $T_{\text{broker\_to\_user\_1}}$, and an i' and j' in $T_{\text{user\_1}}$ such that $\langle(v_{dom(I),i};v_{dom(I),j});\langle v_{I,i''};v_{I,j''};v_{I,k};v_{I,l}\rangle;\langle v_{cdom(I),i'};v_{cdom(I),j'}\rangle\rangle$ is a transmission octet (take $j=5$, $i''=1$, $j''=2$, $k=2$, $l=3$, $i'=2$ and $j'=3$). It also holds that $v_{I,2}\in next_{LT_I}(v_{I,1})$ and $v_{I,3}\in next_{LT_I}(v_{I,3})$, for I=broker_to_user_1. Thus, for $v_{\text{broker},4}$, the first clause of Definition 6.6 holds.
- $v_{\text{broker},5}$: $out(v_{\text{broker},i})$=just_communicated_to(res_1,user_1), and there is no element $\langle(\text{just\_communicated\_to(res\_1,user\_1)};\sigma_2);(\sigma_3;\sigma_4;\sigma_5;\sigma_6);(\sigma_7;\sigma_8)\rangle \in \lambda_{\text{broker\_to\_user\_1}}$ for any $\sigma_2,\ldots,\sigma_8$, so the second clause of Definition 6.6 holds.

Thus, for each state in $lt_{\text{broker},1}$, the requirements of Definition 6.6 hold. ■

Example 6.8. The lossless transmission property only requires that for each state in the domain of a link that contains information that has to be transmitted, there is a state in the co-domain in which this information is present. It does not require that this state is unique. This is illustrated as follows. Suppose that there is a trace $lt_{\text{broker},3}$, which is a copy of $lt_{\text{broker},1}$ extended with two new states, $v_{\text{broker},6}$ and $v_{\text{broker},7}$, with $v_{\text{broker},6}=v_{\text{broker},4}$ and $v_{\text{broker},7}=v_{\text{broker},5}$. The triple $\langle lt_{\text{broker},3};lt_{\text{broker\_to\_user\_1}};lt_{\text{user\_1},1}\rangle$ also satisfies the lossless transmission property: for each state $v_{\text{broker},i}$, either there are states in $lt_{\text{broker\_to\_user\_1}}$ and $lt_{\text{user\_1},1}$ such that these states form a transmission octet, or $v_{\text{broker},i}$ is not applicable to transmission via the (private) link broker_to_user_1:

- $v_{broker,i}$ for $i=1,\ldots,5$: same reason as in the previous example.
- $v_{broker,6}$: According to Example 6.4, the following octet is a transmission octet for the traces $lt_{broker,3}$, $lt_{broker\_to\_user\_1}$ and $lt_{user\_1,1}$:

$$\begin{aligned}
&\langle\{(\varnothing \mid \text{belief(match(res_1,query_1))} \mid \text{to_be_communicated_to(res_1,user_1)};\\
&\quad \varnothing \mid \text{belief(match(res_1,query_1))} \mid \text{just_communicated_to(res_1,user_1)})\rangle;\\
&\langle \text{awake_and_empty};\\
&\ \text{active_and_contents(res_1)};\\
&\ \text{active_and_contents(res_1)};\\
&\ \text{awake_and_empty}\rangle;\\
&\langle \text{ready_for_information} \mid \varnothing \mid \varnothing;\\
&\ \text{communicated_by(res_3,broker)} \mid \varnothing \mid \varnothing\rangle\rangle\\
=&\langle\langle v_{broker,6}; v_{broker,7}\rangle;\\
&\quad \langle v_{broker_to_user_1,1}; v_{broker_to_user_1,2}; v_{broker_to_user_1,2}; v_{broker_to_user_1,3}\rangle;\\
&\quad \langle v_{user_1,2}; v_{user_1,2}\rangle\rangle.
\end{aligned}$$

  Thus, there exist a j in T_{broker}, an i'', j'', k and l in $T_{broker\_to\_user\_1}$, and an i' and j' in T_{user_1} such that $\langle\langle v_{dom(I),i}; v_{dom(I),j}\rangle;\langle v_{I,i''}; v_{I,j''}; v_{I,k}; v_{I,l}\rangle;\langle v_{cdom(I),i'}; v_{cdom(I),j'}\rangle\rangle$ is a transmission octet (take $j=7$, $i''=1$, $j''=2$, $k=2$, $l=3$, $i'=2$ and $j'=3$). It also holds that $v_{I,2} \in next_{LT_I}(v_{I,1})$ and $v_{I,3} \in next_{LT_I}(v_{I,3})$, for I=broker_to_user_1. Thus, for $v_{broker,4}$, the first clause of Definition 6.6 holds.

- $v_{broker,7}$: $out(v_{broker,i})$=just_communicated_to(res_1,user_1), and there is no element $\langle\{\text{just\_communicated\_to(res\_1,user\_1)}; \sigma_2\rangle;\langle\sigma_3;\sigma_4;\sigma_5;\sigma_6\rangle;\langle\sigma_7;\sigma_8\rangle\rangle \in \lambda_{broker\_to\_user\_1}$ for any $\sigma_2,\ldots,\sigma_8$, so the second clause of Definition 6.6 holds.

Thus, for each state in $lt_{broker,1}$, the requirements of Definition 6.6 hold. This example shows that the same pair of states in the co-domain of a link may take part in more than one transmission octet. ■

6.2.2 *Order-Preserving Transmission Property*

The second property of a compatibility relation defined in this chapter, the order-preserving transmission property, requires that the results of two transmissions from a specific component to a specific other component by the same link occur in the same order as the initiations of the transmissions. As the property is a requirement on the occurrence of two transmissions, the formal definition is stated in terms of two transmission octets. Altogether, the formal definition refers to sixteen states.

Definition 6.9. (Order-preserving transmission property). *Let $c\mathcal{R}_I$ be a compatibility relation for a link I. For this compatibility relation the* order-preserving transmission property *holds iff for each $\langle LT_{dom(I)}; LT_I; LT_{cdom(I)}\rangle \in c\mathcal{R}_I$: with $LT_I=\langle\langle T_I; <_I\rangle; V_I\rangle$,*

$LT_{dom(I)}=\langle\langle T_{dom(I)};<_{dom(I)}\rangle;V_{dom(I)}\rangle$, *and* $LT_{cdom(I)}=\langle\langle T_{cdom(I)};<_{cdom(I)}\rangle;V_{cdom(I)}\rangle$: *for all* $i,j,m,n \in T_{dom(I)}$, $i'',j'',k,l \in T_I$, $m'',n'',o,p \in T_I$, $i',j',m',n' \in T_{cdom(I)}$:

- *if*
  - $\langle\langle v_{dom(I),i};v_{dom(I),j}\rangle;\langle v_{I,i''};v_{I,j''};v_{I,k};v_{I,l}\rangle;\langle v_{cdom(I),i'};v_{cdom(I),j'}\rangle\rangle$ *is a transmission octet,*
  - *and* $\langle\langle v_{dom(I),m};v_{dom(I),n}\rangle;\langle v_{I,m''};v_{I,n''};v_{I,o};v_{I,p}\rangle;\langle v_{cdom(I),m'};v_{cdom(I),n'}\rangle\rangle$ *is a transmission octet,*
  - *and* $i<_{dom(I)}m$,
- *then* $j'<_{cdom(I)}n'$.

The definition states that, for the order-preserving transmission property to hold, for each state $v_{A,i}$ in a trace of the domain of the link, if this state is involved in a transmission that results in a state $v_{B,j'}$, and there is a later state $v_{A,m}$ that is involved in another transmission that results in $v_{B,n'}$, then $v_{B,n'}$ must occur later than $v_{B,j'}$. To illustrate the definition of the order-preserving transmission property, a triple of traces for which the property does *not* hold is depicted. The negation of the property is as follows: there is a triple $\langle LT_{dom(I)};LT_I;LT_{cdom(I)}\rangle \in CR_I$ with $LT_I=\langle\langle T_I;<_I\rangle;V_I\rangle$, $LT_{dom(I)}=\langle\langle T_{dom(I)};<_{dom(I)}\rangle;V_{dom(I)}\rangle$, and $LT_{cdom(I)}=\langle\langle T_{cdom(I)};<_{cdom(I)}\rangle;V_{cdom(I)}\rangle$ for which there exist $i,j,m,n \in T_{dom(I)}$, $i'',j'',k,l \in T_I$, $m'',n'',o,p \in T_I$, $i',j',m',n' \in T_{cdom(I)}$ such that:

- $\langle\langle v_{dom(I),i};v_{dom(I),j}\rangle;\langle v_{I,i''};v_{I,j''};v_{I,k};v_{I,l}\rangle;\langle v_{cdom(I),i'};v_{cdom(I),j'}\rangle\rangle$ is a transmission octet,
- and $\langle\langle v_{dom(I),m};v_{dom(I),n}\rangle;\langle v_{I,m''};v_{I,n''};v_{I,o};v_{I,p}\rangle;\langle v_{cdom(I),m'};v_{cdom(I),n'}\rangle\rangle$ is a transmission octet,
- and $i<_{dom(I)}m$,
- and $n'<_{cdom(I)}j'$.

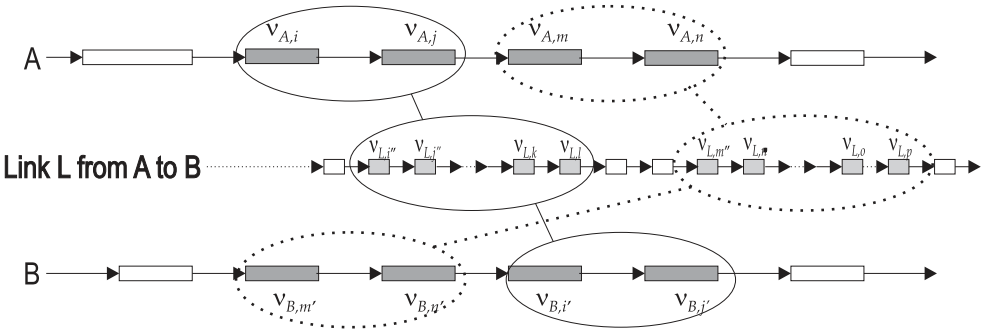

Figure 6.2: Traces that do not satisfy the order-preserving transmission property.

This situation is depicted in Figure 6.2. The two transmission octets are indicated by solid and dashed ovals. (The lines connecting the ovals indicate which states together form a transmission octet.) It is clear that the traces depicted in Figure 6.2 do not satisfy the order-preserving transmission property.

6.2.3 Asynchronous Transmission Property

The third property presented in this chapter is the asynchronous transmission property, which coincides with the commitment to asynchronous transmission presented in Section 2.2.7. This property is defined in terms of another notion, *dependence*, which is itself defined in terms of transmission octets. Dependence is a binary relation on the union of the sets of states that occur in the traces of a link, its domain and co-domain. As it is not assumed that all $\mathcal{S}_S$ are pairwise disjoint, the relation is defined using coloured versions of the states in each $\mathcal{S}_S$. This makes it possible to define notions that uniquely reference states of specific components. Moreover, it is not assumed that for a specific component or link, states in a local component or link trace are unique. Therefore, for reasons explained shortly, states are also coloured with the point in time at which they occur.

Definition 6.10. (Disjoint union of states). *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy and let μ be a multitrace for SH. The* disjoint union *of all local component and link states of the components and links in SH is the set $\mathcal{S}_{SH}=\{\langle v_S;S;i\rangle \mid S\in Comp\cup Lnk,\ v_S\in\mathcal{S}_S \text{ and } \mu_S(i)=v_S\}$.*

For notational convenience, in the rest of this thesis, coloured versions $\langle v_S;S;i\rangle$ are identified with the uncoloured versions, i.e., a coloured state $\langle v;S;i\rangle$ is denoted $v_{S,i}$..

A state $v_{B,j'}$ depends on a state $v_{A,i}$ if either $A=B$ and $i<j'$ for A's local time ordering, or if there is a state $v_{C,m}$ such that $v_{B,j'}$ depends on $v_{C,m}$, and $v_{C,m}$ depends on $v_{A,i}$, or if $A=dom(L)$ for a link L and $B=cdom(L)$ and $v_{B,j'}$ contains the result of information transmission from $v_{A,i}$. The latter requirement can be expressed formally in terms of transmission octets, as shown in the following definition.

Definition 6.11. (Dependence). *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy and let $LT_A=\langle\langle T_A;<_A\rangle;V_A\rangle$, $LT_B=\langle\langle T_B;<_B\rangle;V_B\rangle$ and $LT_L=\langle\langle T_L;<_L\rangle;V_L\rangle$ be three traces of components or links $A,B\in Comp\cup Lnk$ and a link $L\in Lnk$ such that $A=dom(L)$ and $B=cdom(L)$. Let $v_{A,i}$ and $v_{A,j}$ be two states in LT_A, let $v_{L,i''}$, $v_{L,j''}$, $v_{L,k}$ and $v_{L,l}$ be four states in LT_L, and let $v_{B,i'}$ and $v_{B,j'}$ be two states in LT_B. The* dependence *relation $\rightarrow_d\subseteq\mathcal{S}_{SH}\times\mathcal{S}_{SH}$ for LT_A, LT_L and LT_B is defined as the smallest relation $\rightarrow_d$ such that $v_{A,i}\rightarrow_d v_{B,j'}$ iff either*

1. *$A=B$ and $i<_A j'$, or*
2. *$\langle\langle v_{A,i};v_{A,j}\rangle;\langle v_{L,i''};v_{L,j''};v_{L,k};v_{L,l}\rangle;\langle v_{B,i'};v_{B,j'}\rangle\rangle$ is a transmission octet for LT_A, LT_L and LT_B, or*

3. *There is a state $v_{C,m}$ of a component or link $C \in Comp \cup Lnk$ such that $v_{A,i} \rightarrow_d v_{C,m}$ and $v_{C,m} \rightarrow_d v_{B,j'}$.*

Definition 6.12. (Asynchronous transmission property). *Let $c\mathcal{R}_I$ be a compatibility relation for a link I. For this compatibility relation the* asynchronous transmission property *holds iff for each $\langle LT_{dom(I)}; LT_I; LT_{cdom(I)} \rangle \in c\mathcal{R}_I$, the dependence relation $\rightarrow_d$ for $LT_{dom(I)}$, LT_I and $LT_{cdom(I)}$ is a partial order.*

The dependence relation $\rightarrow_d$ is similar to Lamport's (1986) "happens before" relation that defines a temporal order without assuming the existence of global time. (However, Lamport's notion is defined in terms of events and does not support locality.) If $\rightarrow_d$ is a partial order, it cannot contain cycles, so there is no chain of states that all have to occur before their predecessor. This characterisation of asynchronous information transmission is also presented in (Charron-Bost, Mattern & Tel, 1996) in terms of events. Chapter 7 further discusses such partial orders.

Definition 6.12 requires that all states in a local component or link trace of a specific component or link are unique. If states are not unique, $\rightarrow_d$ is naturally symmetric and therefore not a partial order. This is the reason why states are coloured with the point in time at which they occur.

6.2.4 Logically Instantaneous Transmission Property

In Section 2.2.7, a commitment is made to asynchronous transmission. However, in some circumstances, synchronous information transmission is favoured over asynchronous information transmission. From the point of view of the initiator, synchronous information transmission appears to be instantaneous: there are no activities between the initiation of the information transmission and the receipt of the information by another component. Therefore, synchronous communication is more accurately referred to as the logically instantaneous transmission property.

Definition 6.13. (Logically instantaneous transmission property). *Let $SH=\langle Comp; Lnk; \prec; dom; cdom \rangle$ be a structure hierarchy and let LT_A, LT_B and LT_L be three traces of components or links $A,B \in Comp \cup Lnk$ and a link $L \in Lnk$ such that $A=dom(L)$ and $B=cdom(L)$. Let $v_{A,i}$ and $v_{A,j}$ be two states in LT_A, let $v_{L,i''}$, $v_{L,j''}$, $v_{L,k}$ and $v_{L,l}$ be four states in LT_L, and let $v_{B,i'}$ and $v_{B,j'}$ be two states in LT_B.*

- *The relation $K \subseteq \mathcal{S}_{SH} \times \mathcal{S}_{SH}$ for LT_A, LT_L and LT_B is defined as follows: $K=\{\langle v_{A,i}; v_{B,j'} \rangle \mid \langle\langle v_{A,i}; v_{A,j} \rangle; \langle v_{L,i''}; v_{L,j''}; v_{L,k}; v_{L,l} \rangle; \langle v_{B,i'}; v_{B,j'} \rangle\rangle$ is a transmission octet for LT_A, LT_L and $LT_B\}$;*
- *The relation R is defined as $R=\{\langle v_{S,i}; v_{S,j} \rangle \mid (S=A \text{ or } S=B) \text{ and } i<j\} \cup K$;*
- *Let K^{-1} be the set $\{\langle v_{B,j'}; v_{A,i} \rangle \mid \langle\langle v_{A,i}; v_{A,j} \rangle; \langle v_{L,i''}; v_{L,j''}; v_{L,k}; v_{L,l} \rangle; \langle v_{B,i'}; v_{B,j'} \rangle\rangle$ is a transmission octet for LT_A, LT_L and LT_B and $i<j$, $i'<j'\}$;*

- *Let $R'=(R\cup K^{-1})^*\setminus(K\cup K^{-1})$, where R^* denotes the transitive closure of R;*
- *Let $c\mathcal{R}_I$ be a compatibility relation for a link I. For this compatibility relation the* logically instantaneous transmission property *holds iff for each $\langle LT_{dom(I)};LT_I;LT_{cdom(I)}\rangle\in c\mathcal{R}_I$, the relation R' for $LT_{dom(I)}$, LT_I and $LT_{cdom(I)}$ is a partial order.*

The construction $R'=(R\cup K^{-1})^*\setminus(K\cup K^{-1})$ is adapted from a proof in (Charron-Bost *et al.*, 1996) in which an equivalent characterisation of logically instantaneous information transmission in terms of events is given. From the logically instantaneous transmission property, it can be proven that an arbitrary state ν of the domain *D* of a link or the co-domain *C* of a link happens before (in Lamport's sense) a state in which information is made available in *D* if and only if it happens before a state in which this information is received by *C*. Likewise, an arbitrary state ν of *D* or *C* happens after (in Lamport's sense) a state in which information is made available in *D* if and only if it happens after a state in which this information is received by *C*. Thus, at the moment of logically instantaneous information transmission, components *C* and *D* have the same past and future. (See (Charron-Bost *et al.*, 1996) for the proof.)

6.3 Discussion

The reader may be surprised to find that non-local phenomena and properties of information transmission can be defined without any relation to the clocks of the components involved. The following observations may further clarify this issue. In the first place, the definition of compatibility given above amounts to a theory of dependence and nothing more. In particular, the three views on the behaviour of a component abstract from (i) the actual time spent by components between two states according to some observer and (ii) synchronisation characteristics and buffering of information transmission between components. In the second place, a non-local view in terms of dependence is, on the one hand, sufficient for many purposes, while, on the other hand, it is the best possibility if one does not want to assume a global clock. A view in terms of dependence is sufficient for the following reasons. If a global clock is assumed, then a dependence ordering implies a temporal ordering, in the real physical reality as well as in any conceivable computer implementation (Lamport, 1986). However, a real temporal ordering is only required if state transitions have side effects not modelled as communication with other components in the system. Thus, a view in terms of dependence is sufficient, if such side effects are modelled explicitly (which is the case for the semantic structure presented in this thesis). If no global clock is assumed (which is the only possibility in relativistic systems and which is often beneficial when modelling widely distributed systems, see (Pratt, 1986)), then no temporal order can be defined whatsoever, so a view in terms of dependence is the best possibility.

An interesting question is whether there are objective criteria to verify the claim that e.g. the logically instantaneous transmission property does indeed characterise instantaneous transmission. This question can be approached as follows. In the context of models of the dynamics of distributed systems, characterisations of various properties of information transmission have been published (Soneoka & Ibaraki, 1994; Charron-Bost *et al.*, 1996). These properties induce a proper hierarchy of classes of computations that (exclusively) use a specific information transmission property. I.e., Charron-Bost and her co-authors identify the following classes of computations: synchronous computations, causally ordered computations, FIFO-computations and asynchronous computations. A FIFO-computation is a computation in which all information exchange is order-preserving in the terminology used in this thesis. A causally ordered computation is a computation in which the order-preserving property not only holds for each individual information link, but also between all links to a specific component. Charron-Bost and her co-authors show that the class of synchronous computations is a proper subclass of the class of causally ordered computations. The class of causally ordered computations is a proper subclass of the class of FIFO-computations, which is itself a proper subclass of the class of asynchronous computations.

The properties formulated in (Soneoka & Ibaraki, 1994; Charron-Bost *et al.*, 1996) are expressed in terms of a global, event based model of a distributed system. In Chapter 7, a comparable global perspective is developed for the semantic structure developed in this thesis. The characterisations published in (Soneoka & Ibaraki, 1994; Charron-Bost *et al.*, 1996) may serve as formal criteria to verify the properties expressed above by considering those properties from the global perspective developed in the next chapter.

For the logically instantaneous information transmission property, this approach provides the following criteria:

- If for all compatibility relations in a structure hierarchy the logically instantaneous transmission property holds, the behaviour of that structure hierarchy is in the class of synchronous computations. If, for all compatibility relations in a structure hierarchy, the order-preserving transmission property holds, then the behaviour of that structure hierarchy is the class of FIFO-computations. As the class of synchronous computations is a subclass of the classes of FIFO computations, the order-preserving transmission property should by implication hold for a compatibility relation for which the logically instantaneous transmission property holds. Note that this criterion does not rely on a global perspective;

- In (Charron-Bost *et al.*, 1996), a refinement of the causality relation for synchronous computations is given. This relation essentially expresses the requirement that two components involved in synchronous information exchange share the same past and future at their (local) time points at which

this information exchange happens. One way to evaluate the definition of Section 6.2.4 consists of defining a requirement similar to the requirement of Charron-Bost *et al.* and to prove that this requirement is equivalent with the definition of the logically instantaneous information transmission property;

- In (Charron-Bost *et al.*, 1996), an equivalent characterisation of synchronous computations is given by a requirement on the common 'happens before' relation as defined by Lamport. (This requirement states that there should exist at least one linear extension of the 'happens before' relation such that for all pairs of corresponding send and receive events, the interval between two corresponding events is empty. Similar to the previous point, a possible way to evaluate the definition of Section 6.2.4 consists of defining a requirement similar to the requirement of Charron-Bost *et al.* and to prove that this requirement is equivalent with the definition of the logically instantaneous information transmission property;
- In (Soneoka & Ibaraki, 1994), yet another characterisation of synchronous computations is given. Soneoka and Ibaraki develop a formal notion that precisely determines whether messages cross each other. They then prove that a communication is synchronous iff no messages can cross. Thus, yet another way to evaluate the definition of Section 6.2.4 consists of defining a requirement similar to the requirement of Soneoke and Ibaraki and to prove that this requirement is equivalent with the definition of the logically instantaneous information exchange property. As an aside, in (Charron-Bost *et al.*, 1996), the no-message-crossing property is also mentioned.

Chapter 7

Global Perspectives

As stated in Chapter 1, an important characteristic of the semantic structure presented in the previous chapters is locality: the semantic structure provides a local view on the state and dynamics of a compositional system. A more global perspective on the dynamics of a multi-agent system and a compositional system is, however, in some cases interesting. In the first place, a more global perspective provides additional insight in the connections between individual components. A more global perspective enables a detailed comparison with other models of dynamics of concurrent systems, which often only have a global perspective. In the second place, a more global perspective facilitates the development of a refinement of the semantic structure presented in the previous chapters. This refinement, to which Chapter 8 is devoted, involves relations between, on the one hand, components that control other components and, on the other hand, the components controlled by other components.

In this chapter, in Section 7.1, a notion of the global state of the components and links in a structure hierarchy is developed. This notion is discussed and compared in detail with other models of concurrency in Section 7.2. Proofs are collected in Section 7.3.

7.1 Global State in Distributed Systems

The aim of this chapter is to develop a more global perspective on the behaviour of components and links in a compositional system. A global perspective is, like the three views on behaviour defined in Chapter 5, relative to a structure hierarchy. As a consequence, a global perspective on the behaviour of a compositional system is not unique, but depends on a structure hierarchy that describes (part of) the compositional system. One could argue that given a structure hierarchy *SH*, the glass box view on the behaviour of the components and links in *SH* provides a global perspective. However, although the glass box view models the behaviour of single components and links in *SH* in the context of the behaviour of all other components and links in *SH*, the glass box view does not define the *global state* of (the compositional system consisting of) the components and links in *SH*.

The main contribution of this chapter, and the main difference between global perspectives developed in this chapter and the perspective provided by the glass box view, is the notion of a global state. In Section 7.1.1, this notion is defined in a declarative style. Section 7.1.2 presents a similar definition in terms of a transition system that enables the generation of the set of global states for a given structure hierarchy. These subsections are deliberately kept concise and technical. An informal discussion of the notion of global state developed in these subsections is deferred until Section 7.2.1, which is dedicated to a discussion of models of concurrency developed in Theoretical Computer Science in the last decades. In Section 7.2.2, a detailed, formal comparison of the notion of global state developed and a similar notion proposed by Mattern (1992) is presented.

7.1.1 A Notion of Global State

The starting point for the definition of a global state is a structure hierarchy *SH* or a component structure *CS*, a collection of compatibility relations γ, and a collection of sets of local component and link traces (Γ_S) for a specific index set *S*. Given a component *C* in *SH*, the behaviour of the components and links in *SH* can be described by the three views $Beh_{WB}(C,CS,\gamma,(\Gamma_S))$, $Beh_{BB}(C,SH,\gamma,(\Gamma_S))$ and $Beh_{GB}(C,SH,\gamma,(\Gamma_S))$ developed in Section 5.2.2. These definitions are repeated here for ease of reference.

Definition 5.25. (Component behaviour, white box view). *Let C be a component, let* $CS=\langle Comp;Lnk;\prec;dom;cdom\rangle$ *be a non-empty composition structure for C, let* $\gamma=(\gamma_I)_{I\in Lnk}$ *be a collection of compatibility relations and let* $\Gamma=(\Gamma_S)_{S\in SLC(C,CS)}$ *be a collection of sets of traces such that for all* $S\in SLC(C,CS)$, $\Gamma_S\subseteq Beh_{loc}(S)$. *The* white box view *on the behaviour of C,* $Beh_{WB}(C,CS,\gamma,\Gamma)$, *with respect to CS,* γ *and* Γ *is the set of compatible multitraces* $\mu\in MT_{CS}$ *of C such that for each subcomponent or link S of C it holds that the local component trace of S in* μ *is an element of* Γ_S. *Formally:*

$$Beh_{WB}(C,CS,\gamma,\Gamma) = \{\, \mu \mid \mu\in MT_{CS} \text{ is compatible for } \gamma \text{ and } \forall S\in SLC(C,CS): \mu_S\in\Gamma_S \,\}.$$

Definition 5.26. (Component behaviour, black box view). *Let C be a component, let* $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ *be a non-empty structure hierarchy for C, let* $\gamma=(\gamma_I)_{I\in Lnk}$ *be a collection of compatibility relations and let* $\Gamma=(\Gamma_S)_{S\in SLC(C,SH)}$ *be a collection of sets of traces such that for all* $S\in SLC(C,SH)$, $\Gamma_S\subseteq Beh_{loc}(S)$. *The* black box view $Beh_{BB}(C,SH,\gamma,\Gamma)$ *for C with respect to SH,* γ *and* Γ *on the behaviour of C is the subset of* $Beh_{loc}(C)$ *such that each local component trace in this subset is part of a compatible multitrace for SH that is based on the given traces of the subcomponents and links of C. Formally:*

$$Beh_{BB}(C,SH,\gamma,\Gamma) = \{\, \mu_C \mid \mu\in MT_{SH} \text{ is compatible for } \gamma, \forall S\in SLC(C,SH): \mu_S\in\Gamma_S \text{ and } \forall I\in Lnk \text{ such that } dom(I)=cdom(I)=C: \mu_I\in Beh_{loc}(I) \,\}.$$

Definition 5.27. (Component behaviour, glass box view). *Let C be a component, let* $SH=(Comp;Lnk;\prec;dom;cdom)$ *be a non-empty structure hierarchy for C, let* $\gamma=(\gamma_l)_{l\in Lnk}$ *be a collection of compatibility relations and let* $\Gamma=(\Gamma_S)_{S\in Prim(SH)}$ *be a collection of sets of traces for the primitive components in SH. The* glass box view $Beh_{GB}(C,SH,\gamma,\Gamma)$ *for C with respect to SH,* γ *and* Γ *on the behaviour of C is the subset of the set* MT_{SH} *of multitraces for SH such that for all* $\mu\in Beh_{GB}(C,SH,\gamma,\Gamma)$ *it holds that:*

- μ *is compatible for* γ*;*
- $\forall C'\in Prim(SH)\setminus\{C\}$: $\mu_{C'}\in\Gamma_{C'}$ *and*
- $\forall S\in Comp\cup Lnk$: $\mu_S\in Beh_{loc}(S)$.

The elements of $Beh_{BB}(C,SH,\gamma,\Gamma)$ are local component traces. The elements of $Beh_{WB}(C,CS,\gamma,\Gamma)$ and $Beh_{GB}(C,SH,\gamma,\Gamma)$ are compatible multitraces, where each compatible multitrace is a multiset of local component traces, one for each component and link in *SH*. Each different local component trace or compatible multitrace represents different behaviour of the components and links in *SH*.

Global states are defined in terms of another notion (i.e., *snaphots*). Given a compatible multitrace μ, a snapshot selects one local component or link state from each local trace in μ. In other words, a snapshot is a function that maps every component and link in $SH=(Comp;Lnk;\prec;dom;cdom)$ to a state of that component or link. The co-domain of this function is thus the union of all sets $\mathcal{S}_S$ for $S\in Comp\cup Lnk$. In the development of a notion of global state, a binary relation on this set is needed. As the sets of states of different components are not necessarily disjoint, coloured versions of the states, as defined in Definition 6.10, are used. (Similar to the definition of dependence in Chapter 6.) A snapshot is formally defined as follows:

Definition 7.1. (Snapshot). *Let* $SH=(Comp;Lnk;\prec;dom;cdom)$ *be a structure hierarchy, let* $\gamma=(\gamma_l)_{l\in Lnk}$ *be a collection of compatibility relations and let* μ *be a multitrace compatible for* γ*. A* snapshot *of SH for* μ *and* γ *is a function* $snap(SH,\mu,\gamma)$: $Comp\cup Lnk\rightarrow\bigcup_{S\in Comp\cup Lnk}\mathcal{S}_S$ *such that for all* $S\in Comp\cup Lnk$, $snap(SH,\mu,\gamma)(S)=\mu_S(i)$ *for some i. The set of all snapshots of a structure hierarchy SH and a multitrace* μ *is denoted* $SNAP(SH,\mu,\gamma)$*. A snapshot is typically denoted by* σ*.*

A snapshot is thus a multiset indexed by the components and links in *SH* containing a local state of each component and link in *SH*. If μ is an element of the black box view on the behaviour of the components and links in a particular structure hierarchy, a snapshot consists solely of a local state of the top component of that structure hierarchy. Figure 7.1 depicts a snapshot for the multitrace depicted in Figure 5.1.

Alternatively, a snapshot can be defined as an element of the Cartesian product of the sets of local states of the components and links in *SH*, as witnessed by the following definition:

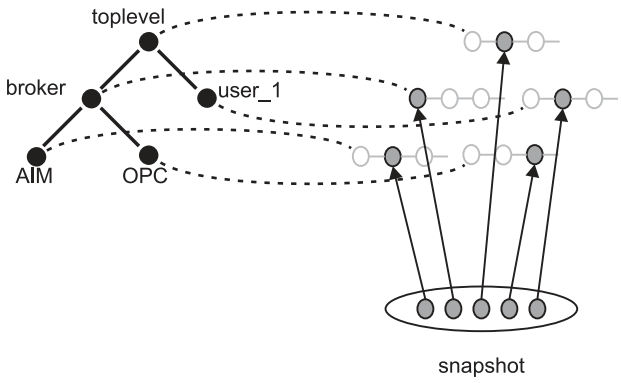

Figure 7.1: A snapshot for the multitrace of Figure 5.1.

Definition 7.2. (Snapshot, alternative definition). *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy, let γ be a collection of compatibility relations and let μ be a multitrace compatible for γ. A* snapshot *of SH and μ is a tuple $\langle v_{S_1};\ldots;v_{S_n}\rangle \in \prod_{S \in Comp \cup Lnk} S_S$ such that for all $S \in Comp \cup Lnk$, $v_S=\mu_S(i)$ for some i.*

Given a structure hierarchy $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$, as is indicated by the definition of a snapshot, a snapshot selects local component and link states from all components and links in $Comp \cup Lnk$, at all levels with respect to the hierarchy relation $\prec$. Consequently, the definition of a snapshot does not constrain possible selections of local states for the subcomponents of a composed component. In other words, it is not the case that for a subcomponent or link $S \prec C$, the definition of a snapshot requires $\sigma(S)$ to have any relation with $\sigma(C)$. This fact is further discussed at the end of this subsection.

As stated just before Definition 7.1, the global state of a structure hierarchy is defined in terms of a snapshot. The relation between snapshots and global states is as follows. Given a multitrace μ that is compatible for γ, any selection of local states from the traces in μ is a snapshot in the set $SNAP(SH,\mu,\gamma)$. Consider the behaviour of the components and links in SH as described by μ. At each moment in time, the current local state of each component and link in SH can be distinguished, and, as time passes, local transitions from one local state to the next occur. A straightforward, but, for reasons to be presented shortly, unsound notion of a *global state* of the structure hierarchy SH can be defined as the composition of the *local states* of all components and links in *SH at the same moment in time*. In other words, the global state is a snapshot σ such that each local state $\sigma(S)$ in the snapshot is the state of the component or link S at the same moment in time. However, by defining a global state as the composition of local states of components and links at the same moment in time, a notion of simultaneity based on global time is assumed, that determines equivalent moments in time. However, as discussed in Section 7.2.1, global time is generally considered to be undefined in

a distributed system (Lamport, 1978, 1986; Pratt, 1986). Therefore, a different definition of global state is needed, which does not rely on a notion of simultaneity based on global time.

The following observation suggests a solution. Not *every* snapshot of a structure hierarchy SH and a multitrace μ is a global state of SH, because some selections of local states of the components and links in SH cannot be considered simultaneous in the behaviour described by μ. An example of a snapshot that is not a global state is a snapshot containing two local states $v_{A,i}$ and $v_{B,j'}$ of two components A and B, in which in $v_{B,j'}$ information is available that, according to compatibility, is produced by state $v_{A,i}$ of A. In other words, $v_{B,j'}$ depends on $v_{A,i}$, or $v_{A,i} \rightarrow_d v_{B,j'}$ in the notation of Definition 6.11. In this case, $v_{A,i}$ and $v_{B,j'}$ cannot be considered to be simultaneous states. (If they were simultaneous, then information transmission from A to B would be instantaneous.) Therefore, the snapshot containing $v_{A,i}$ and $v_{B,j'}$ cannot be considered a global state.

The solution suggested in the previous paragraph is as follows. A global state is defined as a snapshot in which every pair of local component or link states may possibly happen simultaneously, where two states may happen simultaneously only if neither depends on the other, or, in other words, if they are not related by the dependence relation $\rightarrow_d$ defined in Definition 6.11. (Whether they indeed happen simultaneously is not relevant, as explained in Section 7.2.1 below.) Independence of two states is formally represented as follows:

Definition 7.3. (Independence). *Let $\rightarrow_d$ be the dependence relation for states. Two states $v_{A,i}$ and $v_{B,j'}$ are* independent, *denoted $v_{A,i} \mathbin{/\!\!/_d} v_{B,j'}$, iff $v_{A,i} \not\rightarrow_d v_{B,j'}$ and $v_{B,j'} \not\rightarrow_d v_{A,i}$.*

A global state is defined as a snapshot such that all states that are selected from the local component and link traces in a multitrace μ are pairwise independent. Formally:

Definition 7.4. (Global state). *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy, let γ be a collection of compatibility relations and let μ be a multitrace compatible for γ. A* global state *of the compositional system described by SH is a snapshot $\sigma \in SNAP(SH,\mu,\gamma)$ such that for all $S_1 \neq S_2 \in Comp \cup Lnk$, $\sigma(S_1) \mathbin{/\!\!/_d} \sigma(S_2)$. The set of all global states of SH for μ and γ is denoted $GS(SH,\mu,\gamma)$.*

This definition does not depend on the availability of a notion of global time. Instead, the notions of dependence and independence are used.

The notion of dependence defined in Definition 6.11 is applicable for proper traces (as defined by Definition 6.2) as well as for traces that do not have the properness property. Some notions developed in the rest of this chapter, however, only apply to multitraces consisting of proper traces. Therefore, a notion of *strict dependence* is introduced. This definition assumes that all traces in μ are proper traces, to ensure the existence of $prev_{\mu(S)}(v)$ and $next_{\mu(S)}(v)$.

Definition 7.5. (Strict dependence). *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy and let $LT_A=\langle\langle T_A;<_A\rangle;V_A\rangle$, $LT_B=\langle\langle T_B;<_B\rangle;V_B\rangle$ and $LT_L=\langle\langle T_L;<_L\rangle;V_L\rangle$ be three traces of components or links $A,B\in Comp\cup Lnk$ and a link $L\in Lnk$ such that $A=dom(L)$ and $B=cdom(L)$. Let $v_{A,i}$ and $v_{A,j}$ be two states in LT_A, let $v_{L,i''}$, $v_{L,j''}$, $v_{L,k}$ and $v_{L,l}$ be four states in LT_L, and let $v_{B,i'}$ and $v_{B,j'}$ be two states in LT_B. The* strict dependence *relation $\rightarrow_{sd}\subseteq S_{SH}\times S_{SH}$ for LT_A, LT_L and LT_B is defined as the smallest relation $\rightarrow_{sd}$ such that $v_{A,i}\rightarrow_{sd}v_{B,j'}$ iff either*

1. *$A=B$ and $v_{B,j'}\in next_{LT_A}(v_{A,i})$, or*
2. *$\langle\langle v_{A,i};v_{A,j}\rangle;\langle v_{L,i''};v_{L,j''};v_{L,k};v_{L,l}\rangle;\langle v_{B,i'};v_{B,j'}\rangle\rangle$ is a transmission octet for LT_A, LT_L and LT_B, for $v_{A,j}\in next_{LT_A}(v_{A,i})$ and $v_{B,i'}\in prev_{LT_B}(v_{B,j})$ or*
3. *There is a state $v_{C,m}$ of a component or link $C\in Comp\cup Lnk$ such that $v_{A,i}\rightarrow_{sd}v_{C,m}$ and $v_{C,m}\rightarrow_{sd}v_{B,j'}$.*

For realistic compositional systems, the strict dependence relation is assumed to be a partial order (to avoid that cycles of states are possible that all depend on one another). It is *not* the case that for proper traces, dependence and strict dependence are equivalent. Consider, for example, the two traces depicted in Figure 7.2, in which vertical lines represent the discrete time points induced by the next state relation on a dense time line (in this figure, the link from A to B is left implicit). The four black dots, together with four states of a link (which are not depicted in Figure 7.2) between A and B constitute a transmission octet.

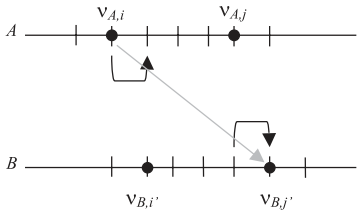

Figure 7.2: Comparing dependence and strict dependence.

It is clear that $v_{A,i}\rightarrow_{d}v_{B,j'}$ by clause 2 of Definition 6.11. However, it is not the case that $v_{A,i}\rightarrow_{sd}v_{B,j'}$ because $v_{A,j}$ is not an element of $next_{LT_A}(v_{A,i})$, and $v_{B,i'}$ is not an element of $prev_{LT_B}(v_{B,j'})$.

Thus, for proper traces, there are pairs of states such that one state depends on the other in the sense of Definition 6.11 (dependence), but not in the sense of Definition 7.5 (strict dependence). However, for proper traces, the difference between the two notions can be made more precise, which shows that both notions formalise the same phenomenon.

Section 6.1.2 motivates transmission octets by the observation that information transmission establishes a relation between the states involved in the transmission. (In fact, in the semantic structure developed in this thesis, it establishes a relation

between eight states.) The relation between the two states is as follows (in which the states mentioned refer to Figure 7.2): a state $\nu_{A,i}$ of the domain contains information, which is transmitted to the co-domain, in which a new state $\nu_{B,j'}$ results from this transmission. This relation between states has a one-to-one correspondence with a similar relation between state transitions. This relation between state transitions relates the following two transitions. On the domain side, there is a state transition from a state of the domain that contains information that is transmitted to the co-domain, and the immediate successor of that state. (This transition is depicted by a black arrow in the time line for *A* in Figure 7.1.) On the co-domain side, there is a relation from the immediate successor of the new state with the results of the transmission and that new state. (This transition is depicted by a black arrow in the time line for *B* in Figure 7.1.) For proper traces, both Definition 6.11 and Definition 7.5 define dependence implicitly in terms of this relation on transitions.

Apart from $\nu_{A,i}$ and $\nu_{B,j'}$, a transmission octet consists of four states of the link between *A* and *B*, a state $\nu_{A,j}$ of *A* and a state $\nu_{B,i'}$ of *B*. As explained in Section 6.1.2, state $\nu_{A,j}$ of *A* and a state $\nu_{B,i'}$ of *B* are used for the result of transmission and to denote an enabling state, respectively. The difference between dependence and strict dependence is that strict dependence requires the state used for the result of a transmission to be the immediate successor of $\nu_{A,i}$ and the enabling state of *B* to be the immediate predecessor of $\nu_{B,j'}$. Non-strict dependence only requires the state used for the result of a transmission to be *a* successor of $\nu_{A,i}$ and the enabling state of *B* to be *a* predecessor of $\nu_{B,j'}$. However, under the assumption of finite variability, it is always the case that there is a finite, maximal chain of discrete states between $\nu_{A,i}$ and $\nu_{A,j}$. It is also the case that there is a (finite) chain of discrete states between $\nu_{B,i'}$ and $\nu_{B,j'}$. Take the leftmost state in the chain from $\nu_{A,i}$ to $\nu_{A,j}$, or, if the length of this chain is one, take $\nu_{A,j}$. Take the rightmost state in the chain from $\nu_{B,i'}$ to $\nu_{B,j'}$, or, if the length of this chain is one, take $\nu_{B,i'}$. Non-strict dependence for these states and $\nu_{A,i}$, $\nu_{B,j'}$ is exactly the same as strict dependence for these states.

The notion of strict dependence gives rise to strict versions of the notions of independence and global states:

Definition 7.6. (Strict independence and strict global state).

- *Let $\rightarrow_{sd}$ be the strict causality relation for states. Two states $\nu_{A,i}$ and $\nu_{B,j'}$ are* strictly independent, *denoted $\nu_{A,i} /\!\!/_{sd} \nu_{B,j'}$, iff $\nu_{A,i} \not\rightarrow_{sd} \nu_{B,j'}$ and $\nu_{B,j'} \not\rightarrow_{sd} \nu_{A,i}$.*
- *Let SH=⟨Comp;Lnk;≺;dom;cdom⟩ be a structure hierarchy, let γ be a collection of compatibility relations and let μ be a multitrace compatible for γ. A* strict global state *of the compositional system described by SH is a snapshot $\sigma \in SNAP(SH,\mu,\gamma)$ such that for all $S_1 \neq S_2 \in Comp \cup Lnk$, $\sigma(S_1) /\!\!/_{sd} \sigma(S_2)$. The set of all strict global states of SH for μ and γ is denoted $SGS(SH,\mu,\gamma)$.*

In principle, each of the components and links in a compositional system represented by a structure hierarchy *SH* is active, which implies that its state changes over time. Consequently, the global state of the compositional system represented by *SH* also changes over time. This induces an order relation on the global states that models the temporal succession of global states. For strict dependence, this order is discrete. The next global state relation is first formally defined for linear, proper traces.

One may expect that the order relation on global states is defined as a binary relation on the set $GS(SH,\mu,\gamma)$ or $SGS(SH,\mu,\gamma)$. However, this is not the case for the following reason. Consider three global states σ, σ' and σ'' such that:

- σ and σ' only differ for the local state of a component *C*. In other words, $\sigma(C)\neq\sigma'(C)$ and for all $S\in Comp\setminus\{C\}\cup Lnk$, $\sigma(S)=\sigma'(S)$.
- σ' and σ'' only differ for the local state of a link *I* with *dom(I)=C*. In other words, $\sigma(I)\neq\sigma'(I)$ and for all $S\in Comp\cup Lnk\setminus\{I\}$, $\sigma(S)=\sigma'(S)$.

Suppose that $\langle\langle\sigma(C);\sigma'(C)\rangle;\langle\sigma(I);\sigma'(I);v_{I,k};v_{I,l}\rangle;\langle v_{cdom(I),i'};v_{cdom(I),j'}\rangle\rangle$ is a transmission octet for *I*. In other words, local state $\sigma(C)$ is a state in which information is available that is to be transmitted. As the domain end of *I* is co-located with *C*, the local state transition from $\sigma(C)$ to $\sigma'(C)$ should be simultaneous with respect to local time with the local state transition from $\sigma(I)$ to $\sigma'(I)$. Thus, σ'' should be an immediate successor of σ, and global state σ', which is an element of $GS(SH,\mu,\gamma)$, should not even be considered for the definition of the next global state relation. Therefore, in the next global state relation, states σ and σ' are considered to be equivalent, and the next global state relation is defined on the set of equivalence classes.

Consider two global states $\sigma,\sigma'\in SGS(SH,\mu,\gamma)$ for a structure hierarchy $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ such that σ' is a successor of σ and there is no global state σ'' such that σ'' is a successor of σ, and σ' is a successor of σ''. For reasons explained in Section 7.2.1, the definition of the global state order imposes an important restriction on σ and σ'. This restriction requires that for at least one and at most two different $S\in Comp\cup Lnk$, $\sigma(S)\neq\sigma'(S)$, and thus for all other $S'\neq S$, $\sigma(S')=\sigma'(S')$. Moreover, if there are two different $S_1,S_2\in Comp\cup Lnk$ such that $\sigma(S_1)\neq\sigma'(S_1)$ and $\sigma(S_2)\neq\sigma'(S_2)$, then it is required that $S_2\in Lnk$, and: $dom(S_2)=S_1$ and $cdom(S_2)=C$, or $dom(S_2)=C$ and $cdom(S_2)=S_1$ for some component *C*. A component or link S_1 and a link S_2 that comply with these requirements are co-located: S_1 is the domain or co-domain of S_2. Thus, the restriction can be reformulated informally as follows: between one global state and an immediate successor, there is exactly one time point at which the local state changes. As explained in Section 7.1.2, this restriction is necessary to accurately model the dynamics of a distributed compositional system under the assumption that global time is not available. In the definition below, the restriction introduced above is made formal. Moreover, an

extra condition is added: only if the local state $\sigma(S_1)$ of a component S_1 is involved in information transmission, a link S_2 can change its state.

In the formal definition of the next global state relation below, first the equivalence relation mentioned above is defined. The next global state relation is then defined on the set of equivalence classes of this relation. This is done as follows. First, for two equivalence classes X and Y, two strict global states $\sigma \in X$ and $\sigma' \in Y$ are selected such that σ is *maximal* in X and σ' is *minimal* in Y with respect to the order on the states in local component and link traces. As traces are assumed to be linear, σ and σ' are unique. A relation for such σ and σ' is then defined. The equivalence classes X and Y are related by the next global state relation if σ and σ' are related.

Definition 7.7. (Next global state). *Let $SH=(Comp;Lnk;\prec;dom;cdom)$ be a structure hierarchy, let $\gamma=(\gamma_l)_{l \in Lnk}$ be a collection of compatibility relations, and let μ be a multitrace compatible for γ. Let:*

- *R be the relation $\{(\sigma;\sigma') |$ there is a component C and a link l such that $\sigma'(C)=next_{\mu(C)}(\sigma(C))$ and $\sigma'(l)=next_{\mu(l)}(\sigma(l))$ and either:*
  - *$\langle(\sigma(C);\sigma'(C));(\sigma(l);\sigma'(l);v_{l,k};v_{l,l});(v_{cdom(l),i'};v_{cdom(l),j'})\rangle$ is a transmission octet, or*
  - *$\langle(v_{dom(l),i};v_{dom(l),j});(v_{l,i''};v_{l,j''};\sigma(l);\sigma'(l));(\sigma(C);\sigma'(C))\rangle$ is a transmission octet.*
- *$PART(SH,\mu,\gamma)$ be the set of equivalence classes with respect to $R \cup R^{-1}$ (where R^{-1} is the symmetric closure of R).*

Let $X,Y \in PART(SH,\mu,\gamma)$ be two equivalence classes with respect to $R \cup R^{-1}$, and let $\sigma \in X$, $\sigma' \in Y$ be two strict global states such that for all S in $Comp \cup Lnk$, there is no $\sigma_1 \in X$ with $\sigma_1(S)=next_{\mu(S)}(\sigma(S))$ and there is no $\sigma_2 \in Y$ with $\sigma_2(S)=prev_{\mu(S)}(\sigma'(S))$. The next global state *relation is a binary relation on $PART(SH,\mu,\gamma)$ such that $Y \in next_{SGS(SH,\mu,\gamma)}(X)$ iff either:*

- *$X=Y$;*
- *$X \neq Y$ and there is exactly one $S \in Comp \cup Lnk$ such that:*
  1. *$\sigma'(S) \in next_{\mu(S)}(\sigma(S))$, and*
  2. *there is no $l \in Lnk$ such that for any i,i',i'',j,j',j'',k,l: $\langle(v_{dom(l),i};v_{dom(l),j});(\sigma(l);\sigma'(l);v_{l,k};v_{l,l});(v_{cdom(l),i'};v_{cdom(l),j'})\rangle$ or $\langle(v_{dom(l),i};v_{dom(l),j});(v_{l,i''};v_{l,j''};\sigma(l);\sigma'(l));(v_{cdom(l),i'};v_{cdom(l),j'})\rangle$ is a transmission octet, and*
  3. *for all other components or links $S' \neq S \in Comp \cup Lnk$, $\sigma'(S)=\sigma(S)$, or*
- *$X \neq Y$ and there is exactly one component or link $S \in Comp \cup Lnk$ and a link $l \in Lnk$ with $dom(l)=S$ such that $\sigma'(S) \in next_{\mu(S)}(\sigma(S))$, $\sigma'(l) \in next_{\mu(l)}(\sigma(l))$, $\langle(\sigma(S);\sigma'(S));(\sigma(l);\sigma'(l);v_{l,k};v_{l,l});(v_{cdom(l),i'};v_{cdom(l),j'})\rangle$ is a transmission octet and*

for all other components or links $S' \in Comp \cup Lnk$, if $S' \neq S$ and $S' \neq I$, then $\sigma'(S') = \sigma(S')$, or

- *$X \neq Y$ and there is exactly one component or link $S \in Comp \cup Lnk$ and a link $I \in Lnk$ with $cdom(I) = S$ such that $\sigma'(S) \in next_{\mu(S)}(\sigma(S))$, $\sigma'(I) \in next_{\mu(I)}(\sigma(I))$, $\langle\langle v_{dom(I),i}; v_{dom(I),j}\rangle; \langle v_{I,i''}; v_{I,j''}; \sigma(I); \sigma'(I)\rangle; \langle\sigma(S); \sigma'(S)\rangle\rangle$ is a transmission octet and for all other components or links $S' \in Comp \cup Lnk$, if $S' \neq S$ and $S' \neq I$, then $\sigma'(S') = \sigma(S')$, or*
- *$X \neq Y$ and there is an equivalence class Z such that $Z \in next_{SGS(SH,\mu,\gamma)}(X)$ and $Y \in next_{SGS(SH,\mu,\gamma)}(Z)$.*

The pair $\langle PART(SH,\gamma,\mu); next_{SGS(SH,\mu,\gamma)}\rangle$ has the structure of a lattice, as is shown by the following proposition:

Proposition 7.8. *Let $SH = \langle Comp; Lnk; \prec; dom; cdom\rangle$ be a structure hierarchy, let $\gamma = (\gamma_I)_{I \in Lnk}$ be a collection of compatibility relations, and let μ be a multitrace compatible for γ. If $\rightarrow_{sd}$ is an irreflexive partial order, then the pair $\langle PART(SH,\gamma,\mu); next_{SGS(SH,\mu,\gamma)}\rangle$ is a lattice.*

This proposition is important for two reasons. First, it enables a formal comparison with another notion of global states which can be attributed to Mattern (1992) and which is based on the ideas of Lamport (1978, 1986). As the title of Mattern's (1992) paper indicates, in his approach global states also form a lattice. A comparison with Mattern's notion is presented in Section 7.2.2. Moreover, Proposition 7.8 indicates a relation between Katz and Peled's (1990) Interleaving Set Temporal Logic (ISTL) and the semantic structure developed in this thesis. ISTL, which is a logic to reason with global states, is interpreted over models with a lattice structure, similar to the lattice of strict global states. A discussion of ISTL is presented in Chapter 12.

Second, this proposition shows that $next_{SGS(SH,\mu,\gamma)}$ is a partial order (if $\rightarrow_{sd}$ is a partial order). In general, this order is not total: a particular global state might have successors that are not related by $next_{SGS(SH,\mu,\gamma)}$. It is insightful to compare this to the use of partial orders in other formal models of concurrency. Usually, unrelated successors of a state in a formal model of concurrency represent choices between different behaviour, and the partial order as a whole represents all possible behaviours. In a deterministic setting, the choices taken are fully determined by the environment of the modelled system, while in a nondeterministic setting, arbitrary choices may be made. In both cases, it is the compositional system of which the behaviour is modelled by the partial order that actually chooses one of the different alternatives. Consequently, for a fixed time interval, different observers of the compositional system all observe the same behaviour of the compositional system.

However, apart from representing choices taken within components in a compositional system, the partiality of the order $next_{SGS(SH,\mu,\gamma)}$ also represents a

different phenomenon. This is best explained by assuming, for the moment, that all local component and link traces in a compatible multitrace μ are linear (i.e., each state in each local component or link trace has a unique next state). Consider the next global state relation $next_{SGS(SH,\mu,\gamma)}$ and the set of global states $SGS(SH,\gamma,\mu)$ relative to this compatible multitrace μ. In the semantic structure developed in this thesis, each compatible multitrace μ represents *one specific behaviour*. Consequently, the set of global states $SGS(SH,\gamma,\mu)$ and its partial order $next_{SGS(SH,\mu,\gamma)}$ necessarily represent one specific behaviour. (In the semantic structure developed in this thesis, *sets* of compatible multitraces such as the white box view represent all possible behaviours.) The compositional system of which the specific behaviour μ is modelled by the set of global states $SGS(SH,\gamma,\mu)$, and its partial order $next_{SGS(SH,\mu,\gamma)}$ do not make any choices: at any moment in time, all components and links in SH each have a unique next local state. However, because a notion of global time is not assumed to be available and because it is assumed that different observers of the compositional system are subject to nondeterminism with respect to their observations, it is not possible to totally order the global states of the compositional system. Thus, the partial order on the set of global states $SGS(SH,\gamma,\mu)$ represents the different observations made by different observers. In Section 7.2.1, the assumption of nondeterminism with respect to observations is clarified.

It is difficult to define a next global state relation for traces that do not have the properness property or that are not linear. For traces that are proper but not linear, Definition 7.7 can easily be adapted. In this case, the only difference is that $next_{\mu(S)}(v_S)$ is no longer unique. In general, the next global state relation does not define a lattice in this case, as two states on different branches do not always have a lowest upper bound. For traces that do not have the properness property, Definition 7.7 must be adapted in such a way that the relation $next_{\mu(S)}$ is no longer needed. This may be achieved by changing occurrences of the form $next_{\mu(S)}(v_S)$ in the definition by the requirement that there is a state v' in $\mu(S)$ such that v' occurs later than v_S. As for two arbitrary global states it is not always possible to find an upper bound that is the lowest upper bound, the resulting order most likely does not have a lattice structure. In this thesis, the next global state relation is mainly used to compare the notion of global state developed in this chapter to approaches found in the literature on modelling distributed systems. In these approaches, traces are considered to be proper and linear. Therefore, Definition 7.7 is not adapted for traces that do not have the properness property or that are not linear.

Definition 7.4 is stated (via independence) in terms of the dependence relation, which, in turn, is stated in terms of transmission octets (among others). The following proposition establishes a direct relation between transmission octets and global states.

Proposition 7.9. *Let $SH=(Comp;Lnk;\prec;dom;cdom)$ be a structure hierarchy, let γ be a collection of compatibility relations, let μ be a multitrace compatible for γ and let*

$\sigma \in SNAP(SH,\mu,\gamma)$ be a snapshot. If for all $S_1 \neq S_2 \in Comp \cup Lnk$ and for all j' such that $\perp < j' \leq j$, with j such that $\sigma(S_1) = \mu_{S_1}(j)$, there is no transmission octet $\langle\langle \sigma(S_1); \sigma(S_1)' \rangle; \langle v_{L,i''}; v_{L,j''}; v_{L,k}; v_{L,l} \rangle; \langle prev_{\mu(S_2)}(\mu_{S_2}(j')); \mu_{S_2}(j') \rangle\rangle \in \gamma_L$ for any link L from S_1 to S_2, $\sigma(S_1)' \in next_{\mu(S_1)}(\sigma(S_1))$ and i'', j'', k, l in the time frame of μ_L, then σ is a strict global state.

The converse of Proposition 7.9 is not true.

The end of this subsection recurs to the remark made earlier in this subsection on the absence of a requirement on the relation between local component or link states in a snapshot of a component or link S with $S \prec C$ and the local component state of C in this snapshot. Such a requirement is also absent from the definition of a global state given above or from the more direct characterisation of global states given by Proposition 7.9. Given a structure hierarchy $SH = \langle Comp; Lnk; \prec; dom; cdom \rangle$, all components or links in $Comp \cup Lnk$ are treated on an equal footing, regardless of their level in the hierarchical composition relation $\prec$. However, because the depence relation is a relation on all states in $\mathcal{S}_{SH}$, the dependence relation induces a relation on the local component state $\sigma(C)$ and the local component or link states of all $S \in Comp \cup Lnk$ such that $S \prec C$, for σ a global state. Such an induced relation might be seen as a relation that defines a notion of 'constituent state': the global state of C consists of the 'constituent states' of its subcomponents and links. This notion is related to the notion of an *abstract event* in event-based approaches, where an abstract event is an event that itself consists of other events. Lamport (1986) extends the dependence relation (actually, a version of the dependence relation defined on events) to abstract events. Kshemkalyani (1998) systematically develops all 24 possible dependence relations on abstract events.

7.1.2 A Transition System

In this subsection, the set of global states associated with a structure hierarchy SH and a multitrace μ compatible for γ are defined as a subset of $SNAP(SH,\mu,\gamma)$ induced by a global transition system that is defined below. This definition of the set of global states is, like Definition 7.4, not defined in terms of global time. The definition developed in this subsection provides a more detailed, constructive notion of global state.

The transition system, called the *global transition system* of a given structure hierarchy SH, is a pair $\langle SNAP(SH,\mu,\gamma); \rightarrow_g \rangle$, where $\rightarrow_g$, the transition relation, is called the *global transition relation*. The global transition relation is defined as follows: there is a transition from a snapshot σ to a snapshot σ' iff there is either an internal global transition, a sending global transition, or a receiving global transition from σ to σ' (these three types of global transitions are defined below).

The global transition relation is related to properties of compatibility relations as defined in Chapter 6. Chapter 6 presents a number of properties of compatibility relations, such as the lossless transmission property and the order preserving

transmission property. These properties model properties of information transmission that may or may not be adopted in an application of the framework developed in this thesis. The properties presented in Chapter 6 only specify *what* is understood to be information transmission in particular applications, and not *how* information transmission can be realised (i.e., what actions to take) such that information transmission indeed possesses the properties attributed to it. In other words, Chapter 6 is dedicated to a declarative, or intensional discussion of information transmission.

The notion of global transitions defined below enables a more operational, or extensional, formalisation of information transmission properties. In fact, the definition of global transitions is suitable for 'encoding' properties of information transmission of the form presented in Chapter 6. The general idea is as follows. Consider a snapshot σ, which is a selection of local states, one for each component or link in a specific structure hierarchy *SH*. The global transition relation relates σ to a set of possible successor snapshots. A successor snapshot σ' differs from σ for one or two local component or link states, depending on whether the global transition is a send global transition, a receive global transition or an internal global transition.

- Consider a snapshot $\sigma \in SNAP(SH,\mu,\gamma)$ and a component or link $S \in Comp \cup Lnk$. The local state of S is given by $\sigma(S)$. From a local point of view, the next local state is determined by the local component or link trace of S in μ. A successor of σ is thus a snapshot σ' such that $\sigma'(S) \in next_{\mu(S)}(\sigma(S))$. Now consider all links in *Lnk* connected to S, that is, all links I such that $dom(I)=S$ or $cdom(I)=S$. For each of these links, a compatibility relation λ_I is assumed to be given. Assume that there is no transmission octet $\langle\langle\sigma(S);\sigma'(S)\rangle;\langle\sigma(I);v_{I,j''};v_{I,k};v_{I,l}\rangle;\langle v_{cdom(I),i'};v_{cdom(I),j'}\rangle\rangle$ (if $dom(I)=S$) or $\langle\langle v_{dom(I),i};v_{dom(I),j}\rangle;\langle v_{I,i''};v_{I,j''};\sigma(I);v_{I,l}\rangle;\langle\sigma(S);\sigma'(S)\rangle\rangle$ (if $cdom(I)=S$) in any of these compatibility relations. This means that information transmission is not involved in the local transition of component S from $\sigma(S)$ to $\sigma'(S)$. The global transition from σ to σ' is an *internal global transition*, and for all $S'' \in Comp \cup Lnk$ *with* $S'' \neq S$: $\sigma'(S'')=\sigma(S'')$;

- Consider a snapshot $\sigma \in SNAP(SH,\mu,\gamma)$ and a component or link $S \in Comp \cup Lnk$. The local state of S is given by $\sigma(S)$. From a local point of view, the next local state is determined by the local component trace of S in μ. A successor of σ is thus a snapshot σ' such that $\sigma'(S) \in next_{\mu(S)}(\sigma(S))$. Now consider a link $I \in Lnk$ such that $dom(I)=S$ and assume that there is a transmission octet $\langle\langle\sigma(S);\sigma'(S)\rangle;\langle\sigma(I);\sigma'(I);v_{I,k};v_{I,l}\rangle;\langle v_{cdom(I),i'};v_{cdom(I),j'}\rangle\rangle \in \lambda_I$, for some $i',j' \in T_{\mu(cdom(I))}$ with $\mu(cdom(I))=\langle\langle T_{\mu(cdom(I))};<_{\mu(cdom(I))}\rangle;V_{\mu(cdom(I))}\rangle$, $\bot \leq i' < j'$. This means that information transmission is involved in the local transition of S from $\sigma(S)$ to $\sigma'(S)$. More specifically, the transition from $v_{cdom(I),i'}$ to $v_{cdom(I),j'}$ relies on receiving information from S. The global

transition from σ to σ' is a *sending global transition*. Moreover, together with the local transition of S from $\sigma(S)$ to $\sigma'(S)$, a transition of the state of link I occurs. Therefore, $\sigma'(I) \in next_{\mu(I)}(\sigma(I))$ and for all $S'' \in Comp \cup Lnk$ with $S'' \neq S$ and $S'' \neq I$: $\sigma'(S'') = \sigma(S'')$;

- Finally, consider a snapshot $\sigma \in SNAP(SH,\mu,\gamma)$ and a component or link $S \in Comp \cup Lnk$. The local state of S is given by $\sigma(S)$. From a local point of view, the next local state is determined by the local component trace of S in μ. A successor of σ is thus a snapshot σ' such that $\sigma'(S) \in next_{\mu(S)}(\sigma(S))$. Now consider a link $I \in Lnk$ such that $cdom(I)=S$ and assume that there is a transmission octet $\langle\langle v_{dom(I),i}; v_{dom(I),j}\rangle;\langle v_{I,i''}; v_{I,j''}; \sigma(I); \sigma'(I)\rangle;\langle\sigma(S);\sigma'(S)\rangle\rangle \in \lambda_I$, for some $i \in T_{\mu(dom(I))}$ with $\mu(dom(I)) = \langle\langle T_{\mu(dom(I))}; <_{\mu(dom(I))}\rangle; V_{\mu(dom(I))}\rangle$, $\bot \leq i < j$. This means that information transmission is involved in the local transition of S from $\sigma(S)$ to $\sigma'(S)$. More specifically, the transition of S from $\sigma(S)$ to $\sigma'(S)$ relies on the occurrence of a transition of S' from $v_{dom(I),i}$ to $v_{dom(I),j}$. The global transition from σ to σ' is a *receiving global transition*. Moreover, together with the local transition of S from $\sigma(S)$ to $\sigma'(S)$, a transition of the state of link I occurs. Therefore, $\sigma'(I) \in next_{\mu(I)}(\sigma(I))$ and for all $S'' \in Comp \cup Lnk$ *with* $S'' \neq S$ *and* $S'' \neq I$: $\sigma'(S'') = \sigma(S'')$;

The three scenarios sketched are formalised in the following definition:

Definition 7.10. (Global transition relation). *Let* $SH = \langle Comp; Lnk; \prec; dom; cdom\rangle$ *be a structure hierarchy, let* $\gamma = (\gamma_I)_{I \in Lnk}$ *be a collection of compatibility relations, let* μ *be a multitrace compatible for* γ *and let* $\sigma, \sigma' \in SNAP(S,\mu,\gamma)$ *be two snapshots. The* global transition relation $\rightarrow_g$ *for SH,* γ *and* μ *is a binary relation on* $SNAP(SH,\mu,\gamma)$ *such that* $\sigma \rightarrow g \sigma'$ *iff* $\langle\sigma;\sigma'\rangle$ *is either an internal global transition, a send global transition or a receive global transition. These transitions are defined as follows:*

- *The pair* $\langle\sigma;\sigma'\rangle$ *is an* internal global transition *iff there is exactly one component or link* $S \in Comp \cup Lnk$ *such that* $\sigma'(S) \in next_{\mu(S)}(\sigma(S))$, *while for all* $S' \in Comp \cup Lnk$ *with* $S' \neq S$: $\sigma'(S') = \sigma(S')$ *and for all links* $I \in Lnk$, *there is, for* i'', j'', k, l, i', j' *no transmission octet* $\langle\langle\sigma(S);\sigma'(S)\rangle;\langle v_{I,i''}; v_{I,j''}; v_{I,k}; v_{I,l}\rangle;\langle v_{cdom(I),i'}; v_{cdom(I),j'}\rangle\rangle \in \gamma_I$ *and no transmission octet* $\langle\langle v_{dom(I),i}; v_{dom(I),j}\rangle;\langle v_{I,i''}; v_{I,j''}; v_{I,k}; v_{I,l}\rangle;\langle\sigma(S);\sigma'(S)\rangle\rangle \in \gamma_I$;

- *The pair* $\langle\sigma;\sigma'\rangle$ *is a* sending global transition *iff there is exactly one component or link* $S \in Comp \cup Lnk$ *and exactly one link* $I \in Lnk$ *with* $dom(I) = S$ *such that* $\sigma'(S) \in next_{\mu(S)}(\sigma(S))$, $\sigma'(I) \in next_{\mu(I)}(\sigma(I))$ *and, for arbitrary* k, l, i', j', $\langle\langle\sigma(S);\sigma'(S)\rangle;\langle\sigma(I);\sigma'(I); v_{I,k}; v_{I,l}\rangle;\langle v_{cdom(I),i'}; v_{cdom(I),j'}\rangle\rangle \in \gamma_I$, *while for all* $S'' \in Comp \cup Lnk$ *with* $S'' \neq S$ *and* $S'' \neq I$: $\sigma'(S'') = \sigma(S'')$, *with* $k, l \in T_{\mu(I)}$, *where* $\mu(I) = \langle\langle T_{\mu(I)}; <_{\mu(I)}\rangle; V_{\mu(I)}\rangle$, *and* $i', j' \in T_{\mu(cdom(I))}$, *where* $\mu(I) = \langle\langle T_{\mu(cdom(I))}; <_{\mu(cdom(I))}\rangle; V_{\mu(cdom(I))}\rangle$;

- *The pair* $\langle\sigma;\sigma'\rangle$ *is a* receiving global transition *iff there is exactly one component or link* $S \in Comp \cup Lnk$ *and exactly one link* $I \in Lnk$ *with* $cdom(I) = S$ *such that*

$\sigma'(S) \in next_{\mu(S)}(\sigma(S))$, $\sigma'(I) \in next_{\mu(I)}(\sigma(I))$ and, for arbitrary i,j,i'',j'', $\langle\langle v_{dom(I),i}; v_{dom(I),j}\rangle;\langle v_{I,i''}; v_{I,j''}; \sigma(I); \sigma'(I)\rangle;\langle\sigma(S);\sigma'(S)\rangle\rangle \in \gamma_I$, while for all $S'' \in Comp \cup Lnk$ with $S'' \neq S$ and $S'' \neq I$: $\sigma'(S'') = \sigma(S'')$ with $i'',j'' \in T_{\mu(I)}$, where $\mu(I) = \langle\langle T_{\mu(I)}; <_{\mu(I)}\rangle; V_{\mu(I)}\rangle$, and $i,j \in T_{\mu(dom(I))}$, where $\mu(I) = \langle\langle T_{\mu(dom(I))}; <_{\mu(dom(I))}\rangle; V_{\mu(dom(I))}\rangle$.

With respect to this definition, a number of remarks can be made:

- The definition of a global transition does not contain a commitment to a specific property of information transmission. Instead, although this definition provides operational details that are not included in the definitions in Chapter 6, it is still general enough to capture different forms of information transmission. Properties of information transmission are properties of the information links. The properties of information transmission presented in Chapter 6 can be encoded by constraining the set of local component traces of information links. For instance, the order preserving property can be encoded by constraining local link traces such that the link behaves as a queue;
- In Section 2.2.7, two forms of information transmission are discussed: synchronous and asynchronous information transmission. For ease of reference, the relevant part of this discussion is summarised. Different researchers take different positions in the issue of synchronous versus asynchronous information transmission: whether synchronous information transmission is more basic than asynchronous information transmission or conversely. On the one hand, synchronous information transmission is a special form of asynchronous information transmission in which the capacity of the communication channels to hold information in transit is restricted. On the other hand, asynchronous information transmission can be realised by synchronous information transmission as follows. There is synchronous information transmission between the sending component and one end of the communication channel and there is synchronous information transmission between the other end and the receiving component. These two instances of information transmission are not synchronous with one another. In the semantic structure developed in this thesis, both views may be held. On the one hand, the definition of sending and receiving global transitions above corresponds to synchronous information transmission with the two ends of an information link: both the state of the sending (respectively receiving) component and of the link change at the same time. However, in this definition the transitions of two components S and S' that exchange information are not related. Thus, asynchronous communication is simulated by two instances of synchronous communication. On the other hand, if one abstracts from the detailed, operational view provided by the definition of global transitions above and

reverts to compatibility relations and their properties, asynchronous information transmission is the basic notion. Synchronous information transmission between components is then a special case of information transmission in which a property is adopted that restricts the capacity to hold information in transit;

- In a global transition, at most two local states change (and at least one). This is a consequence of the assumption that global time is not assumed to be available and is discussed in Section 7.2.1.

The definition of a transition system is now straightforward:

Definition 7.11. (Global transition system). *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy, let $\gamma=(\gamma_l)_{l\in Lnk}$ be a collection of compatibility relations and let μ be a multitrace compatible for γ. A* global transition system *for SH, μ and γ is a pair $GTS(SH,\mu,\gamma)=\langle SNAP(SH,\mu,\gamma);\rightarrow_g\rangle$.*

The following proposition establishes a relationship between global states as defined in the previous subsection, and the set of all snapshots that can be reached from a certain initial state (defined below) in one or more steps by the transition system.

Proposition 7.12. *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy, let $\gamma=(\gamma_l)_{l\in Lnk}$ be a collection of compatibility relations, let μ be a multitrace compatible for γ and let $GTS(SH,\mu,\gamma)$ be a global transition system for SH.*

- *Let $\sigma,\sigma'\in SNAP(SH,\gamma,\mu)$ be two snapshots. Let the* n-step successor *relation $\xrightarrow{n}_g$ on $SNAP(SH,\gamma,\mu)$ be defined inductively for $n\geq 0$ as follows: $\sigma\xrightarrow{0}_g\sigma'$ iff $\sigma=\sigma'$ and, for $n>0$, $\sigma\xrightarrow{n}_g\sigma'$ iff $\sigma\xrightarrow{n-1}_g\sigma''$ and $\sigma''\rightarrow_g\sigma'$;*
- *Let the* closure *of a set of global states be a function $^*: \mathcal{P}(SNAP(SH,\gamma,\mu))\rightarrow \mathcal{P}(SNAP(SH,\gamma,\mu))$ defined as follows:*
  $X^*=\bigcup_{n\geq 0}\{\sigma\in SNAP(SH,\gamma,\mu) \mid \sigma'\xrightarrow{n}_g\sigma \text{ for } \sigma'\in X\}$;
- *Let the* initial global state *of SH be the snapshot $q_0\in SNAP(SH,\gamma,\mu)$ such that for all $S\in Comp\cup Lnk$: $q_0(S)=\mu_S(\perp)$ (where $\perp$ is the beginning of time as defined in Definition 5.11);*

If $\sigma\in\{q_0\}^*$, then σ is a global state.

Similar to the next global state relation defined in Definition 7.7, the global transition relation is a partial order. Again, the partiality of this order represents not only choices made by the components, but also nondeterminism in the observation of the global transition system.

As a final remark, note that this thesis does not assume that a notion of a (non-distributed, sequential) automaton representing the behaviour of a distributed

system is conceptually valid. Therefore, this thesis does not claim that a transition system $GTS(SH,\mu,\gamma)$ is an abstract, non-distributed automaton that corresponds to a compositional system represented by SH.

7.2 Discussion

While developing the notion of global state in Section 7.1, some remarks were made with respect to the relationship between the notion of global state as developed in this chapter and other notions that have been developed in Computer Science. The notion of global state is an aspect of modelling concurrency, for instance in distributed systems. Therefore, in Section 7.2.1, an overview of concurrency models is presented to provide context for a more detailed comparison. A formal comparison of the notion developed in this chapter and other notions is presented in Section 7.2.2. Section 7.2.3 discusses a generalisation of the notion of strict dependence.

7.2.1 Modelling distributed systems

The notion of global state attracted considerable interest in the areas of Theoretical Computer Science and Distributed Systems research, as one of many aspects of modelling concurrency in distributed systems. In this section, a short survey of the results in these areas is provided, inspired by (Pratt, 1986, 1991).

The first formal models of concurrency found their roots in formal models of sequential computation. These models of sequential computation themselves were based on the concepts of transition systems, automata and formal languages. For sequential, non-distributed computations[8], there is no difference between local and global states, because there is no spatial dimension. A sequential computation is modelled as follows: it remains in a particular state for some time, in which it is not active. These static intervals are alternated by periods in which some activity takes place: an action occurs, which results in transition to a new state. Contrary to common parlance in Artificial Intelligence, such an action is not something that tries to alter the state of the environment of the computation. Instead, it is assumed to be internal and although the occurrence of an action is not part of the result of the computation, an occurrence of an action is assumed to be observable. Moreover, it is also assumed that an action is atomic: it does not consist of sub-actions.

A transition system assumes that a set of states Σ of the computation and a set A of action names are given and defines a ternary relation on these sets: the transition relation $TR \subseteq \Sigma \times A \times \Sigma$, where $\langle \sigma_1;a;\sigma_2 \rangle \in TR$ means that if the computation is in state

[8] A sequential, non-distributed computation is the activity of a single computer over some amount of time. A distributed computation is the collective activity of a number of computers, or processors, over some amount of time.

σ_1, performing action a changes the state to σ_2. An automaton is a transition system in which one particular state is distinguished as the starting state, and several other states are distinguished as ending states. Transition systems and automata induce an ordering on the set of states, but not (directly) on the set of actions. The ordering on states induced by a transition system is, in general, not linear, but only partial. This partiality represents states where computations have the possibility to choose between performing different actions. It is still assumed that the sequence of states of one particular computation are linearly ordered. As an aside, underlying this assumption is the definition of a computation as the activity or behaviour of a computer over some amount of time (also known as a *run*), and not as the collection of possible behaviours over different amounts of time (or, a collection of runs).

The formal language theory approach to modelling sequential computations takes an orthogonal view by defining an order on the actions: a sequential computation is identified with the sequence of the actions it performs, which is a word in the language of all computations. Technically, such a sequence or word is a linearly ordered multiset of action names. (A sequence is a multiset because actions can occur more than one time. To distinguish between different occurrences, a set E of symbols identifying action occurrences can be introduced. Each element from E is labelled by the action of which it represents an occurrence. This way, no element from E has to be in a sequence more than once and multisets are no longer needed.) While transition systems and automata order states, sequences order action occurrences. However, an order on states can easily be obtained from this linear order on action occurrences, by taking the state that results after each action occurrence. This order is again linear. As an aside, sequences of action occurrences are often called *traces*, which is an unlucky coincidence with temporal logic, in which a trace usually refers to a sequence of possible worlds or states[9].

The first approaches to modelling concurrency tried to stay as close to the approaches used for sequential computations as possible. To this end, these first approaches identified the concurrent occurrence of two actions a and b with a nondeterministic choice between their two possible sequential occurrences: either a followed by b or the other way around. The two possible sequential occurrences are called the *interleavings* of a and b. If the (atomic) actions a and b are assumed to have duration, this means that a has stopped before b begins (or the other way around): their occurrences are completely mutually exclusive. This approach is called the *atomic mutual exclusion* view on concurrent computations[10]. The choice between the two interleavings is nondeterministic and differs from computation

[9] This coincidence clearly shows the independent development of temporal logic and models of computation until Pnueli's (1977) landmark paper.

[10] The more common name 'interleaving' is not used here to avoid confusion with the notion 'interleaving view' which has a very specific meaning, to be defined below.

(or run) to computation. However, all observers of a *single* computation observe the same interleaving. As a consequence, the states and actions of a single computation are still linearly ordered. Pratt (1986) lists the following arguments advocating this approach:

- The atomic mutual exclusion view needed only minimal adaptations of the approaches used for modelling sequential computations. Formal language theory was extended with a binary operator on strings, the shuffle operator, which, given two strings, produces the interleavings of the actions in these strings. All other operators (concatenation, Kleene star, etc.) are still applicable;
- Every partially ordered set is representable as the set of its linearisations (Abraham *et al.*, 1990). Therefore, if a partial order would be needed to model concurrency, its set of linearisations suffices and is preferable, as the linearisations can easily be described using formal language theory;
- In real systems, time is totally ordered. Only in non-rigid systems (systems in which parts move with respect to one another), time is no longer totally ordered as is described by relativity theory. Especially in the early days of concurrency theory, when most concurrent systems were actually multiprogramming systems, the assumptions underlying the atomic mutual exclusion view were intuitively appealing. (In a multiprogramming system, there is no concurrency in the sense that several processors are computing and exchanging information simultaneously. Instead, one processor rapidly switches between executing several programs. For an observer, these programs appear to be running simultaneously. However, at any moment in time, only one program is active. In a multiprogramming system, the processor actually chooses between interleavings.)

An important disadvantage of this approach is that behaviour of a concurrent system is not preserved in the case of action refinement (Castellano *et al.*, 1987). Consider two processes, one of which performs an action a, while the other one performs action b. The behaviour of the concurrent execution of these processes is described by the set of the two interleavings $\{ab,ba\}$ (where the string ab models the case in which first a is performed, followed by b, and ba models the case in which first b is performed, followed by a). In either case, conform the mutual exclusion assumption, the first action is fully finished before the last action starts. Now suppose that action a is decomposed into two sequencial actions a_1 and a_2. The behaviour of the concurrent execution of processes a_1a_2 and b is described by the set of *three* interleavings $\{a_1a_2b,ba_1a_2,a_1ba_2\}$. The interleaving a_1ba_2 shows that behaviour is not preserved in the case of action refinement: action b occurs while action $a=a_1a_2$ has already started but not finished. Glabbeek and Goltz (1989) show that partial orders preserve behaviour in the case of action refinement. (As the title of their paper suggests, a model in between interleaving and partial orders in

expressive power is often sufficient for the preservation of behaviour in the case of action refinement.)

A second disadvantage of the atomic mutual exclusion view is of a more conceptual nature. The atomic mutual exclusion view implicitly assumes that a notion of global time is available, because the activities of concurrent processes are synchronised. However, the assumption that global time is available is conceptually invalid in current-day distributed systems for two reasons: (i) it is impossible to make global time available to physically separated parts of a wide-area distributed system and (ii) if it nevertheless were available, it cannot be proven to be available.

At first sight, global time seems to be available automatically and unavoidably. As stated above, time is totally ordered for rigid physical systems. As distributed computer systems are physical systems (most often rigid or almost rigid), at a physical (hardware) level, global time is available in the form of the everyday notion of continuous wall clock time. For this reason, it is tempting to assume that global (synchronised) time is available also at higher levels, and use the atomic mutual exclusion view to model distributed systems. However, at the level of interest for modelling distributed systems, i.e. at the level of abstraction of action performance and information exchange, continuous wall clock time, synchronised for the entire distributed system, is not available, for reasons explained below. If, nevertheless, models of distributed systems assume that global time is available, either implementing these systems is difficult (as each processor has to actively synchronise with all other processors), or the implemented systems may have erroneous properties that cannot be predicted by the model.

At the level of abstraction that is of interest for modelling distributed systems (and at higher levels), global time is not available because at this level, individual processors in a distributed system perform their processes in discrete steps, governed by a very regular square-wave electrical signal called the 'clock' signal, typically with a frequency between 1 and 1000 MHz. At this level, time is measured in terms of this clock signal. The clock signal of each processor is a function of wall clock time, but in general, this function varies from processor to processor. For the notion of time derived from the clock signal to be considered global, either the signal has to come from the same source for each processor, or different sources have to be synchronised[11].

Providing one source for the clock signal to which each processor is physically connected is only feasible for fine-grained parallel computers, i.e. with all

[11] It is probably helpful to compare these two possibilities to looms and workers in a nineteenth-century cotton mill. In those days, all looms were driven by a single, central steam engine, connected to the looms by chains and gears. The number of revolutions of the engine acted as a time source for the looms. The workers, however, each have their own, common human sense of time, which they could synchronise with one another by looking at the wall clock, that may or may not have been present.

processors connected to one backplane that distributes the clock signal. (It is even doubtful whether this is feasible for fine-grained parallel computers.) However, on a world-wide scale, already at the physical (hardware) level, immense if not insurmountable problems arise. (As stated above, it is dangerous that nevertheless, at higher levels, global time *is* available.) As in the area of multi-agent systems, focus is more on world-scale distribution (e.g., agents that roam the Internet) than on fine-grained parallel computers, the first possibility for providing global time is not applicable.

For the second possibility, individual sources have to be synchronised. Several algorithms for synchronisation with UTC (Universal Coordinated Time, which is measured using cesium clocks and broadcasted by special radio transmitters) exist (Tanenbaum, 1992, p. 471-476). However, Tanenbaum draws the conclusion that: "All in all, getting the clocks in a distributed system synchronised to within 5 or 10 msec of UTC is an expensive and nontrivial business." Compared to the speed of today's processors, synchronisation to an accuracy of 5 or 10 msec can only be called rudimentary at best.

It is thus very impractical, if not impossible, to provide clock signals that are synchronised or come from the same source. Even if it were possible to achieve synchronisation as envisioned in the atomic mutual exclusion view, it is of limited value. In current-day engineering practice, distributed systems are connected to other systems and/or observed using electronic information transmission, i.e. via computer networks. Such information transmission is very slow compared to the internal activity of the individual processors in a system (i.e., the discrete steps governed by the clock signal). Moreover, observation delays are unpredictable and in general vary from observer to observer of the same system. As a result, the temporal order of observed events is no longer well-defined: two observers of the *same* activity of the *same* distributed system may observe the system differently. As observation delays in a current-day wide-area distributed system are enormous compared to the internal processing speed of the processors in the network, these effects are as relevant as relativity is at speeds close to the speed of light. In this sense, global time cannot be proven to be available. It is therefore of limited value for modelling and analysing distributed systems in an engineering context.

Thus, given the focus of the multi-agent systems area at wide-area distributed systems consisting of autonomous processors (the agents) that are truly active simultaneously, the interleaving metaphor of a distibuted system as a centralised automaton that picks a non-deterministic but well-defined linear order of atomic actions is conceptually invalid. Instead, an alternative concept is needed as a foundation for modelling distributed systems. Many authors, e.g., (Lamport, 1978), propose to use causality as this alternative concept.

This concept is best introduced as follows. Similar to the formal language approach to modelling sequential computations, a set of events, or action occurences, forms the starting point. An event e_1 is causally dependent on an event e_2 if both occur in the same process and e_1 occurs (locally) later than e_2, or if both

occur in different processes and e_1 is an event in which information is received that is sent during the occurrence of event e_2, or if there is a third event e_3 on which e_1 is causally dependent and which itself is causally dependent on e_2. In any realistic distributed system, the causality relation must be a partial order, as, if it were not, cycles of events can exist that all depend causally on one another.

Thus, the starting point for the alternative view on modelling concurrency (often called the *true concurrency view*) consists of a set of events together with a partial order. In fact, in his paper, Pratt (1986) does not (explicitly) refer to causal dependence to introduce partial orders. Instead, Pratt proposes to generalise the formal language theory approach as follows. In the formal language theory approach, the behaviour of a distributed system is described by strings of actions performed by the system. A string is a linearly ordered set (or, actually, a multiset, as multiple occurrences are allowed) of action symbols. In his generalisation, Pratt allows strings to be partially ordered (partially ordered stings are called pomsets, for partially ordered multisets. In his paper, Pratt then proceeds to develop an algebraic model for concurrency based on partial orders. This work culminated in a very abstract algebraic framework, Chu spaces, applicable not only to model concurrency but also to reason about mathematical relations in general (Pratt, 1995). Pratt's work, which is almost completely algebraic in nature, is not exploited further in this thesis. As an aside, neither Pratt nor Lamport are the first authors to use partial orders. Partial orders already appear in Petri nets (Petri, 1962) and in Winskel's event structures (Winskel, 1989), which were developed roughly at the same time as Pratt's approach.

So far, the true concurrency view has been introduced as a generalisation of the formal language approach to modelling sequential computations. In fact, in the true concurrency view, events are predominant. However, it is possible to develop a notion of global state for a distributed system in the true concurrency view. If a distributed system is active, the set of all events that may occur for a specific processor can be partitioned in the set of events that have already occurred and the set of events that still have to occur. The state of a distributed system as a whole can be described by the set consisting of all events that have already occurred for each processor. In a realistic system, this set has to be closed with respect to the causal dependence relation: if this set contains an event e_1 that causally depends on an event e_2, then the set should also contain e_2. (Otherwise an action's effect would preceed its cause.) Such a set is called a *consistent cut*. Consistent cuts can be ordered by set inclusion. This order represents the activity of the distributed system over time: from the empty set (no events have occurred) to the set of all events. Mattern (1992) shows that the set of all consistent cuts ordered by set inclusion is a lattice. This is interesting, as time in relativity theory also has the structure of a lattice.

A consequence of using set inclusion to order consistent cuts is that the smallest transitions from one consistent cut to the next consist of pairs of sets that differ for only one element. (E.g., the transition from $\varnothing$ to $\{e_1\}$ to $\{e_1,e_2\}$ and not from $\varnothing$ to

$\{e_1,e_2\}$ as smallest transition.) The question is: is this correct, even if e_1 and e_2 actually happened simultaneously? To answer this question, first the notion of simultaneity in the absence of global time should be clarified. Suppose that a distributed system is observed by a number of (sequential) observers. Due to observation delays, some observers observe the events e_1 and e_2 in the sequence e_1e_2, while other observers observe the sequence e_2e_1. (These sequences are *interleaving views* of the same computation.) This is only possible if e_1 and e_2 are causally independent, and therefore, e_1 and e_2 are considered simultaneous. Nevertheless, to be able to capture all possible observations in this case, all consistent cuts $\varnothing$, $\{e_1\}$, $\{e_2\}$ and $\{e_1,e_2\}$ should be present. (Observation of the sequence e_1e_2 corresponds to the sequence of consistent cuts $\varnothing$, $\{e_1\}$ and $\{e_1,e_2\}$, while the sequence e_2e_1 corresponds to the sequence $\varnothing$, $\{e_2\}$ and $\{e_1,e_2\}$.) When ordered by set inclusion, the smallest transitions consists of consistent cuts that differ by only one event. (Katz and Peled (1990) summarise this argument by noting that it cannot be proven that the distributed system did not actually go through a state in which only e_1 (or e_2) has occurred.)

However, as the definition of the next global state relation indicates, the semantic structure developed in this thesis sometimes allows global state transitions that consists of two local state transitions. (This is the case for sending and receiving global state transitions, where the state of a link changes as well as the state of another link or a component.) As each local state transition can be compared to the occurrence of an event, the semantic structure developed in this thesis seems to divert from the ordering of consistent cuts. In a sending or receiving global transition, however, the two state transitions are related in the sense that they are state changes occuring at the same location: either the domain side of a link and the domain itself, or the co-domain side of a link and the co-domain itself. Thus, the two transitions can be considered as one event, either the sending or receipt of information. Consequently, in the semantic structure developed in this thesis, the smallest global state transitions consist of only one event.

To conclude this short survey on modelling distributed systems, a few key differences between the atomic mutual exclusion view and the true concurrency view are summarised.

- In the atomic mutual exclusion view, the (global) state of a distributed system is predominant. Concurrency is identified with (nondeterministic) choice between interleavings of the concurrent actions, indistinguishable from choices made in the computations carried out by individual processors in the system. States are partially ordered, where the partial order represents choices that can be made by the system, resulting in different possible runs of the system, one for each linearisation of the partial order. Given a specific run, all observers of that run observe the same behaviour.

- In the true concurrency view, events are a predominant concept. Events are partially ordered, where the partial order represents causal dependence of the events. A concept of global state can be derived from the partial order of events. These global states are partially ordered (more precisely, they form a lattice). A partial order represents one specific run of a distributed system. Each linearisation of the partial order corresponds with one observation of the same run. For each choice made by computations carried out by individual processes, only one of the alternatives is represented in the partial order. Other alternatives are represented in different partial orders, one for each alternative[12].

7.2.2 An Event-Based Model of Concurrency

Section 7.1.1 defined a notion of global state that does not rely on the availability of global time. The notion of global state developed in Section 7.1.1 has interesting relationships with a similar notion originally proposed by Lamport (1978), and further developed by many authors as stated in Section 7.2.1. In fact, e.g., Section 7.1.1 frequently refers to Lamport's ideas as a motivation. In this subsection, Lamport's notion is surveyed and the relationships between the two notions of global state are developed. First, in Section 7.2.2.1, the equivalence is established between the notion of strict dependence as defined by Definition 7.5 and the notion of causality in event based approaches. Second, in Section 7.2.2.2, the accuracy of the notion of global state developed in Section 7.1.1 is established by an equivalence to a similar notion in an event-based approach.

7.2.2.1 Equivalence with an Event-Based Approach

As stated in Section 7.2.1, the starting point in Lamport's approach and similar approaches is a set of *events* or *state transitions,* which can be partitioned in subsets of events that occur in the same process. A similar notion can be defined within the semantic structure developed in this thesis. As indicated in Section 7.2.1, event-based approaches usually assume that events within a single, primitive process are linearly ordered. Therefore, in this section, all local component and link traces are assumed to be linear. Moreover, it is assumed that traces are proper. As local traces are assumed to be linear and proper, the sets $next_{LT}(v_{S,i})$ and $prev_{LT}(v_{S,i})$ each contain only one element. For ease of notation, this element itself is denoted

[12] True concurrency models always seem to be two-dimensional in the sense that there is a concept for the representation of choice (e.g., the conflict relation in event structures, different outgoing edges from places in Petri nets, *sets* of partial orders in ISTL (Katz & Peled, 1990), and one of the two matrix dimensions in Chu spaces (Pratt, 1995)) and a concept for the representation of the partial order of events (e.g., the partial order in event structures and ISTL, different outgoing edges from transitions in Petri nets, and the other matrix dimension in Chu spaces).

$next_{LT}(v_{S,i})$ or $prev_{LT}(v_{S,i})$. For each component or link with proper traces, a set of transitions is defined as follows.

Definition 7.13. (Transitions). *Let S be a component or link and let* LT_S *be a local component or link trace of S. The set* $\mathcal{T}_S$ *of* transitions *of S is defined as follows:* $\mathcal{T}_S=\{\langle v_{S,i};v_{S,j}\rangle \mid v_{S,i}=prev_{LT_S}(v_{S,j})\}$. *An element* $\langle v_{S,i};v_{S,j}\rangle$ *of* $\mathcal{T}_S$ *is denoted* $trans(v_{S,i},v_{S,j})$. *The set* $\mathcal{T}_{SH}$ *of all transitions for a structure hierarchy* $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ *is defined as* $\mathcal{T}_{SH}=\bigcup_{S\in Comp\cup Lnk}\mathcal{T}_S$

Given a structure hierarchy *SH*, the set $\mathcal{T}_{SH}$ is the set of state transitions or events that is assumed to be given as a starting point in Lamport's or similar approaches (in fact, in this section the terminology and notation of Charron-Bost *et al.* (1996) is used). In such approaches, it is also assumed that three kinds of events can be distinguished: send event, receive events and internal events. As events are assumed to be atomic, an event is either a send event, a receive event, or an internal event. As stated above, events within a single process are assumed to be totally ordered (by a relation $\lhd_l$).

Approaches such as Lamport's (1986) informally assume that each send event corresponds with a receive event and vice versa. (Thus, no information is lost and no information is received without being sent.) A relation Γ is assumed to be given which relates send events to their 'corresponding receive events' (Charron-Bost *et al.*, 1996). Thus, if C_i and C_j are the sets of events of processes *i* and *j* respectively, then $\Gamma\subseteq C_i\times C_j$ such that $\langle s;r\rangle\in\Gamma$ denotes that *s* corresponds to *r.* Mattern (1992, Def. 1.4, clause 2) makes the assumption that each send event corresponds with a receive event and vice versa formally explicit by the requirement that Γ is left-unique and right unique. (A binary relation *R* is right-unique if for all *x,y,z*, if *xRy* and *xRz*, then *y=z*. A binary relation *R* is left-unique, or functional, if for all *x,y,z*, if *yRx* and *zRx*, then *y=z*.)

The notions developed in Chapter 5 and Chapter 6 can be used to provide specific $\lhd_l$ and Γ, which are assumed to be given in Lamport's or similar approaches. First, the correspondence relation Γ on transitions can be defined in terms of transmission octets. A transmission octet relates *states*—two states of the domain of a link, for states of the link itself, and two states of the co-domain of the link—that correspond similarly to send and receive transitions in Γ. More precisely, a transmission octet relates information that is present in the first of the two states of the domain with information received in the second of the two states of the co-domain. Second, the order $\lhd_l$ on transitions of the same process can be defined in terms of the next-state relation. Formally:

Definition 7.14. (Local order and correspondence for transitions). *Let* $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ *be a structure hierarchy and let* μ *be a multitrace for SH.*

- *The* local order relation *for* μ *is the relation* $\lhd_l$ *defined as follows:*

$\lhd_l = \{\langle trans(v_{A,i}, next_{\mu(A)}(v_{A,i})); trans(prev_{\mu(A)}(v_{A,j'}), v_{A,j'})\rangle \mid next_{\mu(A)}(v_{A,i}) = prev_{\mu(A)}(v_{A,j'})\}$

- *The* correspondence relation *for μ is the relation Γ defined as follows:*

  $\Gamma = \{\langle trans(v_{A,i}, next_{\mu(A)}(v_{A,i})); trans(prev_{\mu(B)}(v_{B,j'}), v_{B,j'})\rangle \mid$ *there is a link L with dom(L)=A and cdom(L)=B and there are i'',j'',k,l in the time frame of μ(L) such that* $\langle\langle v_{A,i}; next_{\mu(A)}(v_{A,i})\rangle; \langle v_{L,i''}; v_{L,j''}; v_{L,k}; v_{L,l}\rangle; \langle prev_{\mu(B)}(v_{B,j'}); v_{B,j'}\rangle\rangle$ *is a transmission octet}*

Given a set *C* of events, a local order relation $\lhd_l$, and a correspondence relation Γ, a binary relation, called 'happens before' (Lamport) or 'causally depends' (e.g. Schwarz & Mattern, 1994) is defined as follows. (The definition, as well as the definition of Γ, is taken literally (modulo notation) from (Charron-Bost *et al.*, 1996)):

Definition 7.15. (Causality (Charron-Bost *et al.*, 1996)). *The causality relation $\lhd$ in C is the smallest relation that satisfies the following three properties:*

1. *If $a \lhd_l b$, then $a \lhd b$;*
2. *If $\langle s;r\rangle \in \Gamma$, then $s \lhd r$;*
3. *If $a \lhd b$ and $b \lhd c$, then $a \lhd c$.*

This notion of causality can be further refined by additional constraints to model properties of information transmission. (This is the topic of the paper by Charron-Bost *et al.* (1996)).

The definition of (Charron-Bost *et al.*, 1996) is directly applicable to the semantic structure developed in this thesis. Given a structure hierarchy *SH* and a multitrace μ, take $\mathcal{T}_{SH}$ as defined in Definition 7.13 above for the set of events *C* and take the local order relation and the correspondence relation as defined in Definition 7.14 for $\lhd_l$ and Γ, respectively. In this chapter, the relation $\lhd$ defined in Definition 7.15 is written as $\rightarrow_{tr}$ and called *dependence for transitions.*

In the rest of this subsection, a formal relation is established between causality for transitions and strict dependence. Below, a proposition is presented which states that, under specific assumptions, a transition from $v_{A,i}$ to its immediate successor state is causally related to a transition from the immediate predecessor of $v_{B,j'}$ to $v_{B,j'}$ if and only if $v_{B,j'}$ strictly depends on $v_{A,i}$, and $v_{B,j'}$ is not itself the immediate successor of $v_{A,i}$. Thus, this proposition precisely relates the notion of strict dependence to the well-known notions first developed by Lamport.

However, the proposition that relates strict dependence to causality for transitions only holds for multitraces that fulfil a number of specific assumptions. These assumptions are mentioned at the beginning of this subsection: each transition is either a send transition, a receive transition or an internal transition, and Γ is left-unique and right-unique. These assumptions are derived from a more general assumption, which states that transitions or events are atomic: they do not consist of sub-events. Mattern (1992) defines a 'computation' as a partially ordered

set of events for which these assumptions hold. In this thesis a notion of atomic computation is defined as follows:

Definition 7.16. (Atomic computation). *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy, let μ be a multitrace for SH, let $A\in Comp$ be a component and let i be a point in the time frame of $\mu(A)$.*

- *The transition $trans(\nu_{A,i},next_{\mu(A)}(\nu_{A,i}))$ is a* send transition *if there is a link $L\in Lnk$ such that $dom(L)=A$ and there are i'',j'',k,l and j' in the time frames of $\mu(L)$ and $\mu(cdom(L))$, respectively, such that $\langle\langle\nu_{A,i},next_{\mu(A)}(\nu_{A,i})\rangle;\langle\nu_{L,i''};\nu_{L,j''};\nu_{L,k};\nu_{L,l}\rangle;\langle prev_{\mu(cdom(L))}(\nu_{cdom(L),j'});\nu_{cdom(L),j'}\rangle\rangle$ is a transmission octet.*
- *The transition $trans(\nu_{A,i},next_{\mu(A)}(\nu_{A,i}))$ is a* receive transition *if there is a link $L\in Lnk$ such that $cdom(L)=A$ and there are i'',j'',k,l and i' in the time frames of $\mu(L)$ and $\mu(dom(L))$, respectively, such that $\langle\langle\nu_{dom(L),i'},next_{\mu(dom(L))}(\nu_{dom(L),i})\rangle;\langle\nu_{L,i''};\nu_{L,j''};\nu_{L,k};\nu_{L,l}\rangle;\langle\nu_{A,i},next_{\mu(A)}(\nu_{A,i})\rangle\rangle$ is a transmission octet.*
- *Otherwise, the transition $trans(\nu_{A,i},next_{\mu(A)}(\nu_{A,i}))$ is an* internal transition.

The multitrace μ is called an atomic computation *if for all $S\in Comp\cup Lnk$ and for all $i\in T_S$ with $\mu_S=\langle\langle T_S;\prec_S\rangle;V_S\rangle$:*

- *$trans(\nu_{A,i},next_{\mu(A)}(\nu_{A,i}))$ is either a send transition, a receive transition or an internal transition,*
- *and the correspondence relation for μ is left-unique and right-unique.*

Left-uniqueness and right-uniqueness of the correspondence relation ensures that each pair of states in the domain of a link in which information is present that is transmitted via that link corresponds to exactly one pair of states in the co-domain in which the information is received. The importance of left-uniqueness and right-uniqueness is further discussed below. The stage is now set to present Proposition 7.17, which states that a transition from $\nu_{A,i}$ to its immediate successor state is causally related to a transition from the immediate predecessor of $\nu_{B,j'}$ to $\nu_{B,j'}$ if and only if $\nu_{B,j'}$ strictly depends on $\nu_{A,i}$ and $\nu_{B,j'}$ is not itself the immediate successor of $\nu_{A,i}$:

Proposition 7.17. *Let SH be a structure hierarchy and let μ be an atomic computation for SH. Then:*

$$t_1=trans(\nu_{A,i},next_{\mu(A)}(\nu_{A,i}))\rightarrow_{tr} trans(prev_{\mu(B)}(\nu_{B,j'}),\nu_{B,j'})=t_2$$
$$\Leftrightarrow\quad \nu_{A,i}\rightarrow_{sd}\nu_{B,j'} \text{ and } \nu_{B,j'}\neq next_{\mu(A)}(\nu_{A,i}).$$

Proposition 7.17 is illustrated in Figure 7.3, where the filled rectangles represent the transmission octet $\langle\langle\nu_{A,i};\nu_{A,j}\rangle;\langle\nu_{L,i''};\nu_{L,j''};\nu_{L,k};\nu_{L,l}\rangle;\langle\nu_{B,i'};\nu_{B,j'}\rangle\rangle$. In this figure, the thick arrow labelled *tr* depicts the causality relation for the transitions $trans(\nu_{A,i},next_{LT_A}(\nu_{A,i}))$ and $trans(prev_{LT_B}(\nu_{B,j'}),\nu_{B,j'})$. The thick arrow labelled *sd*

depicts the corresponding strict dependence relation between the states $\nu_{A,i}$ and $\nu_{B,j'}$. Thus, Proposition 7.17 shows the relation between the notion of dependence as presented in this thesis and the notion of causality for events or transitions as defined in (Charron-Bost *et al.*, 1996).

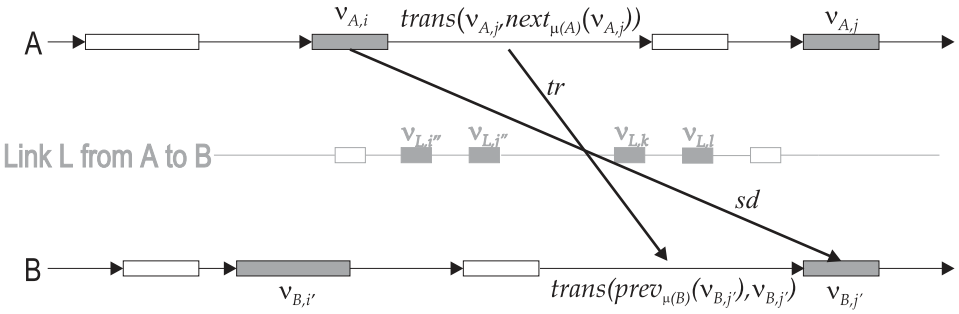

Figure 7.3: Strict precedence and causality for transitions.

Proposition 7.17 reveals a difference between strict dependence and causality for transitions: strict dependence is more fine-grained than causality for transitions. Both strict dependence and causality for transitions are irreflexive. Therefore, in terms of states, strict dependence relates two states, while causality for transitions relates at least three states (begin and end states of transitions related by causality for transitions). In other words, if there are only two different states, there is at most one transition, which cannot be related with itself. For this reason, Proposition 7.17 requires that $\nu_{B,j'} \neq next_{\mu(A)}(\nu_{A,i})$.

Proposition 7.17 also requires that μ is an atomic computation. In the semantic structure presented in Chapter 5 and Chapter 6, sets of states for all components and links in a structure hierarchy are assumed to be given. No notion of 'atomic state change' or operation is assumed. Consequently, it may be the case that in a local component or link trace, new states are only distinguished after several activities (internal and information exchange) have taken place. In this case, the relation Γ as defined in Definition 7.16 need not be left-unique and right-unique: during a transition from $\nu_{A,i}$ to its immediate successor state, information may be received from another component C_1 and information may be send to yet another component C_2, and so forth. In this case, the transition is related by Γ to more than one other transition. However, in this case, strict dependence need not be transitive when causality for transitions is. As a consequence, strict dependence is not similar to causality for transitions. This can be made more precise as follows. In Figure 7.4, it holds that $t_1 = trans(\nu_{A,i}, next_{\mu(A)}(\nu_{A,i})) \rightarrow_{tr} trans(\nu_{C,z'}, next_{\mu(C)}(\nu_{C,z'})) = t_4$, $t_3 = trans(\nu_{C,z}, next_{\mu(C)}(\nu_{C,z'})) \rightarrow_{tr} trans(prev_{\mu(B)}(\nu_{B,j'}), \nu_{B,j'})) = t_2$ and $t_4 \rightarrow_{tr} t_3$. By transitivity, $t_1 \rightarrow_{tr} t_2$. Because $t_1 \rightarrow_{tr} t_2$, Proposition 7.17 suggests that $\nu_{A,i} \rightarrow_{sd} \nu_{B,j'}$. If μ is an atomic computation, than this is indeed the case. However, if μ is not an

atomic computation, then this is not the case. This is explained as follows. By Proposition 7.17, $v_{A,i} \rightarrow_{sd} next_{\mu(C)}(v_{C,z'})$ and $v_{C,z} \rightarrow_{sd} v_{B,j'}$. Two cases are distinguished:

- If μ is not an atomic computation, it cannot be established that $v_{A,i} \rightarrow_{sd} v_{B,j'}$. In this case, it may be that $t_4=t_3$: the transition from $v_{C,z'}$ to $next_{\mu(C)}(v_{C,z'})$ is both a receive transition (information received from A) and a send transmission (information send to B). In this case, Γ is not left and right unique: t_4 is related by Γ to both t_1 and t_2. In Figure 7.4, the curved arrow from t_4 to t_3 would go directly to t_2 in this case, and the dashed arrow from $v_{C,z}$ to $v_{B,j'}$ would go from $v_{C,z'}$ to $v_{B,j'}$. It is possible to establish $v_{A,i} \rightarrow_{sd} v_{B,j'}$, via transitivity if $next_{\mu(C)}(v_{C,z'}) \rightarrow_{sd} v_{C,z}$, but this cannot be the case as $v_{C,z}=v_{C,z'}$.
- If μ is an atomic computation, then $v_{A,i} \rightarrow_{sd} v_{B,j'}$ in Figure 7.4. Because μ is an atomic computation, t_3 is strictly later than t_4. Therefore, $next_{\mu(C)}(v_{C,z'}) \rightarrow_{sd} v_{C,z}$, or $next_{\mu(C)}(v_{C,z'})=v_{C,z}$ (as is the case in Figure 7.4). By transitivity of $\rightarrow_{sd}$, $v_{A,i} \rightarrow_{sd} v_{B,j'}$.

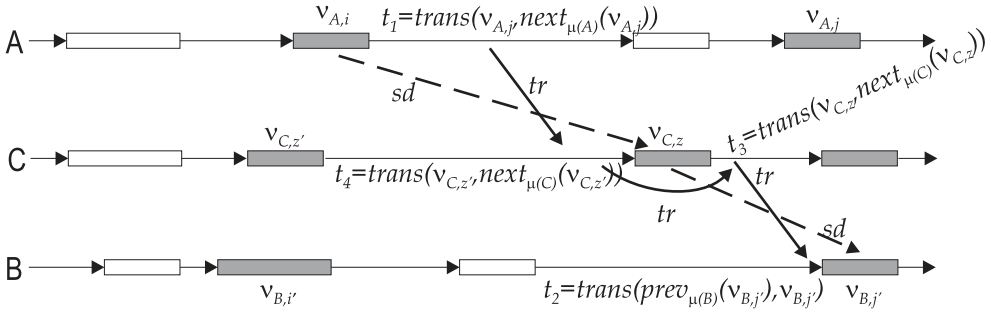

Figure 7.4: Atomic computation and transitivity of $\rightarrow_{tr}$ and $\rightarrow_{sd}$.

To finalise the comparison of the notion of strict dependence with causality for transitions, a different perspective is provided by *defining* strict dependence in terms of a notion of causality for transitions. This is done in (Fromentin & Raynal, 1994) as follows:

Definition 7.18. (Causality for states (Fromentin & Raynal, 1994)). *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy, let $\gamma=(\gamma_l)_{l \in Lnk}$ be a collection of compatibility relations, let μ be a multitrace compatible for γ, and let $v_{A,i}$ and $v_{B,j'}$ be two local component states of components or links $A,B \in Comp \cup Lnk$. The* causality for states *relation $\rightarrow_{fr} \subseteq S_{SH} \times S_{SH}$ is defined as follows: $v_{A,i} \rightarrow_{fr} v_{B,j'}$ iff:*

1. *$A=B$ and $v_{A,i} \in prev_{\mu(A)}(v_{B,j'})$, or*
2. *$trans(v_{A,i}, next_{\mu(A)}(v_{A,i})) \rightarrow_{tr} trans(prev_{\mu(B)}(v_{B,j'}), v_{B,j'})$.*

The notion of causality as defined by Fromentin & Raynal (1994) is equivalent with the notion of strict dependence developed in this thesis:

Proposition 7.19**.** *Let SH be a structure hierarchy and let* μ *be an atomic computation for SH. Then* $v_{A,i} \rightarrow_{sd} v_{B,j'}$ *iff* $v_{A,i} \rightarrow_{fr} v_{B,j'}$.

This proposition ends the formal comparison between the notion of (strict) dependence developed in this thesis and similar notions developed in the area of distributed systems research. In the next section, the notion of global state as defined in Section 7.1 is compared to a similar notion that is often employed in the area of distributed systems research.

7.2.2.2 Accuracy of Global States

The rest of this subsection proceeds as follows. In Section 7.1, for a structure hierarchy *SH*, a multitrace μ, and a collection of compatibility relations γ, the set of snapshots $SNAP(SH,\mu,\gamma)$ is defined in Definition 7.1. A specific subset $GS(SH,\mu,\gamma)$ of $SNAP(SH,\mu,\gamma)$ is defined in Definition 7.4 as the set of global states of the structure hierarchy *SH*. Likewise, a subset of *strict* global states $SGS(SH,\mu,\gamma)$ is defined in Definition 7.6. The current section evaluates the choice of these specific subsets by comparing them to a well-known notion in event based approaches, the notion of a *consistent cut*. Consistent cuts are a good approximation of the global state of a distributed system, as is argued in e.g. (Schwarz & Mattern, 1994). It can be proven that the set of consistent cuts (or, more precisely, the set of snapshots determined by consistent cuts) equals the set of strict global states as defined in Definition 7.6. Therefore, the notion of strict global state developed in Definition 7.4. is a good approximation of the global state of a distributed system as well. General (i.e., not necessarily strict) global states as defined in Definition 7.4 are difficult to compare with consistent cuts. As explained below, consistent cuts are defined in terms of transitions, which assume traces to be proper. With proper traces, strict dependence (and, consequently, strict global states) is a more appropriate notion to use.

Consistent cuts are defined without any reference to a notion of global time. Instead, they formalise the following view on the behaviour of a distributed system. In a distributed system, as a result of local activity the state of each component and link changes over (local) time, thus generating a sequence of state transitions. For the system as a whole, it is possible to partition the set of all state transitions of all components and links in two parts: one part consisting of, for every component and link, all transitions that, according to the local time of the component or link they belong to, have already happened and the other part consisting of the rest of the possible transitions. The part consisting of all transitions that have already happened is completely determined by the set of the last transitions of each component or link that has already happened. This set is

called a *cut* and is defined as follows (in this definition, a set $\mathcal{T}_S$ is the set of all transitions of a component or link S as defined in Definition 7.13):

Definition 7.20. (Cut). *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy, let $\gamma=(\gamma_l)_{l\in Lnk}$ be a collection of compatibility relations and let μ be a multitrace compatible for γ. A* cut *of SH for μ and γ is a function $cut(SH,\mu,\gamma)$: $Comp\cup Lnk\rightarrow\bigcup_{S\in Comp\cup Lnk}\mathcal{T}_S$ such that for all $S\in Comp\cup Lnk$: $cut(SH,\mu,\gamma)(S)\in\mathcal{T}_S$.*

However, not every cut of a structure hierarchy *SH* can occur in a computation, because a state transition can only occur if all state transitions on which it depends (in the sense of Definition 7.15) have also occurred. Cuts that fulfil this requirement are called *consistent cuts*. A consistent cut exactly represents the global state of a computation of the components and links in a structure hierarchy *SH*. The definition of consistency for cuts refers to the transitive closure of a cut, which is defined below. (The transitive closure of a cut resembles the transitive closure of a relation. However, as a cut is not a relation, it is not exactly the same.) The transitive closure is exactly the set of all state transitions that have already happened (as in Section 7.2.2.1, traces are assumed to be linear):

Definition 7.21. (Consistent cut). *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy, let $\gamma=(\gamma_l)_{l\in Lnk}$ be a collection of compatibility relations, let μ be a multitrace compatible for γ, and let $cut(SH,\mu,\gamma)$ be a cut of SH for μ and γ.*

- *For a component or link S, the* transitive closure *$(cut(SH,\mu,\gamma)(S))^*$ of $cut(SH,\mu,\gamma)(S)=trans(v_{S,i},next_{\mu(S)}(v_{S,i}))$ is the set $\{trans(v_{S,j},next(v_{S,j}))\mid \bot\le j\le i\}$.*
- *The* transitive closure *of $cut(SH,\mu,\gamma)$ is a function $cut^*(SH,\mu,\gamma)$ on $Comp\cup Lnk$ such that for all $S\in Comp\cup Lnk$ it holds that $cut^*(SH,\mu,\gamma)(S)=cut^*(S)$. If there is a component or link $S\in Comp\cup Lnk$ such that for a certain transition t it holds that $t\in cut^*(SH)(S)$, then this is written as $t\in cut^*(SH,\mu,\gamma)$.*
- *A cut is* consistent *iff for all $S\in Comp\cup Lnk$ and for all $t'\in\mathcal{T}_{SH}$: if $t'\rightarrow_{tr}cut(SH,\mu,\gamma)(S)$, then $t'\in cut^*(SH,\mu,\gamma)$. The set of all consistent cuts of SH with respect to μ and γ is denoted $CUT(SH,\mu,\gamma)$.*

Theorem 7.22. (Mattern, 1992). *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy, let $\gamma=(\gamma_l)_{l\in Lnk}$ be a collection of compatibility relations and let μ be a multitrace compatible for γ. The triple $\langle CUT(SH,\mu,\gamma);\cap;\cup\rangle$ is a lattice ordered by $\subseteq$.*

The proof presented in (Mattern, 1992) is straightforward. To show that $\langle CUT(SH);\cap;\cup\rangle$ is a lattice, it suffices to show that it is closed under $\cap$ and $\cup$, because *CUT(SH)* is a subset of the powerset of $\mathcal{T}_{SH}$, and the powerset of any set is a lattice itself under $\subseteq$. To show that it is ordered by $\subseteq$, it is sufficient to show that $C_1\subseteq C_2$ iff $C_1\cap C_2=C_1$.

A computation, or run, of the components and links in a structure hierarchy SH is modelled as a sequence of consistent cuts such that the transitive closure of each cut in this sequence is a biggest subset of its successor:

Definition 7.23. (Run). *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy, let $\gamma=(\gamma_I)_{I\in Lnk}$ be a collection of compatibility relations, and let μ be a multitrace compatible for γ. A* run *of SH with respect to μ and γ is a sequence $(cut(SH,\mu,\gamma)_n)_{n\in N}$ of consistent cuts such that for all $0<i\leq n$: $\bigcup_{S\in Comp\cup Lnk}(cut^*(SH,\mu,\gamma)_{i-1}(S))\subset\bigcup_{S\in Comp\cup Lnk}(cut^*(SH,\mu,\gamma)_i(S))$ and $|\bigcup_{S\in Comp\cup Lnk}(cut^*(SH,\mu,\gamma)_i(S))|=i$.*

The notion of a run of a system as defined above is, in a sense, the most precise view of a system that can be obtained: it is impossible to prove that during a run, the system did not go through the sequence of consistent cuts as defined above. Therefore, a consistent cut represents a global state of the system. As stated in the previous section, the notion of a run, although defined in a somewhat different manner, occurs in many approaches to modelling true concurrency, such as for instance (Winskel, 1989).

It is possible to associate with each cut a set of states consisting of, for each component, the state obtained by the state transition for that component in the cut:

Definition 7.24. (Final states). *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy, let $\gamma=(\gamma_I)_{I\in Lnk}$ be a collection of compatibility relations, and let μ be a multitrace compatible for γ. The* final states *of a cut $cut(SH,\mu,\gamma)$ is the function $fs(cut(SH,\mu,\gamma)): Comp\cup Lnk\rightarrow\bigcup_{S\in Comp\cup Lnk}S_S$ such that for all $S\in Comp\cup Lnk$: if $cut(SH,\mu,\gamma)(S)=trans(\nu_{S,i},next_{\mu(S)}(\nu_{S,i}))$, then $fs(cut(SH,\mu,\gamma)(S))=next_{\mu(S)}(\nu_{S,i})$.*

As is witnessed by the following theorem, if a cut is consistent, then its associated set of final states is a global state as defined in Definition 7.4. Therefore, a global state as defined in Definition 7.4 is a good approximation of the notion of a global state of a distributed system.

Proposition 7.25. *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy, let $\gamma=(\gamma_I)_{I\in Lnk}$ be a collection of compatibility relations, let μ be a multitrace compatible for γ and let $cut(SH,\mu,\gamma)$ be a cut for SH with respect to μ and γ. If $cut(SH,\mu,\gamma)$ is a consistent cut, then $fs(cut(SH,\mu,\gamma))$ is a global state.*

Moreover, only consistent snapshots represent global states, which is witnessed by the next theorem. This theorem states that the set consisting of, for each component and link, the transition that leads to the state occurring in a consistent snapshot is a consistent cut.

Definition 7.26. (Final transition set). *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy, let $\gamma=(\gamma_I)_{I\in Lnk}$ be a collection of compatibility relations, let μ be a multitrace compatible for γ, and let $\sigma\in SNAP(SH,\mu,\gamma)$ be a snapshot of SH for μ and γ. The* final

transition set *of* σ *is the function* $fts(\sigma)$*:* $Comp \cup Lnk \rightarrow \bigcup_{S \in Comp \cup Lnk} \mathcal{T}_S$ *such that for all* $S \in Comp \cup Lnk$*: if* $\sigma(S) = \nu_{S,i}$*, then* $fts(\sigma)(S) = trans(prev_{\mu(S)}, \nu_{S,i})$*.*

Proposition 7.27. *Let* $SH = \langle Comp; Lnk; \prec; dom; cdom \rangle$ *be a structure hierarchy, let* $\gamma = (\gamma_l)_{l \in Lnk}$ *be a collection of compatibility relations, let* μ *be a multitrace compatible for* γ *and let* $\sigma \in SNAP(SH, \mu, \gamma)$ *be a snapshot of SH for* μ *and* γ*. If* σ *is a strict global state of a structure hierarchy SH, then* $fts(\sigma)$ *is a consistent cut.*

This section on the accuracy of the notion of global states defined in this thesis concludes with some remarks on the lattice structure of global states. As shown by Proposition 7.8, $\langle PART(SGS, \mu, \gamma); next_{SGS(SH,\mu,\gamma)} \rangle$ is a lattice. According to Theorem 7.22, the pair $\langle CUT(SH, \mu, \gamma); \subseteq \rangle$ is also a lattice. These two lattices are expected to be isomorphic, as is shown by the following conjecture;

Conjecture 7.28. *Let* $SH = \langle Comp; Lnk; \prec; dom; cdom \rangle$ *be a structure hierarchy, let* $\gamma = (\gamma_l)_{l \in Lnk}$ *be a collection of compatibility relations, and let* μ *be a multitrace compatible for* γ*. The lattice* $\langle PART(SGS, \mu, \gamma); next_{SGS(SH,\mu,\gamma)} \rangle$ *is isomorphic to* $\langle CUT(SH, \mu, \gamma); \subseteq \rangle$*.*

This conjecture states that it is possible to use state-based and event-based representations interchangeably. A possible way to prove this conjecture may be as follows. First, $\langle CUT(SH, \mu, \gamma); \subseteq \rangle$ is proven to be a distributive lattice, which is possibly infinite. Second, $PART(SH, \mu, \gamma)$ must be shown to represent order-ideals with respect to $\rightarrow_{sd}$ of $CUT(SH, \mu, \gamma)$. In the finite case, Birkhoff's (1933) representation theorem for finite distributive lattices can be applied to complete the proof. (This theorem states that each lattice is isomorphic to the lattice of its prime ideals ordered by subset inclusion.) For the infinite case, algebra does not seem to suffice. Instead, a topological structure proposed by Stone (1936) can be imposed on the set of prime ideals of a lattice. With the help of this construction, Stone (1936) established a representation theorem for boolean algebras (which are a special class of distibutive lattices). Priestley (1970) combined the results of Birkhoff and Stone, resulting in a general representation theorem for distributive lattices. (See also (Davey & Priestley, 1990, Th. 10.18).) His theorem states that a lattice is isomorphic to the set of closed-open subsets of a compact totally-order disconnected space. (Which is a set bearing Stone's topology.) Thus, to apply Priestley's result, $\langle PART(SGS, \mu, \gamma); next_{SGS(SH,\mu,\gamma)} \rangle$ must be proven to consist of such closed-open subsets. The use of Priestley's result in the area of models of distributed computation is suggested by Pratt (1992). Pratt refers to Priestley's result as the Birkhoff-Stone duality.

7.2.3 A Generalisation of Strict Dependence

Section 7.1.1 discussed differences between strict dependence and general (i.e., non-strict) dependence. It is possible to generalise the notion of strict dependence in order to resemble the notion of dependence. This is done by adapting the second clause of the definition such that $\nu_{B,j'}$ depends on $\nu_{A,i}$ if there is a transmission

octet with $v_{A,i}$ and $v_{A,j}$ as first and second elements, where $i<j$ and there is a state $v_{A,k}$ such that $i<k\leq j$ and $v_{A,k}\in next_{LT_A}(v_{A,i})$, and similar for the last two elements of the transmission octet. The difference with the non-generalised notion of strict dependence is that $v_{A,j}$ is not itself required to be an element of $next_{LT_A}(v_{A,i})$ (and likewise for $v_{B,i'}$). The generalised version of strict dependence is formally defined as follows:

Definition 7.29. (Generalised strict dependence). *Let $SH=(Comp;Lnk;\prec;dom;cdom)$ be a structure hierarchy and let $LT_A=(\langle T_A;<_A\rangle;V_A)$, $LT_B=(\langle T_B;<_B\rangle;V_B)$ and $LT_L=(\langle T_L;<_L\rangle;V_L)$ be three traces of components or links $A,B\in Comp\cup Lnk$ and a link $L\in Lnk$ such that $A=dom(L)$ and $B=cdom(L)$. Let $v_{A,i}$ and $v_{A,j}$ be two states in LT_A, let $v_{L,i''}$, $v_{L,j''}$, $v_{L,k}$ and $v_{L,l}$ be four states in LT_L, and let $v_{B,i'}$ and $v_{B,j'}$ be two states in LT_B. The* generalised strict dependence *relation for LT_A, LT_L and LT_B is a binary relation $\rightarrow_{gsd}$ on $\mathcal{S}_{SH}$ defined as the smallest relation $\rightarrow_{gsd}$ such that $v_{A,i}\rightarrow_{gsd} v_{B,j'}$ iff either*

1. *$A=B$ and $v_{B,j'}\in next_{LT_A}(v_{A,i})$, or*
2. *$\langle(v_{A,i};v_{A,j});(v_{L,i''};v_{L,j''};v_{L,k};v_{L,l});(v_{B,i'});v_{B,j'}\rangle\rangle$ is a transmission octet for LT_A, LT_L and LT_B, and there is a state $v_{A,k}\in next_{LT_A}(v_{A,i})$ with $k\leq j$ and there is a state $v_{B,k'}\in prev_{LT_B}(v_{B,j})$ with $i'\leq k'$ or*
3. *There is a state $v_{C,m}$ of a component or link $C\in Comp\cup Lnk$ such that $v_{A,i}\rightarrow_{gsd} v_{C,m}$ and $v_{C,m}\rightarrow_{gsd} v_{B,j'}$.*

The rest of this chapter mainly focusses on (formal, detailed) comparisons between the notion of global state developed in this chapter and similar notions found in Computer Science literature. As (non-generalised) strict dependence fits better to assumptions made for these similar notions, in this chapter non-generalised strict dependence is used.

7.3 Proofs

Before presenting, in Section 7.3.2, proofs of the properties and theorems presented in this chapter, first some properties of transitive relations used in these proofs are developed in Section 7.3.1.

7.3.1 A Note on Transitive Relations

The third (inductive) clause in Definition **6.11** (the dependence relation) is the source of complications in a number of proofs of properties of this relation. This clause seems to naturally correspond to inductive proofs. This turns out to be true, however, it is somewhat involved to find an isomorphism with a discretely ordered set on which to base the induction. Consider a proof in which one tries to prove a property for all $v_{A,i}$ and $v_{B,j'}$ such that $v_{A,i}\rightarrow_{sd} v_{B,j'}$. At first sight, chains from $v_{A,i}$ to $v_{B,j'}$ (i.e, totally ordered subsets of $\mathcal{S}_{SH}$ with $v_{A,i}$ as minimal and $v_{B,j'}$ as

maximal element) seem to be suitable candidates. Under certain circumstances, it is always the case that if $\nu_{A,i} \rightarrow_{sd} \nu_{B,j'}$, there is a *maximal* chain from $\nu_{A,i}$ to $\nu_{B,j'}$ (i.e., a chain from $\nu_{A,i}$ to $\nu_{B,j'}$ such that all other chains from $\nu_{A,i}$ to $\nu_{B,j'}$ have fewer elements). The property can then be proven by induction to the length of this maximal chain. However, if the order on $\mathcal{S}_{SH}$ is dense, which is not prohibited, such a maximal chain might have an uncountable number of elements, which implies that induction is not applicable.

There are, however, other candidates. One possibility is to define the dependence relation as the least fixed point of a recursively defined transitive closure operator. Below it is proven that this definition is equivalent to Definition 6.11. Properties of the dependence relation can then be proven by induction to the number of times the closure operator is applied before the fixed point is reached. As proven below, this is countable number of times, even if the order on $\mathcal{S}_{SH}$ is dense. The proof uses some concepts from the theory of partial orders, which are defined below. (In the following definitions, *sup(D)* denotes the *least upper bound*, or *supremum*, of set *D*, if the supremum exists. A *fixed point*, or *fixpoint* of a function $\varphi: P \rightarrow Q$ is an x in P such that $\varphi(x)=x$. If a smallest x such that $\varphi(x)=x$ exists, it is denoted $lfp(\varphi)$, for *least fixed point* of φ.)

Definition 7.30**.** (Davey & Priestley, 1992, p. 51) *Let $\langle P;<\rangle$ be a partial order.*

- *A non-empty subset $S \subseteq P$ is* directed *iff, for every finite subset $F \subseteq S$, there exists $z \in S$ such that z is an upper bound of F;*
- *The partial order P is a* complete partial order (CPO) *iff P has a smallest element and sup(D) exists for each directed subset of $D \subseteq P$;*
- *Let $\langle Q;<'\rangle$ be a partial order. A function $\varphi: P \rightarrow Q$ is* order-preserving *(or, alternatively,* monotone*) if for all $x,y \in P$, $x<y$ implies $\varphi(x)<'\varphi(y)$;*
- *Let Q be a CPO. A function $\varphi: P \rightarrow Q$ is* continuous *if, for every directed set $D \subseteq P$, $\varphi(sup(D))=sup(\{\varphi(x) \mid x \in D\})$.*

For any set X, the partial order $\langle \mathcal{P}(X);\subseteq\rangle$ is a complete lattice and therefore a complete partial order. In this partial order, $\varnothing$ is the bottom element and for arbitrary subset $S=\{A_i \mid i \in I\} \subseteq \mathcal{P}(X)$, $sup(S)=\bigcup_{i \in I} A_i$.

The following theorem establishes the existence of a least fixpoint $lfp(\Phi)$ for an order-preserving map Φ. The set of all (not necessarily least) fixpoints of Φ is denoted $fix(\Phi)$.

Theorem 7.31**.** (CPO Fixpoint Theorem I, adapted from (Davey & Priestley, 1992)). *Let P be a CPO, let $\Phi: P \rightarrow P$ be an order-preserving map and define $\alpha := \sup_{n \geq 0} \Phi^n(\bot)$.*

1. *If $\alpha \in fix(\Phi)$, then $lfp(\Phi)$ exists and equals α.*
2. *If Φ is continuous then $lfp(\Phi)$ exists and equals α.*

Proof. See (Davey & Priestley, 1992, p. 89).

The transitive closure of a relation $<$ can now be defined as follows, where Φ^n is defined as: Φ^0 is the identity map, and $\Phi^m=\Phi(\Phi^{m-1})$ for $m>0$.

Definition 7.32. (Transitive closure). *Let P be a set and let $<$ be a binary relation on P.*

- *Define a function $\Phi_<: \mathcal{A}(P\times P)\to\mathcal{A}(P\times P)$ such that*
  $\Phi_<(X)=\{\langle x;y\rangle \mid x<y\} \cup X \cup \{\langle x;z\rangle \mid \langle x;y\rangle,\langle y;z\rangle\in X\}$;
- *The* transitive closure $<^*$ *of* $<$ *is the relation* $<^*=lfp(\Phi_<)=\bigcup_{n\geq 0}\Phi^n_<(\varnothing)$.

The co-domain of $\Phi_<$ (namely $\mathcal{A}(P\times P)$), is a lattice of sets. In a lattice of sets, $sup(D)=\cup D$ for any $D\subseteq\mathcal{A}(P\times P)$. Therefore, $\alpha:=\sup_{n\geq 0}\Phi^n(\bot)=\bigcup_{n\geq 0}\Phi^n_<(\varnothing)$. The transitive closure of $<$ is well-defined because $\Phi_<$ is order-preserving and continuous, as is proven in the next two propositions. After that, it is proven that $<^*$ is the smallest transitive relation such that $<\subseteq<^*$.

Proposition 7.33. *$\Phi_<$ is order-preserving.*

Proof. Proof sketch: it has to be proven that for $X\subseteq Y$, $\Phi_<(X)\subseteq\Phi_<(Y)$. This is done by proving that all three cases given by Definition 7.32 for $\langle x;y\rangle\in\Phi_<(X)$ imply that $\langle x;y\rangle\in\Phi_<(Y)$.
Assume: 1. $X,Y\in\mathcal{A}(P\times P)$ such that $X\subseteq Y$, 2. $x,y\in P$ such that $\langle x;y\rangle\in\Phi_<(X)$.
Prove: $\langle x;y\rangle\in\Phi_<(Y)$.

⟨1⟩1. Case: x<y.
Proof: by assumption ⟨0⟩:1, $\langle x;y\rangle\in Y$. By the definition of $\Phi_<(Y)$, $\langle x;y\rangle\in\Phi_<(Y)$.

⟨1⟩2. Case: $\langle x;y\rangle\in X$.
Proof: $\langle x;y\rangle\in Y$ by assumption ⟨0⟩:1, $\langle x;y\rangle\in\Phi_<(Y)$ by definition of $\Phi_<(Y)$.

⟨1⟩3. Case: $\langle x;y\rangle\in\{\langle x';y'\rangle \mid \langle x';z\rangle,\langle z;y'\rangle\in X\}$.
⟨2⟩1. $\exists z\in P$: $\langle x;z\rangle\in X$ and $\langle z;y\rangle\in X$.
Proof: by assumption ⟨1⟩.
⟨2⟩2. $\exists z\in P$: $\langle x;z\rangle\in Y$ and $\langle z;y\rangle\in Y$.
Proof: by step ⟨2⟩1 and assumption ⟨0⟩:1.
⟨2⟩3. Q.E.D.
Proof: $\langle x;y\rangle\in\{\langle x';y'\rangle \mid \langle x';z\rangle,\langle z;y'\rangle\in Y\}\subseteq\Phi_<(Y)$ by step ⟨2⟩2 and definition of $\Phi_<(Y)$.

⟨1⟩4. Q.E.D.
Proof: steps ⟨1⟩1, ⟨1⟩2, and ⟨1⟩3 enumerate all three cases for $\langle x;y\rangle\in\Phi_<(X)$.

Proposition 7.34. *$\Phi_<$ is continuous.*

Proof. Proof sketch: by the definition of continuity, it has to be proven that for $D\subseteq\mathcal{A}(P\times P)$ directed, $\Phi_<$ preserves suprema, that is: $\Phi_<(sup(D))=sup(\{\Phi_<(X) \mid X\in D\})$. First, it is established that $sup(\{\Phi_<(X) \mid X\in D\})$ exists. After that, the equality is proven by proving that all three cases for $\langle x;y\rangle\in\Phi_<(sup(D))$ imply that $\langle x;y\rangle\in sup(\{\Phi_<(X) \mid X\in D\})$.

Assume: $D \subseteq \mathcal{P}(P \times P)$ is directed.
Prove: $\Phi_<(sup(D)) = sup(\{\Phi_<(X) \mid X \in D\})$.

⟨1⟩1. $\{\Phi_<(X) \mid X \in D\}$ is directed.
  Assume: $F \subseteq \{\Phi_<(X) \mid X \in D\}$ finite.
  Prove: there is a $Z \in \{\Phi_<(X) \mid X \in D\}$ such that Z is an upper bound of F.
  ⟨2⟩1. Let $F = \{\Phi_<(X_1),\ldots,\Phi_<(X_n)\}$ for finite subset $\{X_1,\ldots,X_n\}$ of D. Then there is a $Z' \in D$ such that Z' is an upper bound of $\{X_1,\ldots,X_n\}$.
    Proof: by directedness of D.
  ⟨2⟩2. $\forall X_i$ with $1 \leq i \leq n$, $X_i \subseteq Z'$.
    Proof: by step ⟨2⟩2 and the definition of an upper bound of $\{X_1,\ldots,X_n\}$.
  ⟨2⟩3. $\forall X_i$ with $1 \leq i \leq n$, $\Phi_<(X_i) \subseteq \Phi_<(Z')$.
    Proof: by the order-preserving property of $\Phi_<$ (Proposition 7.33) and step ⟨2⟩2.
  ⟨2⟩4. $\Phi_<(Z')$ is an upper bound of F.
    Proof: by step ⟨2⟩3 and the definition of an upper bound of $F = \{\Phi_<(X_1),\ldots,\Phi_<(X_n)\}$.
  ⟨2⟩5. Q.E.D.
    Proof: take $Z = \Phi_<(Z') \in \{\Phi_<(X) \mid X \in D\}$. By step ⟨2⟩4, Z is an upper bound of F.

⟨1⟩2. $sup(\{\Phi_<(X) \mid X \in D\})$ exists.
  Proof: step ⟨1⟩1, $\{\Phi_<(X) \mid X \in D\}$ is directed. By the remark following Definition 7.30, $\mathcal{P}(P \times P)$ is a CPO, and by definition of a CPO, every directed subset has a supremum.

⟨1⟩3. $\forall X \in D$, $\Phi_<(X) \subseteq sup(\{\Phi_<(X) \mid X \in D\})$.
  Proof: by step ⟨1⟩2 and the definition of supremum.

⟨1⟩4. $\Phi_<(sup(D)) \subseteq sup(\{\Phi_<(X) \mid X \in D\})$.
  Assume: $\langle x;y\rangle \in \Phi_<(sup(D)) = \{\langle x;y\rangle \mid x<y\} \cup sup(D) \cup \{\langle x;z\rangle \mid \langle x;y\rangle, \langle y;z\rangle \in sup(D)\}$.
  Prove: $\langle x;y\rangle \in sup(\{\Phi_<(X) \mid X \in D\})$.
  ⟨2⟩1. Case: $x<y$.
    Proof: $x<y$ implies $\langle x;y\rangle \in \Phi_<(X)$ for any $X \in D$ by definition of $\Phi_<(X)$. By step ⟨1⟩3, $\langle x;y\rangle \in sup(\{\Phi_<(X) \mid X \in D\})$
  ⟨2⟩2. Case: $\langle x;y\rangle \in sup(D)$.
    Proof: let X be an arbitrary element of D. Then $\langle x;y\rangle \in X$ by assumption ⟨1⟩. By the definition of $\Phi_<$, $X \subseteq \Phi_<(X)$. Thus, $\langle x;y\rangle \in \Phi_<(X)$. By step ⟨1⟩3, $\langle x;y\rangle \in sup(\{\Phi_<(X) \mid X \in D\})$.
  ⟨2⟩3. Case: $\langle x;y\rangle \in \{\langle x;z\rangle \mid \langle x;y\rangle, \langle y;z\rangle \in sup(D)\}$.
    ⟨3⟩1. There is a z such that $\langle x;z\rangle, \langle z;y\rangle \in sup(D)$
      Proof: by assumption ⟨2⟩.
    ⟨3⟩2. Let X be an arbitrary element of D. Then $\langle x;z\rangle, \langle z;y\rangle \in X$.
      Proof: by step ⟨3⟩1.
    ⟨3⟩3. $\langle x;y\rangle \in \Phi_<(X)$.

Proof: y the definition of $\Phi_<$.

⟨3⟩4. Q.E.D.
Proof: by step ⟨1⟩3, $\Phi_<(X) \subseteq sup(\{\Phi_<(X) \mid X \in D\})$.

⟨2⟩4. Q.E.D.
Proof: steps ⟨2⟩1, ⟨2⟩1 and ⟨2⟩1 enumerate all cases for $\langle x;y\rangle \in \Phi_<(sup(D))$.

⟨1⟩5. $sup(\{\Phi_<(X) \mid X \in D\}) \subseteq \Phi_<(sup(D))$.

⟨2⟩1. $\{\Phi_<(X) \mid X \in D\}$ is directed.
Proof: take $F \subseteq D$ finite. Because D is directed, there is a $Z \in D$ such that Z is an upper bound of F. Thus, for all $Z' \in F$, $Z' \subseteq Z$. By order-preservation, for all $Z' \in F$, $\Phi_<(Z') \subseteq \Phi_<(Z)$. Thus, $\Phi_<(Z)$ is an upper bound of $\{\Phi_<(X) \mid X \in D\}$, and therefore, $\{\Phi_<(X) \mid X \in D\}$ is directed.

⟨2⟩2. $\Phi_<(sup(D))$ is an upper bound of $\{\Phi_<(X) \mid X \in D\}$.
Proof: For all $X \in D$, $X \subseteq sup(D)$. By order-preservation, for all $X \in D$, $\Phi_<(X) \subseteq \Phi_<(sup(D))$.

⟨2⟩3. Q.E.D.
Proof: by step ⟨2⟩1, $\{\Phi_<(X) \mid X \in D\}$ is directed and therefore, $sup(\{\Phi_<(X) \mid X \in D\})$ exists. As $sup(\{\Phi_<(X) \mid X \in D\})$ is by definition the *least* upper bound of $\{\Phi_<(X) \mid X \in D\}$, $sup(\{\Phi_<(X) \mid X \in D\}) \subseteq \Phi_<(sup(D))$.

⟨1⟩6. Q.E.D.
Proof: by steps ⟨1⟩4 and ⟨1⟩5.

Proposition 7.35. *Let P be a set and let $<$ be a binary relation on P. The relation $<^* = lfp(\Phi_<) = \bigcup_{n \geq 0} \Phi^n_<(\emptyset)$ on P is the smallest transitive relation such that $< \subseteq <^*$.*

Proof. Proof sketch: first, it is established that $<^*$ is well defined by checking whether the least fixpoint of $\Phi_<$ exists and equals $\bigcup_{n \geq 0} \Phi^n_<(\emptyset)$. After that, it is proven that for $x,y \in P$ such that $x<y$, (i) $x<^*y$, (ii) $<^*$ is transitive and (iii) $<^*$ is the smallest transitive relation such that $< \subseteq <^*$. This is proven by assuming that there is a smaller transitive relation and deriving a contradiction from the fact that this smaller relation must be transitive.

Assume: $x,y \in P$ and $x<y$.

Prove: $x<^*y$, $<^*$ is transitive and $<^*$ is the smallest relation such that $< \subseteq <^*$.

⟨1⟩1. The least fixed point of $\Phi_<$ exists and equals $\bigcup_{n \geq 0} \Phi^n_<(\emptyset)$.

⟨2⟩1. The pair $\langle \mathcal{P}(P \times P); \subseteq \rangle$ is a complete partial order in which $\emptyset$ is the bottom element and for arbitrary subset S of $\mathcal{P}(P \times P)$, $sup(S) = \cup S$.
Proof: by the remark presented just after Definition 7.30.

⟨2⟩2. $\Phi_<$ is order-preserving and continuous.
Proof: by Propositions 7.33 and 7.34.

⟨2⟩3. Q.E.D.
Proof: by steps ⟨2⟩1 and ⟨2⟩2 and Theorem 7.31 (CPO Fixpoint Theorem I).

⟨1⟩2. $x<^{*}y$.
Proof: by the definition of $\Phi_{<}$, $\Phi^{1}_{<}(\emptyset)=\{\langle x;y\rangle \mid x<y\}$. Thus, $\langle x;y\rangle\in<^{*}$.

⟨1⟩3. For all $x,y,z\in P$, if $x<^{*}y$ and $y<^{*}z$, then $x<^{*}z$.
Assume: $x,y,z\in P$, $x<^{*}y$ and $y<^{*}z$.
Prove: $x<^{*}z$.

⟨2⟩1. There exists an m such that $\langle x;y\rangle,\langle y;z\rangle\in\Phi^{m}_{<}(\emptyset)$.
Proof: by assumption ⟨1⟩ and the definition of $<^{*}$.

⟨2⟩2. $\langle x;z\rangle\in\Phi^{m+1}_{<}(\emptyset)$.
Proof: by step ⟨2⟩1 and the definition of $\Phi_{<}$.

⟨2⟩3. Q.E.D.
Proof: by step ⟨2⟩2 and the definition of $<^{*}$.

⟨1⟩4. $<^{*}$ is the smallest relation such that $<\subseteq<^{*}$.
Assume: there is a transitive relation $<'\subset<^{*}$ such that $<\subseteq<'$.
Prove: false.

⟨2⟩1. $<'$ is not a fixpoint of $\Phi_{<}$.
Proof: $<^{*}$ is the least fixpoint of $\Phi_{<}$, so any relation that is smaller than $<^{*}$ cannot be a fixpoint of $\Phi_{<}$.

⟨2⟩2. $<'\neq\Phi_{<}(<')=\{\langle x;y\rangle \mid x<y\}\cup<'\cup\{\langle x;y\rangle \mid \langle x;z\rangle,\langle z;y\rangle\in<'\}$.
Proof: by step ⟨2⟩1, $<'$ is not a fixpoint of $\Phi_{<}$.

⟨2⟩3. Q.E.D.

⟨3⟩1. Case: $<'\subset\Phi_{<}(<')$.

⟨4⟩1. Case: $\langle x;y\rangle\in\{\langle x;y\rangle \mid x<y\}$ and $\langle x;y\rangle\notin<'$.
Proof: this is impossible because if $\langle x;y\rangle\notin\{\langle x;y\rangle \mid x<y\}$, then $\langle x;y\rangle\notin<'$ by assumption ⟨1⟩.

⟨4⟩2. Case: $\langle x;y\rangle\in\{\langle x;y\rangle \mid \langle x;z\rangle,\langle z;y\rangle\in<'\}$ and $\langle x;y\rangle\notin<'$.
Proof: this is impossible because by assumption ⟨1⟩, $<'$ is transitive. Therefore, if $\langle x;y\rangle\in\{\langle x;y\rangle \mid \langle x;z\rangle,\langle z;y\rangle\in<'\}$, then $\langle x;y\rangle\in<'$.

⟨4⟩3. Q.E.D.
Proof: steps ⟨4⟩1 and ⟨4⟩2 list all cases for $<'\subset\Phi_{<}(<')$.

⟨3⟩2. Case: $\Phi_{<}(<')\subset<'$.
Proof: this is impossible because $X\subseteq\Phi_{<}(X)$ for any X.

⟨3⟩3. Q.E.D.
Proof: steps ⟨3⟩1 and ⟨3⟩2 list all cases for step ⟨2⟩2 ($<'\neq\Phi_{<}(<')$).

⟨1⟩5. Q.E.D.
Proof: by steps ⟨1⟩2, ⟨1⟩3 and ⟨1⟩4.

7.3.2 Proofs of Properties and Theorems in Chapter 7.

A number of theorems and properties presented in this chapter is proven by induction to the number of times the recursive function defined in the previous subsection is applied. The following definition and lemma enable this method by

establishing a relation between the dependence relation defined in Section 7.1 and a fixed point characterisation of a similar notion:

Definition 7.36. (One-step relations). *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy and let $LT_A=\langle\langle T_A;<_A\rangle;V_A\rangle$, $LT_B=\langle\langle T_B;<_B\rangle;V_B\rangle$ and $LT_L=\langle\langle T_L;<_L\rangle;V_L\rangle$ be three traces of components or links $A,B\in Comp\cup Lnk$ and a link $L\in Lnk$ such that $A=dom(L)$ and $B=cdom(L)$. Let $v_{A,i}$ and $v_{A,j}$ be two states in LT_A, let $v_{L,i''}$, $v_{L,j''}$, $v_{L,k}$ and $v_{L,l}$ be four states in LT_L, and let $v_{B,i'}$ and $v_{B,j'}$ be two states in LT_B.*

- *The* one-step strict dependence relation *is the relation $\rightarrow_{sd1}\subseteq\mathcal{S}_{SH}\times\mathcal{S}_{SH}$ be a binary relation such that $v_{A,i}\rightarrow_{sd1}v_{B,j'}$ iff either*
  1. *$A=B$ and $i<_A j'$, or*
  2. *$\langle\langle v_{A,i};v_{A,j}\rangle;\langle v_{L,i''};v_{L,j''};v_{L,k};v_{L,l}\rangle;\langle v_{B,i'};v_{B,j'}\rangle\rangle$ is a transmission octet for LT_A, LT_L and LT_B, and $i<_A j$, $i''<_I j''<_I k<_I l$, $i'<_B j'$, there is a state $v_{A,k}\in next_{LT_A}(v_{A,i})$ with $k\leq i$ and there is a state $v_{B,k'}\in prev_{LT_B}(v_{B,j})$ with $i'\leq k'$*
- *The* one-step dependence for transitions relation *is the relation $\rightarrow_{tr1}\subseteq\mathcal{S}_{SH}\times\mathcal{S}_{SH}$ be a binary relation such that $t_1=trans(v_{A,i},next_{LT_A}(v_{A,i}))\rightarrow_{tr1}trans(prev_{LT_B}(v_{B,j'}),v_{B,j'})=t_2$ iff either*
  1. *$A=B$ and $next_{LT_A}(v_{A,i})=prev_{LT_B}(v_{B,j'})$, or*
  2. *There is a link L with $dom(L)=A$ and $cdom(L)=B$ and there are i'',j'',k,l in the time frame of $\mu(L)$ such that $\langle\langle v_{A,i};next_{\mu(A)}(v_{A,i})\rangle;\langle v_{L,i''};v_{L,j''};v_{L,k};v_{L,l}\rangle;\langle prev_{\mu(B)}(v_{B,j'});v_{B,j'}\rangle\rangle$ is a transmission octet.*

Lemma 7.37. *Let $\rightarrow_{sd1}$ and $\rightarrow_{tr1}$ be the one-step relations defined in Definition* 7.36*.*

- 1. $\rightarrow_{sd}=(\rightarrow_{sd1})^*$,
- 2. $\rightarrow_{tr}=(\rightarrow_{tr1})^*$.

Proof. Proof sketch: Both $\rightarrow_{sd}$ and $(\rightarrow_{sd1})^*$ are shown to be the smallest elements of a subset of all relations on a set of states. The smallest element in a set of relations is unique. Therefore, $\rightarrow_{sd}=(\rightarrow_{sd1})^*$. The proof for the second part of the lemma is analogous.

⟨1⟩1. $\rightarrow_{sd}=(\rightarrow_{sd1})^*$.

⟨2⟩1. Let Σ be a set of states and let $\Pi=\{R\subseteq\mathcal{P}(\Sigma\times\Sigma)\mid R \text{ transitive and } \rightarrow_{sd1}\subseteq R\}$. Then $\rightarrow_{sd}\in\Pi$.

⟨3⟩1. $\rightarrow_{sd1}\subseteq\rightarrow_{sd}$.
Proof: by the definitions of $\rightarrow_{sd1}$ and $\rightarrow_{sd}$.

⟨3⟩2. $\rightarrow_{sd}$ is transitive.
Proof: by the definition of $\rightarrow_{sd}$, clause 3.

⟨3⟩3. Q.E.D.

Proof: by steps ⟨3⟩1 and ⟨3⟩2.

⟨2⟩2. $\rightarrow_{sd}$ is the smallest element of Π.

Proof: by the definition of $\rightarrow_{sd}$, $\rightarrow_{sd}$ is the smallest relation that complies with the three clauses of the definition.

⟨2⟩3. $(\rightarrow_{sd1})^* \in \Pi$.

Proof: by Proposition 7.35, $(\rightarrow_{sd1})^*$ is transitive and $\rightarrow_{sd1} \subseteq (\rightarrow_{sd1})^*$.

⟨2⟩4. $(\rightarrow_{sd1})^*$ is the smallest element of Π.

Proof: by Proposition 7.35, $(\rightarrow_{sd1})^*$ is the *smallest* transitive relation such that $\rightarrow_{sd} \subseteq (\rightarrow_{sd1})^*$.

⟨2⟩5. Q.E.D.

Proof: by antisymmetry of the partial order $\subseteq$ on $\mathcal{P}(\Sigma \times \Sigma)$, if the subset Π of $\mathcal{P}(\Sigma \times \Sigma)$ has a smallest element, then it is unique. By steps ⟨2⟩2 and ⟨2⟩4, both $\rightarrow_{sd}$ and $(\rightarrow_{sd1})^*$ are smallest elements of Π.

⟨1⟩2. $\rightarrow_{tr} = (\rightarrow_{tr1})^*$

Proof: analogous to proof of step ⟨1⟩1.

⟨1⟩3. Q.E.D.

Proof: steps ⟨1⟩1 and ⟨1⟩2.

The previous lemma enables application of induction to the number of times the function $\Phi_{\rightarrow_{sd1}}$ is applied. In proofs of properties of, e.g., $\rightarrow_{sd}$, induction is applied to prove the same property for $\rightarrow_{sd1}$. By the previous lemma, the property then also holds for, e.g., $\rightarrow_{sd}$.

Proposition 7.8. *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy, let $\gamma=(\gamma_l)_{l \in Lnk}$ be a collection of compatibility relations, and let μ be a multitrace compatible for γ. If $\rightarrow_{sd}$ is an (irreflexive) partial order, then the pair $\langle PART(SH,\gamma,\mu);next_{SGS(SH,\mu,\gamma)}\rangle$ is a lattice.*

Proof. Proof sketch: To show that $\langle PART(SH,\gamma,\mu);next_{SGS(SH,\mu,\gamma)}\rangle$ is a lattice, it suffices to show that for each pair of equivalence classes $X,Y \in PART(SH,\gamma,\mu)$, a lowest upper bound and greatest lower bound exist. These bounds are constructed as follows. First, for an equivalence class X, a function $I(X)$ is defined that maps X to the set of all local states that are before (or equal to) the local states of $\sigma \in X$ with respect to the local orders of the local traces in μ. (The letter I is chosen for these sets as they resemble *order ideals* with respect to the local order of states in μ.) Then, for a set X' of local component and link states, a function $I^{\leftarrow}(X')$ is defined that maps X' to the set of all local component and link states, one for each component or link, that are maximal in X' with respect to the local order. (Please note that it is essential in this construction that states are coloured.) The result is *not* an equivalence class, as it contains only one element. Therefore, an equivalent element with respect to $R \cup R^{-1}$ (as defined in Definition 7.7) is added. The lowest upper bound of X and Y is defined as $I^{\leftarrow}(I(X) \cup I(Y))$. A number of lemmas is used to show that this bound is itself an equivalence class of global states, that it is an upper

bound, and that it is the lowest upper bound. For the greatest lower bound, $I^{\leftarrow}$ is defined as a function that maps an "order ideal" to the set of *minimal* local component and link states. The greatest lower bound is then defined as $I^{\leftarrow}(I(X)\cap I(Y))$. (Note the use of intersection instead of union.)

⟨1⟩1. Define:

1. $\mathcal{OI}(SH,\mu,\gamma)$ is the set of all order ideals for SH, μ and γ:
   $\mathcal{OI}(SH,\mu,\gamma)=\{\bigcup_{S\in Comp\cup Lnk}\{\mu(S)(i) \mid \bot\leq i\leq i' \text{ with } \sigma(S)=v_{S,i'}\} \mid \sigma\in SNAP(SH,\mu,\gamma)\}$
2. A function $I: PART(SH,\mu,\gamma)\rightarrow\mathcal{OI}(SH,\mu,\gamma)$ such that:
   $I(X)=\bigcup_{S\in Comp\cup Lnk}\{\mu(S)(i) \mid \bot\leq i\leq i' \text{ with } \sigma(S)=v_{S,i'} \text{ for } \sigma\in X\}$,
3. A function $I^{max}: \mathcal{OI}(SH,\mu,\gamma)\rightarrow SNAP(SH,\mu,\gamma)$ such that:
   $I^{max}(X')=\bigcup_{S\in Comp\cup Lnk}\{\mu(S)(i) \mid \mu(S)(i)\in X' \text{ and } \neg\exists i'>i: \mu(S)(i')\in X'\}$,
4. A function $I^{\leftarrow}: \mathcal{OI}(SH,\mu,\gamma)\rightarrow PART(SH,\mu,\gamma)$ such that:
   $I^{\leftarrow}(X')=\{I^{max}(X')\}\cup\{\bigcup_{S\in Comp\cup Lnk}\{\mu(S)(i) \mid \exists! S'\in Comp\cup Lnk: v_{S',i}=prev_{\mu(S')}(v_{S,i'}) \text{ and } \forall S''\in Comp\cup Lnk \text{ with } S''\neq S': v_{S'',i}=v_{S,i'} \text{ with } v_{S,i'}\in I^{max}(X')\}\}$.

Then $I^{\leftarrow}(I(X)\cup I(Y))$ is an equivalence class of strict global states with respect to $R\cup R^{-1}$.

Proof: by Lemma 7.38.

⟨1⟩2. $I^{\leftarrow}(I(X)\cup I(Y))$ is an upper bound of X and Y.

Proof: by Lemma 7.39.

⟨1⟩3. $I^{\leftarrow}(I(X)\cup I(Y))$ is the *least* upper bound of X and Y.

Proof: by Lemma 7.40.

⟨1⟩4. Define:

1. A function $I^{min}: \mathcal{OI}(SH,\mu,\gamma)\rightarrow SNAP(SH,\mu,\gamma)$ such that:
   $I^{min}(X')=\bigcup_{S\in Comp\cup Lnk}\{\mu(S)(i) \mid \mu(S)(i)\in X' \text{ and } \neg\exists i'<i: \mu(S)(i')\in X'\}$
2. A function $I^{\leftarrow}: \mathcal{OI}(SH,\mu,\gamma)\rightarrow PART(SH,\mu,\gamma)$ such that:
   $I^{\leftarrow}(X')=\{I^{min}(X')\}\cup\{\bigcup_{S\in Comp\cup Lnk}\{\mu(S)(i) \mid \exists! S'\in Comp\cup Lnk: v_{S',i}=next_{\mu(S')}(v_{S,i'}) \text{ and } \forall S''\in Comp\cup Lnk \text{ with } S''\neq S': v_{S'',i}=v_{S,i'} \text{ with } v_{S,i'}\in I^{max}(X')\}\}$.

Then $I^{\leftarrow}(I(X)\cap I(Y))$ is an equivalence class of strict global states with respect to $R\cup R^{-1}$.

Proof: similar to the proof of Lemma 7.38. The difference is that in steps ⟨6⟩1-4, the cases are the other way around. I.e., for the first case (step ⟨6⟩1), $i\leq i'$ and $j\leq j'$, $\sigma''(S_1)=\sigma(S_1)$ and $\sigma''(S_2)=\sigma(S_2)$ instead of $\sigma''(S_1)=\sigma'(S_1)$ and $\sigma''(S_2)=\sigma'(S_2)$.

⟨1⟩5. $I^{\leftarrow}(I(X)\cap I(Y))$ is a lower bound of X and Y.

Proof: similar to the proof of Lemma 7.39: $I(X)\cap I(Y)\subseteq I(X)$: as $I^{\leftarrow}$ is order-preserving, $\langle I^{\leftarrow}(I(X)\cap I(Y)); I^{\leftarrow}(I(X))\rangle\in next_{SGS(SH,\mu,\gamma)}$. As $\langle I^{\leftarrow}(I(X)); X\rangle\in next_{SGS(SH,\mu,\gamma)}$, by transitivity, $\langle I^{\leftarrow}(I(X)\cap I(Y); X\rangle\in next_{SGS(SH,\mu,\gamma)}$, and similar for Y.

⟨1⟩6. $I^{\leftarrow}(I(\sigma)\cap I(\sigma'))$ is the greatest lower bound of X and Y.

Proof: similar to the proof of Lemma 7.40. The difference is that in steps ⟨6⟩1-4, the cases are the other way around, similar to step ⟨1⟩4 above.

⟨1⟩7. Q.E.D.

Proof: for arbitrary equivalence classes X and Y, by steps ⟨1⟩1, ⟨1⟩2, and ⟨1⟩3, the least upper bound exists, and by steps ⟨1⟩4, ⟨1⟩5, and ⟨1⟩6, the greatest lower bound exists.

Lemma 7.38. *Let $X,Y \in PART(SH,\mu,\gamma)$. Then $I^{\leftarrow}(I(X) \cup I(Y))$ is an equivalence class of strict global states.*

Proof. Proof sketch: Take σ'' as the maximal element in $I^{\leftarrow}(I(X) \cup I(Y))$. Assuming that σ'' is not a strict global state, a contradiction is derived as follows. If σ'' is not a strict global state, then there must be two components or links S_1 and S_2 such that $\sigma''(S_1)$ and $\sigma''(S_2)$ are not independent. In other words, $\sigma''(S_1) \rightarrow_{sd} \sigma''(S_2)$ or $\sigma''(S_2) \rightarrow_{sd} \sigma''(S_1)$. Assume that $\sigma''(S_1) \rightarrow_{sd} \sigma''(S_2)$. With induction to the number of times the fixpoint operator is applied, it is proven that a contradiction can be derived by case analysis. In these cases, the most important step is as follows. Take $\sigma \in X$ and $\sigma' \in Y$ such that σ and σ' are maximal in X and Y, respectively, with respect to the order of local states for each local component or link trace $\mu(S)$, $S \in Comp \cup Lnk$. (σ and σ' are unique as traces are assumed to be linear.) For each component or link S, either $\sigma(S)=\sigma'(S)$, or $\sigma(S)$ is earlier than $\sigma'(S)$, with respect to the local order of states in a local component or link trace, or $\sigma(S)$ is later than $\sigma'(S)$. As σ'' is an element of the union of the "order ideals" $I(X)$ and $I(Y)$, $\sigma''(S)=\sigma'(S)$ or $\sigma''(S)=\sigma(S)$, respectively. (In other words, by definition, for each component or link, the latest state in either σ or σ' is chosen[13].) This enables 'translating' the case $\sigma''(S_1) \rightarrow_{sd} \sigma''(S_2)$ to a statement about either σ or σ'. A contradiction between such a statement and the assumption that σ and σ' are strict global states can then be derived. The proof for the case that $\sigma''(S_2) \rightarrow_{sd} \sigma''(S_1)$ is similar.

Assume:

1. $\sigma \in X$ such that $\forall S \in Comp \cup Lnk: \neg\exists \sigma_1 \in X$ such that $\sigma_1(S)=next_{\mu(S)}(\sigma(S))$
2. $\sigma' \in Y$ such that $\forall S \in Comp \cup Lnk: \neg\exists \sigma_1 \in Y$ such that $\sigma_1(S)=next_{\mu(S)}(\sigma'(S))$
3. σ'' is not a strict global state.

Prove: false.

⟨1⟩1. $\exists S_1,S_2 \in Comp \cup Lnk: \sigma''(S_1) \rightarrow_{sd} \sigma''(S_2) \vee \sigma''(S_2) \rightarrow_{sd} \sigma''(S_1)$.

Proof: by assumption ⟨0⟩:3 and the definition of a strict global state.

⟨1⟩2. Case: $\sigma''(S_1) \rightarrow_{sd} \sigma''(S_2)$.

⟨2⟩1. $\exists S_1,S_2: \langle \sigma''(S_1);\sigma''(S_2) \rangle \in \rightarrow_{sd}=(\rightarrow_{sd1})^*=\bigcup_{n \geq 0} \Phi^n(\emptyset)$, with $\Phi=\Phi_{\rightarrow_{sd1}}$.

Proof: by assumption ⟨1⟩ and Lemma 7.37.

⟨2⟩2. $\exists S_1,S_2: \exists n \geq 0: \langle \sigma''(S_1);\sigma''(S_2) \rangle \in \Phi^n(\emptyset)$.

Proof: by step ⟨2⟩1.

---

[13] For the lower bound $I^{\leftarrow}(I(\sigma) \cap I(\sigma'))$, the earliest state is chosen, because the lower bound is defined as the intersection of the "order ideals" $I(X)$ and $I(Y)$ instead of the union.

⟨2⟩3. $\forall m \geq 0: \langle \sigma''(S_1); \sigma''(S_2) \rangle \in \Phi^m(\emptyset) \Rightarrow$ false.

⟨3⟩1. Case: m=0.

Proof: by definition, $\Phi^m(\emptyset)$ for $m=0$ is the empty set, because for $m=0$, Φ^m is the identity function.

⟨3⟩2. Case: 1. m>0,

2. $\langle \sigma''(S_1); \sigma''(S_2) \rangle \in \Phi^{m-1}(\emptyset) \Rightarrow$ false,

3. $\langle \sigma''(S_1); \sigma''(S_2) \rangle \in \Phi^m(\emptyset) = \Phi(\Phi^{m-1}(\emptyset)) =$
$\{\langle v_{S_1}; v_{S_2} \rangle \mid v_{S_1} \rightarrow_{sd1} v_{S_2}\} \cup \Phi^{m-1}(\emptyset) \cup$
$\{\langle \langle v_{S_1}; v_{S_2} \rangle \mid \langle v_{S_1}; v_{S_3} \rangle, \langle v_{S_3}; v_{S_2} \rangle \in \Phi^{m-1}(\emptyset)\}$

⟨4⟩1. Case: $\sigma''(S_1) \rightarrow_{sd1} \sigma''(S_2)$.

⟨5⟩1. Case: $S_1=S_2$ and $\sigma''(S_2)=next_{\mu(S_1)}(\sigma''(S_1))$.

Proof: impossible, because $\sigma''(S_1)$ cannot be the maximal element in $\mu(S_1)$ as required by the definition of $I^{\leftarrow}$, as $\sigma''(S_2)$ is later by ass. ⟨5⟩.

⟨5⟩2. Case: $\langle\langle \sigma''(S_1); next_{\mu(S_1)}(\sigma''(S_1)) \rangle; \langle v_{L,i''}; v_{L,j''}; v_{L,k}; v_{L,l} \rangle; \langle prev_{\mu(S_2)}(\sigma''(S_2)); \sigma''(S_2) \rangle\rangle$ is a transmission octet for a link L, traces $\mu(S_1)$, $\mu(L)$ and $\mu(S_2)$ and states $v_{L,i''}$, $v_{L,j''}$, $v_{L,k}$, and $v_{L,l}$.

⟨6⟩1. Let $\sigma(S_1)=v_{S_1,i}$, $\sigma'(S_1)=v_{S_1,i'}$, $\sigma(S_2)=v_{S_2,j}$ and $\sigma'(S_2)=v_{S_2,j'}$. Case: $i \leq i'$ and $j \leq j'$.

Proof: $\sigma''(S_1)=\sigma'(S_1)$ and $\sigma''(S_2)=\sigma'(S_2)$ (see proof sketch). By substitution in assumption ⟨5⟩, $\sigma'(S_1) \rightarrow_{sd} \sigma'(S_2)$, which contradicts ⟨0⟩:2.

⟨6⟩2. Case: $i \leq i'$ and $j' < j$.

Proof: $\sigma''(S_1)=\sigma'(S_1)$ and $\sigma''(S_2)=\sigma(S_2)$. By substitution in ⟨5⟩, $\sigma'(S_1) \rightarrow_{sd} \sigma(S_2)$. By ⟨6⟩, $\sigma(S_2) \rightarrow_{sd} \sigma'(S_2)$. By transitivity, $\sigma'(S_1) \rightarrow_{sd} \sigma'(S_2)$, which contradicts ⟨0⟩:2.

⟨6⟩3. Case: $i' < i$ and $j \leq j'$.

Proof: $\sigma''(S_1)=\sigma(S_1)$ and $\sigma''(S_2)=\sigma'(S_2)$. By substitution in ⟨5⟩, $\sigma(S_1) \rightarrow_{sd} \sigma'(S_2)$. By ⟨6⟩, $\sigma'(S_1) \rightarrow_{sd} \sigma(S_1)$. By transitivity, $\sigma(S_1) \rightarrow_{sd} \sigma(S_2)$, which contradicts ⟨0⟩:1.

⟨6⟩4. Case: $i' < i$ and $j' < j$.

Proof: $\sigma''(S_1)=\sigma(S_1)$ and $\sigma''(S_2)=\sigma(S_2)$. By substitution in assumption ⟨5⟩, $\sigma(S_1) \rightarrow_{sd} \sigma(S_2)$, which contradicts ⟨0⟩:1.

⟨6⟩5. Q.E.D.

Proof: steps ⟨6⟩1 to ⟨6⟩4 list all possible cases.

⟨5⟩3. Q.E.D.
Proof: steps ⟨5⟩1 and ⟨5⟩2 list both cases for assumption ⟨4⟩.

⟨4⟩2. Case: $\langle\sigma''(S_1);\sigma''(S_2)\rangle\in\Phi^{m-1}(\varnothing)$.
Proof: by assumption ⟨3⟩:2.

⟨4⟩3. Case: $\exists\nu$: $\langle\sigma''(S_1);\nu\rangle,\langle\nu;\sigma''(S_2)\rangle\in\Phi^{m-1}(\varnothing)$.
Proof: for arbitrary state ν, if $\langle\sigma''(S_1);\nu\rangle\in\Phi^{m-1}(\varnothing)$, then by assumption ⟨3⟩:2, false is derived. By assumption ⟨4⟩, there is a state ν such that $\langle\sigma''(S_1);\nu\rangle\in\Phi^{m-1}(\varnothing)$.

⟨4⟩4. Q.E.D.
Proof: steps ⟨4⟩1, ⟨4⟩2 and ⟨4⟩3 list all cases for assumption ⟨3⟩:3.

⟨3⟩3. Q.E.D.
Proof: steps ⟨3⟩1 and ⟨3⟩2 and induction to m.

⟨2⟩4. Q.E.D.
Proof: by steps ⟨2⟩2 and ⟨2⟩3.

⟨1⟩3. Case: $\sigma''(S_2)\rightarrow_{sd}\sigma''(S_1)$.
Proof: analogous to proof of step ⟨1⟩2.

⟨1⟩4. Q.E.D.
Proof: steps ⟨1⟩2 and ⟨1⟩3 enumerate both cases of ⟨1⟩1.

Lemma 7.39. *Let $X,Y\in PART(SH,\mu,\gamma)$ and let $Z=I^{\leftarrow}(I(X)\cup I(Y))$. Then Z is an upper bound of X and Y.*

Proof. Proof sketch: First, $I^{\leftarrow}$ is shown to be order-preserving, i.e., for sets of states X' and Y', if $X'\subseteq Y'$, then $\langle I^{\leftarrow}(X');I^{\leftarrow}(Y')\rangle\in next_{SGS(SH,\mu,\gamma)}$. This is done using induction to the number of elements in the difference between X' and Y'. (For the application of the induction hypothesis, it is shown that $\langle I^{\leftarrow}(X');I^{\leftarrow}(Y')\rangle\in next_{SGS(SH,\mu,\gamma)}$ for the non-transitive part of the definition of $next_{SGS(SH,\mu,\gamma)}$). After this, the order-preserving property of $I^{\leftarrow}$ is applied twice to show that Z is greater than X and Y, respectively.

⟨1⟩1. $I^{\leftarrow}$ is order-preserving.
Assume: $X',Y'\in\mathcal{O}(SH,\mu,\gamma)$ such that $X'\subseteq Y'$.
Prove: $\langle I^{\leftarrow}(X');I^{\leftarrow}(Y')\rangle\in next_{SGS(SH,\mu,\gamma)}$.

⟨2⟩1. $\forall m\geq 0$: if $|Y'\setminus X'|=m$, then $\langle I^{\leftarrow}(X');I^{\leftarrow}(Y')\rangle\in next_{SGS(SH,\mu,\gamma)}$.

⟨3⟩1. Case: $m=0$
Proof: assume that $|Y'\setminus X'|=0$, thus $X'=Y'$, thus $I^{\leftarrow}(X')=I^{\leftarrow}(Y')$. By reflexiveness, $\langle I^{\leftarrow}(X');I^{\leftarrow}(Y')\rangle\in next_{SGS(SH,\mu,\gamma)}$.

⟨3⟩2. Case: 1. $m>0$,
2. if $|Y''\setminus X''|=m-1$, then $\langle I^{\leftarrow}(X'');I^{\leftarrow}(Y'')\rangle\in next_{SGS(SH,\mu,\gamma)}$,
3. $|Y'\setminus X'|=m$.

⟨4⟩1. Let $S \in Comp \cup Lnk$ and take $v_{S,i'} \in Y'$ such that $v_{S,i'} = next_{\mu(S)}(v_{S,i})$ with $v_{S,i}$ maximal for $\mu(S)$ in X'. Then $|Y' \setminus X'| = \{v_{S,i'}\} \cup (Y' \setminus (X' \cup \{v_{S,i'}\}))$.

Proof: as $X', Y' \in \mathcal{O}(SH,\mu,\gamma)$ and $X' \subseteq Y'$, $v_{S,i'} \notin X'$, as $v_{S,i}$ is maximal in X' and $v_{S,i'} = next_{\mu(S)}(v_{S,i})$.

⟨4⟩2. $|Y' \setminus (X' \cup \{v_{S,i'}\})| = m-1$.

Proof: by step ⟨4⟩1, $v_{S,i'} \notin X'$ and by assumption ⟨3⟩:3, $|Y' \setminus X'| = m$.

⟨4⟩3. $\langle I^{\leftarrow}(X' \cup \{v_{S,i'}\}); I^{\leftarrow}(Y') \rangle \in next_{SGS(SH,\mu,\gamma)}$.

Proof: by step ⟨4⟩2 and assumption ⟨3⟩:2 (induction).

⟨4⟩4. $\langle I^{\leftarrow}(X'); I^{\leftarrow}(X' \cup \{v_{S,i'}\}) \rangle \in next_{SGS(SH,\mu,\gamma)}$.

⟨5⟩1. Let σ be the maximal element in $I^{\leftarrow}(X')$. Then $\forall S' \in Comp \cup Lnk$ with $S' \neq S$: $\sigma(S') = \sigma(S)$.

Proof: $(X' \cup \{v_{S,i'}\}) \setminus X' = \{v_{S,i'}\}$, so $v_{S,i'}$ is the only difference.

⟨5⟩2. Case: $S \in Comp$ and there is no $I \in Lnk$ such that $\langle\langle v_{dom(I),i}; v_{dom(I),j} \rangle; \langle v_{I,i''}; v_{I,j''}; \sigma(I); next_{\mu(I)}(\sigma(I)) \rangle; \langle \sigma(S); v_{S,i'} \rangle\rangle$ is a transmission octet.

Proof: clause 2 of the definition of $next_{SGS(SH,\mu,\gamma)}$.

⟨5⟩3. Case: $S \in Lnk$ and there is no $I \in Lnk$ such that $\langle\langle v_{dom(I),i}; v_{dom(I),j} \rangle; \langle v_{I,i''}; v_{I,j''}; \sigma(I); next_{\mu(I)}(\sigma(I)) \rangle; \langle \sigma(S); v_{S,i'} \rangle\rangle$ is a transmission octet.

Proof: clause 3 of the definition of $next_{SGS(SH,\mu,\gamma)}$.

⟨5⟩4. Case: $S \in Lnk$ and there is an $I \in Lnk$ such that $\langle\langle v_{dom(I),i}; v_{dom(I),j} \rangle; \langle v_{I,i''}; v_{I,j''}; \sigma(I); next_{\mu(I)}(\sigma(I)) \rangle; \langle \sigma(S); v_{S,i'} \rangle\rangle$ is a transmission octet.

Proof: same equivalence class.

⟨5⟩5. Case: $S \in Comp$ and there is an $I \in Lnk$ such that $\langle\langle v_{dom(I),i}; v_{dom(I),j} \rangle; \langle v_{I,i''}; v_{I,j''}; \sigma(I); next_{\mu(I)}(\sigma(I)) \rangle; \langle \sigma(S); v_{S,i'} \rangle\rangle$ is a transmission octet.

Proof: the case is empty.

⟨5⟩6. Q.E.D.

Proof: as $X' \neq Y'$ by assumption ⟨3⟩:1, steps ⟨5⟩2-5 list all cases.

⟨4⟩5. Q.E.D.

Proof: by steps ⟨4⟩3, ⟨4⟩4 and transitivity of $next_{SGS(SH,\mu,\gamma)}$.

⟨3⟩2. Q.E.D.

Proof: steps ⟨3⟩1 and ⟨3⟩2 and induction to m.

⟨2⟩2. Q.E.D.

Proof: by assumption ⟨1⟩, $\exists m$: $|X' \setminus Y'| = m$. By step ⟨2⟩1, $\langle I^{\leftarrow}(X); I^{\leftarrow}(Y) \rangle \in next_{SGS(SH,\mu,\gamma)}$..

⟨1⟩2. $\langle X;Z \rangle \in next_{SGS(SH,\mu,\gamma)}$.

⟨2⟩1. $I(X) \subseteq I(X) \cup I(Y)$.

Proof: trivial.

⟨2⟩2. $\langle I^{\leftarrow}(I(X)); I^{\leftarrow}(I(X) \cup I(Y)) \rangle \in next_{SGS(SH,\mu,\gamma)}$.

Proof: by steps ⟨2⟩1 and ⟨1⟩1.

⟨2⟩3. $\langle I^{\leftarrow}(I(X)); Z \rangle \in next_{SGS(SH,\mu,\gamma)}$.

Proof: substitute $Z = I^{\leftarrow}(I(X) \cup I(Y))$ in step ⟨2⟩2.

⟨2⟩4. $\langle X; I^{\leftarrow}(I(X)) \rangle \in next_{SGS(SH,\mu,\gamma)}$.

Proof: X and $I^{\leftarrow}(I(X))$ are in the same equivalence class with respect to $R \cup R^{-1}$, and $next_{SGS(SH,\mu,\gamma)}$ is reflexive.

⟨2⟩5. Q.E.D.

Proof: by steps ⟨2⟩3 and ⟨2⟩4 and transitivity of $next_{SGS(SH,\mu,\gamma)}$.

⟨1⟩3. $\langle Y;Z \rangle \in next_{SGS(SH,\mu,\gamma)}$.

Proof: analogous to ⟨1⟩2.

⟨1⟩4. Q.E.D.

Proof: steps ⟨1⟩2 and ⟨1⟩3.

Lemma 7.40**.** *Let $X,Y \in PART(SH,\mu,\gamma)$ and let $Z = I^{\leftarrow}(I(X) \cup I(Y))$. Then Z is the least upper bound of X and Y.*

Proof. Proof sketch: From the assumption that there is an upper bound Z' of X and Y that is smaller than Z, a contradiction is derived as follows. First, $next_{SGS(SH,\mu,\gamma)} = (onestep_{SGS(SH,\mu,\gamma)})^*$, where $onestep_{SGS(SH,\mu,\gamma)}$ is the non-transitive part of the relation $next_{SGS(SH,\mu,\gamma)}$. With induction to the number of times the fixpoint operator is applied in $(onestep_{SGS(SH,\mu,\gamma)})^*$, a contradiction is derived from the assumption that Z' is smaller than Z, while Z' is an upper bound of X and Y. In the proof, case analysis similar to the proof of Lemma 7.38 is applied. In these cases, the most important step is as follows. Take $\sigma_X \in X$, $\sigma_Y \in Y$ and $\sigma_Y \in Y$ such that σ_X, σ_Y and σ_Z are maximal in X, Y and Z, respectively, with respect to the order of local states for each local component or link trace $\mu(S)$, $S \in Comp \cup Lnk$. For each component or link S, either $\sigma_X(S) = \sigma_Y(S)$, or $\sigma_X(S)$ is earlier than $\sigma_Y(S)$, with respect to the local order of states in a local component or link trace, or $\sigma_X(S)$ is later than $\sigma_Y(S)$. As σ_Z is an element of the union of the "order ideals" $I(X)$ and $I(Y)$, $\sigma_Z(S) = \sigma_Y(S)$ or $\sigma_Z(S) = \sigma_X(S)$, respectively. This enables 'translating' statement that Z' is smaller than Z to a statement about either σ_X or σ_Y. A contradiction between such a statement and the assumption that Z' is an upper bound of X and Y can then be derived.

Assume: Z is not the least upper bound of X and Y.

Prove: false.

⟨1⟩1. Z is an upper bound of X and Y.

Proof: by Lemma 7.39.

⟨1⟩2. There is an equivalence class $Z' \neq Z$ such that Z' is an upper bound of X and Y and $\langle Z';Z\rangle \in next_{SGS(SH,\mu,\gamma)} = (onestep_{SGS(SH,\mu,\gamma)})^* = \bigcup_{n\geq 0} \Phi^n(\emptyset)$, with $\Phi = \Phi_{onestep_{SGS(SH,\mu,\gamma)}}$.
Proof: by assumption ⟨0⟩.

⟨1⟩3. $\exists Z'$: $Z' \neq Z$ is an upper bound of X and Y, and $\exists n$: $\langle Z';Z\rangle \in \Phi^n(\emptyset)$.
Proof: by step ⟨1⟩2.

⟨1⟩4. $\forall m \geq 0$: if $\langle Z';Z\rangle \in \Phi^m(\emptyset)$, then false, for Z' upper bound of X and Y.

⟨2⟩1. Case: $m=0$.
Proof: by definition, $\Phi^m(\emptyset)$ for $m=0$ is the empty set, because for $m=0$, Φ^m is the identity function.

⟨2⟩2. Case: 1. $m>0$,
2. if $\langle Z';Z\rangle \in \Phi^{m-1}(\emptyset)$, then false, for Z' upper bound of X and Y,
3. $\langle Z';Z\rangle \in \Phi^m(\emptyset) = \Phi(\Phi^{m-1}(\emptyset)) = \{\langle x;y\rangle \mid \langle x;y\rangle \in onestep_{SGS(SH,\mu,\gamma)}\} \cup \Phi^{m-1}(\emptyset) \cup \{\langle x;z\rangle \mid \langle x;y\rangle, \langle y;z\rangle \in \Phi^{m-1}(\emptyset)\}$,
4. Z' upper bound of X and Y.

⟨3⟩1. Case: $\langle Z';Z\rangle \in onestep_{SGS(SH,\mu,\gamma)}$.

⟨4⟩1. Let $\sigma_{Z'} \in Z'$: $\forall S \in Comp \cup Lnk$: $\neg\exists \sigma_1 \in Z'$: $\sigma_1(S) = next_{\mu(S)}(\sigma_{Z'}(S))$
Let $\sigma_Z \in Z$: $\forall S \in Comp \cup Lnk$: $\neg\exists \sigma_1 \in Z$: $\sigma_1(S) = next_{\mu(S)}(\sigma_Z(S))$
Let $\sigma_X \in X$: $\forall S \in Comp \cup Lnk$: $\neg\exists \sigma_1 \in X$: $\sigma_1(S) = next_{\mu(S)}(\sigma_X(S))$
Let $\sigma_Y \in Y$: $\forall S \in Comp \cup Lnk$: $\neg\exists \sigma_1 \in Y$: $\sigma_1(S) = next_{\mu(S)}(\sigma_Y(S))$
Case: there is exactly one $C \in Comp$ such that $\sigma_Z(C) = next_{\mu(C)}(\sigma_{Z'}(C))$ and $\forall S \in Comp \cup Lnk$ with $S \neq C$: $\sigma_Z(S) = \sigma_{Z'}(S)$ and there is no transmission octet $\langle\langle v_{dom(I),i}; v_{dom(I),j}\rangle; \langle v_{I,i''}; v_{I,j''}; \sigma(I); next_{\mu(I)}(\sigma(I))\rangle; \langle \sigma(S);\ v_{S,i'}\rangle\rangle$ for any link I.

⟨5⟩1. Let $\sigma_X(C) = v_{C,i}$ and let $\sigma_Y(C) = v_{C,i'}$. Case: $i \leq i'$.
Proof: $\sigma_Z(C) = \sigma_Y(C) = next_{\mu(C)}(\sigma_{Z'}(C))$. Thus $\langle Z';Y\rangle \in next_{SGS(SH,\mu,\gamma)}$ by clause 2 of the definition of $next_{SGS(SH,\mu,\gamma)}$. Thus, Z' is not an upper bound of Y, which contradicts ass. ⟨2⟩:4.

⟨5⟩2. Case: $i' < i$.
Proof: $\sigma_Z(C) = \sigma_X(C) = next_{\mu(C)}(\sigma_{Z'}(C))$. Thus $\langle Z';X\rangle \in next_{SGS(SH,\mu,\gamma)}$ by clause 2 of the definition of $next_{SGS(SH,\mu,\gamma)}$. Thus, Z' is not an upper bound of X, which contradicts ass. ⟨2⟩:4.

⟨5⟩3. Q.E.D.
Proof: steps ⟨5⟩1 and ⟨5⟩2 list both cases.

⟨4⟩2. Case: there is exactly one $I \in Lnk$ such that $\sigma_Z(I) = next_{\mu(I)}(\sigma_{Z'}(I))$ and $\forall S \in Comp \cup Lnk$ with $S \neq I$: $\sigma_Z(S) = \sigma_{Z'}(S)$ and there is no transmission octet

$\langle\langle v_{dom(I),i}; v_{dom(I),j}\rangle;\langle v_{I,i''}; v_{I,j''}; \sigma(I); next_{\mu(I)}(\sigma(I))\rangle;\langle\sigma(S); v_{S,i'}\rangle\rangle$ for any link I.

Proof: similar to proof of the previous step.

⟨4⟩3. Case: there is exactly one $C \in Comp$ such that $\sigma_Z(C)=next_{\mu(C)}(\sigma_{Z'}(C))$ and there is exactly one $I \in Comp$ such that $\sigma_Z(I)=next_{\mu(I)}(\sigma_{Z'}(I))$and $\forall S \in Comp \cup Lnk$ with $S \neq C$ and $S \neq I$: $\sigma_Z(S)=\sigma_{Z'}(S)$ and there is either a transmission octet $\langle\langle v_{dom(I),i}; v_{dom(I),j}\rangle;\langle v_{I,i''}; v_{I,j''}; \sigma(I); next_{\mu(I)}(\sigma(I))\rangle;\langle\sigma(S); v_{S,i'}\rangle\rangle$ or $\langle\langle v_{dom(I),i}; v_{dom(I),j}\rangle;\langle v_{I,i''}; v_{I,j''}; \sigma(I); next_{\mu(I)}(\sigma(I))\rangle;\langle\sigma(S); v_{S,i'}\rangle\rangle$.

Proof: similar to proof of the previous step.

⟨4⟩4. Q.E.D.

Proof: as $Z' \neq Z$, steps ⟨4⟩1-3 list all cases for assumption ⟨3⟩.

⟨3⟩2. Case: $\langle Z'; Z\rangle \in \Phi^{m-1}(\varnothing)$.

Proof: by assumption ⟨2⟩:2.

⟨3⟩3. Case: $\exists Z'': \langle Z'; Z''\rangle, \langle Z''; Z\rangle \in \Phi^{m-1}(\varnothing)$.

Proof: for arbitrary Z'', if $\langle Z''; Z\rangle \in \Phi^{m-1}(\varnothing)$, then false by assumption ⟨2⟩:2. By assumption ⟨3⟩, such Z'' exists.

⟨3⟩4. Q.E.D.

Proof: steps ⟨3⟩1, ⟨3⟩2 and ⟨3⟩3 list all cases for assumption ⟨2⟩:3.

⟨2⟩3. Q.E.D.

Proof: steps ⟨2⟩1 and ⟨2⟩2 and induction to m.

⟨1⟩5. Q.E.D.

Proof: by steps ⟨1⟩1, ⟨1⟩3 and ⟨1⟩4.

Proposition 7.9. *Let $SH=\langle Comp; Lnk; \prec; dom; cdom\rangle$ be a structure hierarchy, let γ be a collection of compatibility relations, let μ be a multitrace compatible for γ and let $\sigma \in SNAP(SH,\mu,\gamma)$ be a snapshot. If for all $S_1 \neq S_2 \in Comp \cup Lnk$ and for all j' such that $\perp < j' \leq j$, with j such that $\sigma(S_1)=\mu_{S_1}(j)$, there is no transmission octet $\langle\langle\sigma(S_1);\sigma(S_1)'\rangle;\langle v_{L,i''}; v_{L,j''}; v_{L,k}; v_{L,l}\rangle;\langle prev_{\mu(S_2)}(\mu_{S_2}(j')); \mu_{S_2}(j')\rangle\rangle \in \gamma_L$ for any link L from S_1 to S_2, $\sigma(S_1)' \in next_{\mu(S_1)}(\sigma(S_1))$ and i'',j'',k,l in the time frame of μ_L, then σ is a strict global state.*

Proof. Proof sketch: Assuming that there is no transmission octet for any j' as defined in the proposition and nevertheless, σ is not a strict global state, a contradiction is derived.

Assume:

1. $\forall S_1 \neq S_2 \in Comp \cup Lnk$, $\forall j': \perp < j' \leq j$ with j such that $\sigma(S_1)=\mu_{S_1}(j)$, there is no transmission octet $\langle\langle\sigma(S_1);\sigma(S_2)'\rangle;\langle v_{L,i''}; v_{L,j''}; v_{L,k}; v_{L,l}\rangle;\langle prev_{\mu(S_2)}(\mu_{S_2}(j'));\mu_{S_2}(j')\rangle\rangle \in \gamma_L$ for any link L from S_1 to S_2, $\sigma(S_1)' \in next_{\mu(S_1)}(\sigma(S_1))$ and i'',j'',k,l in the time frame of μ_L.
2. σ is not a strict global state.

Prove: false.

⟨1⟩1. $\exists S_1,S_2 \in Comp \cup Lnk: \sigma(S_1) \rightarrow_{sd} \sigma(S_2)$ or $\sigma(S_2) \rightarrow_{sd} \sigma(S_1)$.

Proof: by assumption ⟨0⟩:2 and the definition of a strict global state.

⟨1⟩2. Case: $\sigma(S_1) \rightarrow_{sd} \sigma(S_2)$.

⟨2⟩1. $\exists S_1,S_2: \langle \sigma(S_1);\sigma(S_2) \rangle \in \rightarrow_{sd} = (\rightarrow_{sd1})^* = \bigcup_{n \geq 0} \Phi^n(\emptyset)$, with $\Phi = \Phi_{\rightarrow_{sd1}}$.

Proof: by assumption ⟨1⟩ and Lemma 7.37.

⟨2⟩2. $\exists S_1,S_2: \exists n \geq 0: \langle \sigma(S_1);\sigma(S_2) \rangle \in \Phi^n(\emptyset)$.

Proof: by step ⟨2⟩1.

⟨2⟩3. $\forall m \geq 0: \langle \sigma(S_1);\sigma(S_2) \rangle \in \Phi^m(\emptyset) \Rightarrow \exists j': \perp < j' < j$ such that there is a transmission octet.

⟨3⟩1. Case: $m=0$.

Proof: by definition, $\Phi^m(\emptyset)$ for $m=0$ is the empty set, because for $m=0$, Φ^m is the identity function.

⟨3⟩2. Case: 1. $m>0$,

2. $\langle \sigma(S_1);\sigma(S_2) \rangle \in \Phi^{m-1}(\emptyset) \Rightarrow \exists j': \perp < j' < j$ such that there is a transmission octet,

3. $\langle \sigma(S_1);\sigma(S_2) \rangle \in \Phi^m(\emptyset) = \Phi(\Phi^{m-1}(\emptyset)) =$
$\{\langle v_{S_1};v_{S_2} \rangle \mid v_{S_1} \rightarrow_{sd1} v_{S_2}\} \cup \Phi^{m-1}(\emptyset) \cup$
$\{\langle \langle v_{S_1};v_{S_2} \rangle \mid \langle v_{S_1};v_{S_3} \rangle, \langle v_{S_3};v_{S_2} \rangle \in \Phi^{m-1}(\emptyset)\}$

⟨4⟩1. Case: $\sigma(S_1) \rightarrow_{sd1} \sigma(S_2)$.

⟨5⟩1. Case: $S_1=S_2$ and $\sigma(S_2)=next_{\mu(S_1)}(\sigma(S_1))$.

Proof: the case is empty, as the next state relation is not reflexive.

⟨5⟩2. Case: $\langle \langle \sigma(S_1);next_{\mu(S_1)}(\sigma(S_1)) \rangle;\langle v_{L,i''};v_{L,j''};v_{L,k};v_{L,l} \rangle;$
$\langle prev_{\mu(S_2)}(\sigma(S_1));\sigma(S_2) \rangle \rangle$ is a transmission octet for a link L, traces $\mu(S_1)$, $\mu(L)$ and $\mu(S_2)$ and states $v_{L,i''}$, $v_{L,j''}$, $v_{L,k}$, and $v_{L,l}$.

Proof: trivial.

⟨5⟩3. Q.E.D.

Proof: steps ⟨5⟩1 and ⟨5⟩2 list all cases for assumption ⟨3⟩:3.

⟨4⟩2. Case: $\langle \sigma(S_1);\sigma(S_2) \rangle \in \Phi^{m-1}(\emptyset)$.

Proof: by assumption ⟨3⟩:2.

⟨4⟩3. Case: $\exists S_3: \langle v_{S_1};v_{S_3} \rangle, \langle v_{S_3};v_{S_2} \rangle \in \Phi^{m-1}(\emptyset)$.

Proof: for arbitrary state v, if $\langle v_{S_1};v \rangle \in \Phi^{m-1}(\emptyset)$, then by assumption ⟨3⟩:2, $\exists j': \perp < j' < j$ such that there is a transmission octet. By assumption ⟨4⟩, there is a state v such that $\langle v_{S_1};v \rangle \in \Phi^{m-1}(\emptyset)$.

⟨4⟩4. Q.E.D.

Proof: steps ⟨4⟩1, ⟨4⟩2 and ⟨4⟩3 list all cases for assumption ⟨3⟩:3.

⟨3⟩3. Q.E.D.
Proof: steps ⟨3⟩1 and ⟨3⟩2 and induction to m.

⟨2⟩4. Q.E.D.
Proof: by application of ⟨2⟩3, there is a transmission octet, which contradicts assumption ⟨0⟩:1.

⟨1⟩3. Case: $\sigma(S_2)\rightarrow_{sd}\sigma(S_1)$.
Proof: analogous to proof of step ⟨1⟩2.

⟨1⟩4. Q.E.D.
Proof: steps ⟨1⟩2 and ⟨1⟩3 enumerate both cases of ⟨1⟩1.

Proposition 7.12. *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy, let $\gamma=(\gamma_I)_{I\in Lnk}$ be a collection of compatibility relations, let μ be a multitrace compatible for γ and let $GTS(SH,\mu,\gamma)$ be a global transition system for SH. If $\sigma\in\{q_0\}^*$, then σ is a strict global state.*

Proof. Proof sketch: The proof consists of a straightforward induction to the length of the transition from q_0 to σ. A lemma is applied for the inductive case.

Assume: $\sigma\in\{q_0\}^*=\bigcup_{n\geq 0}\{\sigma'\in SNAP(SH,\mu,\gamma)\mid q_0\overset{n}{\rightarrow}_g\sigma'\}$.
Prove: σ is a strict global state.

⟨1⟩1. $\exists n\geq 0$: $\sigma\in\{\sigma'\in SNAP(SH,\mu,\gamma)\mid q_0\overset{n}{\rightarrow}_g\sigma'\}$.
Proof: by assumption ⟨0⟩.

⟨1⟩2. $\forall m\geq 0$: if $\sigma\in\{\sigma'\in SNAP(SH,\mu,\gamma)\mid q_0\overset{m}{\rightarrow}_g\sigma'\}$, then σ is a strict global state.

⟨2⟩1. Case: $m=0$.
Proof: by definition of $\overset{m}{\rightarrow}_g$, $\sigma=q_0$ in this case. q_0 is a strict global state.

⟨2⟩2. Case: 1. $m>0$.
2. $\forall\sigma''$: if $\sigma''\in\{\sigma'\in SNAP(SH,\mu,\gamma)\mid q_0\overset{m-1}{\rightarrow}_g\sigma'\}$, then σ'' is a strict global state.
3. $\sigma\in\{\sigma'\in SNAP(SH,\mu,\gamma)\mid q_0\overset{m}{\rightarrow}_g\sigma'\}$.

⟨3⟩1. $\exists\sigma''$: $q_0\overset{m-1}{\rightarrow}_g\sigma''$ and $\sigma''\rightarrow_g\sigma$.
Proof: by assumption ⟨2⟩:3 and the definition of $\overset{m}{\rightarrow}_g$.

⟨3⟩2. σ'' is a strict global state.
Proof: by step ⟨3⟩1 assumption ⟨2⟩:2.

⟨3⟩3. Q.E.D.
Proof: by Lemma 7.41 and step ⟨3⟩2.

⟨2⟩3. Q.E.D.
Proof: by steps ⟨2⟩1 and ⟨2⟩2 and induction to m.

⟨1⟩3. Q.E.D.
Proof: by steps ⟨1⟩1 and ⟨1⟩2.

Lemma 7.41. *$\forall\sigma,\sigma' \in SNAP(SH,\mu,\gamma)$ such that $\sigma \rightarrow_g \sigma'$: if σ is a strict global state, then σ' is a strict global state.*

Proof. Proof sketch: At the highest level, the proof consists of proofs of the lemma for the three cases for $\sigma \rightarrow_g \sigma'$: $\langle\sigma,\sigma'\rangle$ is an internal global transition, $\langle\sigma,\sigma'\rangle$ is a sending global transition, or $\langle\sigma,\sigma'\rangle$ is a receiving global transition. Each case itself proves that for arbitrary $S_1,S_2 \in Comp \cup Lnk$, $\sigma'(S_1) /\!\!/ \sigma'(S_2)$. This is done by case analysis for possible choices for S_1 and S_2.

Assume: 1. $\sigma \rightarrow_g \sigma'$, 2. σ is a strict global state.

Prove: σ' is a strict global state.

⟨1⟩1. Case: $\langle\sigma,\sigma'\rangle$ is an internal global transition.

⟨2⟩1. Let $S \in Comp \cup Lnk$ such that $\sigma'(S) \in next_{\mu(S)}(\sigma(S))$ and $\forall S' \in Comp \cup Lnk$ with $S' \neq S$: $\sigma'(S') = \sigma(S')$ and there is no transmission octet $\langle\langle\sigma(S);\sigma'(S)\rangle;\langle v_{I,i''}; v_{I,j''}; v_{I,k}; v_{I,l}\rangle;\langle v_{cdom(I),i'}; v_{cdom(I),j'}\rangle\rangle$ for any link I. Let $S_1,S_2 \in Comp \cup Lnk$. Then $\sigma'(S_1) /\!\!/ \sigma'(S_2)$.

⟨3⟩1. Case: $S_1 \neq S$ and $S_2 \neq S$.

Proof: by assumptions ⟨2⟩ and ⟨3⟩, $\sigma'(S_1) = \sigma(S_1) /\!\!/ \sigma(S_2) = \sigma'(S_2)$, as σ is a strict global state by assumption ⟨0⟩:2.

⟨3⟩2. Case: $S_1 = S$ and $S_2 = S$.

Proof: by reflexivity of the independence relation.

⟨3⟩3. Case: $S_1 = S$ and $S_2 \neq S$.

⟨4⟩1. $\neg(\sigma'(S_1) \rightarrow_{sd} \sigma'(S_2))$.

Proof: Assume that $\sigma'(S_1) \rightarrow_{sd} \sigma'(S_2)$. By substitution of ass. ⟨3⟩ and ⟨2⟩, $v \rightarrow_{sd} \sigma(S_2)$ for $v \in next_{\mu(S1)}(\sigma(S_1))$. Then $\sigma(S_1) \rightarrow_{sd} \sigma(S_2)$ by transitivity of $\rightarrow_{sd}$. This contradicts assumption ⟨0⟩:2.

⟨4⟩2. $\neg(\sigma'(S_2) \rightarrow_{sd} \sigma'(S_1))$.

Proof: Assume that $\sigma'(S_2) \rightarrow_{sd} \sigma'(S_1)$. By substitution of ass. ⟨3⟩ and ⟨2⟩, $\sigma(S_2) \rightarrow_{sd} v$ for $v \in next_{\mu(S1)}(\sigma(S_1))$. As $S_1 \neq S_2$, there must be a transmission octet $\langle\langle\sigma(S_2);next_{\mu(S_2)}(\sigma(S_2))\rangle;\langle v_{I,i''}; v_{I,j''}; v_{I,k}; v_{I,l}\rangle;\langle\sigma(S_1);v\rangle\rangle$, which contradicts assumption ⟨1⟩.

⟨4⟩3. Q.E.D.

Proof: by steps ⟨4⟩1 and ⟨4⟩2, and the definition of the independence relation, $\sigma'(S_1) /\!\!/ \sigma'(S_2)$.

⟨3⟩4. Case: $S_1 \neq S$ and $S_2 = S$.

Proof: similar to the proof of the previous case.

⟨3⟩5. Q.E.D.

Proof: steps ⟨3⟩1-4 list all cases for S_1 and S_2.

⟨2⟩2. Q.E.D.

Proof: by step ⟨2⟩1, $\sigma'(S_1)$ and $\sigma'(S_2)$ are independent for arbitrary S_1 and S_2.

⟨1⟩2. Case: $\langle\sigma;\sigma'\rangle$ is a sending global transition.

⟨2⟩1. Let $S \in Comp \cup Lnk$ such that $\sigma'(S) \in next_{\mu(S)}(\sigma(S))$, let $I \in Lnk$ such that $\sigma'(I) \in next_{\mu(I)}(\sigma(I))$ and $\forall S' \in Comp \cup Lnk$ with $S' \neq S$ and $S' \neq I$: $\sigma'(S') = \sigma(S')$ and there is a transmission octet $\langle\langle\sigma(S);\sigma'(S)\rangle;\langle\sigma(I);\sigma'(I);v_{I,k};v_{I,l}\rangle;\langle v_{cdom(I),i'};v_{cdom(I),j'}\rangle\rangle$ for I. Let $S_1,S_2 \in Comp \cup Lnk$. Then $\sigma'(S_1) /\!\!/ \sigma'(S_2)$.

⟨3⟩1. Case: $S_1 \neq C$ and $S_2 \neq C$ and $S_1 \neq I$ and $S_2 \neq I$.
Proof: by assumptions ⟨2⟩ and ⟨3⟩, $\sigma'(S_1) = \sigma(S_1) /\!\!/ \sigma(S_2) = \sigma'(S_2)$, as σ is a strict global state by assumption ⟨0⟩:2.

⟨3⟩2. Case: $(S_1 = C$ and $S_2 = C)$ or $(S_1 = I$ and $S_2 = I)$.
Proof: by reflexivity of the independence relation.

⟨3⟩3. Case: $S_1 = C$ and $S_2 = I$.
Proof: by definition of $\rightarrow_{sd}$.

⟨3⟩4. Case: $S_1 = I$ and $S_2 = C$.
Proof: similar to previous case.

⟨3⟩5. Case: $S_1 = C$ and $S_2 \neq C$ and $S_2 \neq I$.

⟨4⟩1. $\neg(\sigma'(S_1) \rightarrow_{sd} \sigma'(S_2))$.
Proof: Assume that $\sigma'(S_1) \rightarrow_{sd} \sigma'(S_2)$. By substitution of ass. ⟨3⟩ and ⟨2⟩, $v \rightarrow_{sd} \sigma(S_2)$ for $v \in next_{\mu(S1)}(\sigma(S_1))$. Then $\sigma(S_1) \rightarrow_{sd} \sigma(S_2)$ by transitivity of $\rightarrow_{sd}$. This contradicts assumption ⟨0⟩:2.

⟨4⟩2. $\neg(\sigma'(S_2) \rightarrow_{sd} \sigma'(S_1))$.
Proof: Assume that $\sigma'(S_2) \rightarrow_{sd} \sigma'(S_1)$. By substitution of ass. ⟨3⟩, $\sigma(S_2) \rightarrow_{sd} \sigma'(S)$. As $S_1 \neq S_2$, there must be a transmission octet $\langle\langle\sigma(S_2);next_{\mu(S_2)}(\sigma(S_2))\rangle;\langle v_{I,i''};v_{I,j''};v_{I,k};v_{I,l}\rangle;\langle\sigma(S);\sigma'(S)\rangle\rangle$. This contradicts ass. ⟨1⟩, as by ass. ⟨1⟩, $\langle\sigma(S);\sigma'(S)\rangle$ is the first element of a transmission octet and $\rightarrow_{sd}$ is a partial order.

⟨4⟩3. Q.E.D.
Proof: by steps ⟨4⟩1 and ⟨4⟩2, and the definition of the independence relation, $\sigma'(S_1) /\!\!/ \sigma'(S_2)$.

⟨3⟩6. Case: $S_1 \neq C$ and $S_1 \neq I$ and $S_2 = C$.
Proof: similar to previous case.

⟨3⟩7. Case: $S_1 = I$ and $S_2 \neq C$ and $S_2 \neq I$.
Proof: by definition of $\rightarrow_{sd}$.

⟨3⟩8. Case: $S_1 \neq C$ and $S_1 \neq I$ and $S_2 = I$.
Proof: similar to previous case.

⟨3⟩9. Q.E.D.

Proof: steps ⟨3⟩1-8 list all cases for step ⟨2⟩1.

⟨1⟩3. Case: $\langle\sigma,\sigma'\rangle$ is a receiving global transition.
Proof: similar to the proof of step ⟨1⟩2.

⟨1⟩4. Q.E.D.
Proof: steps ⟨1⟩1-3 list all cases for assumption ⟨0⟩:1.

Proposition 7.17. *Let SH be a structure hierarchy and let μ be an atomic computation for SH. Then:*

$$t_1=trans(\nu_{A,i},next_{\mu(A)}(\nu_{A,i}))\rightarrow_{tr}trans(prev_{\mu(B)}(\nu_{B,j'}),\nu_{B,j'})=t_2$$
$$\Leftrightarrow\quad \nu_{A,i}\rightarrow_{sd}\nu_{B,j'} \text{ and } \nu_{B,j'}\neq next_{\mu(A)}(\nu_{A,i}).$$

Proof. Proof sketch: At the highest level, the proof consists of proofs for the two implications in either direction. Both implications are proven with induction to the number of times the one-step relations defined in Definition 7.36 are applied. By Lemma 7.37, it follows that the implications hold for $\rightarrow_{tr}$ and $\rightarrow_{sd}$. The proof of the implication from left to right is the most involved. The central step is step 1.2.2.3 (denoted ⟨4⟩3 in the proof—a four-part step number ending with '3'). The assumption that μ is an atomic computation is crucial in this step. The proof of step 1.2.2.3 proceeds as follows. In this step, it is assumed that there is a transition $t_3=trans(\nu_{C,z},next_{\mu(C)}(\nu_{C,z}))$ such that $\langle t_1;t_3\rangle\in\Psi^{m-1}(\varnothing)$ and $\langle t_3;t_2\rangle\in\Psi^{m-1}(\varnothing)$. By induction, $\nu_{A,i}\rightarrow_{sd}next_{\mu(C)}(\nu_{C,z})$ and $\nu_{C,z}\rightarrow_{sd}\nu_{B,j'}$. As μ is an atomic computation, either $t_1\not\rightarrow_{tr1}t_3$ or $t_3\not\rightarrow_{tr1}t_2$. (If $t_1\rightarrow_{tr1}t_3$ and $t_3\rightarrow_{tr1}t_2$, then t_3 has to be both a receive transition and a send transition, respectively. This is impossible by the definition of an atomic computation.) Consider the case $t_1\not\rightarrow_{tr1}t_3$. In step 1.2.2.3, as t_1 and t_3 are related by $(\rightarrow_{tr1})^*$, there is a transition t_4 such that the induction hypothesis can be applied to $\langle t_4;t_3\rangle$, and the proof can be completed.

⟨1⟩1. $\langle t_1;t_2\rangle\in\rightarrow_{tr}=(\rightarrow_{tr1})^*=\bigcup_{n\geq 0}\Psi^n(\varnothing)$, with $\Psi=\Psi_{\rightarrow tr1}$ $\Rightarrow$ $\langle\nu_{A,i};\nu_{B,j'}\rangle\in\rightarrow_{sd}=(\rightarrow_{sd1})^*=\bigcup_{n\geq 0}\Phi^n(\varnothing)$, with $\Phi=\Phi_{\rightarrow sd1}$, and $\nu_{B,j'}\neq next_{\mu(A)}(\nu_{A,i})$.

Assume: $\langle t_1;t_2\rangle\in\rightarrow_{tr}=(\rightarrow_{tr1})^*=\bigcup_{n\geq 0}\Psi^n(\varnothing)$.

Prove: $\langle\nu_{A,i};\nu_{B,j'}\rangle\in\rightarrow_{sd}=(\rightarrow_{sd1})^*=\bigcup_{n\geq 0}\Phi^n(\varnothing)$ and $\nu_{B,j'}\neq next_{\mu(A)}(\nu_{A,i})$.

⟨2⟩1. There is an $n\geq 0$ such that $\langle t_1;t_2\rangle\in\Psi^n(\varnothing)$.
Proof: by assumption ⟨1⟩.

⟨2⟩2. $\forall m\geq 0$: if $\langle t_1;t_2\rangle\in\Psi^m(\varnothing)$ then $(\langle\nu_{A,i};\nu_{B,j'}\rangle\in(\rightarrow_{sd1})^*$ and $\nu_{B,j'}\neq next_{\mu(A)}(\nu_{A,i}))$.

⟨3⟩1. Case: $m=0$.
Proof: by definition, $\Psi^m(\varnothing)$ for $m=0$ is the empty set, because for $m=0$, Ψ^m is the identity function.

⟨3⟩2. Case: 1. $m>0$,
2. $\forall m'$ with $0<m'<m$, $\langle t_1;t_2\rangle\in\Psi^{m'}(\varnothing)\Rightarrow(\langle\nu_{A,i};\nu_{B,j'}\rangle\in(\rightarrow_{sd1})^*$ and $\nu_{B,j'}\neq next_{\mu(A)}(\nu_{A,i}))$,
3. $\langle t_1;t_2\rangle\in\Psi^m(\varnothing)=\Psi(\Psi^{m-1}(\varnothing))=\{\langle t';t''\rangle\mid t'\rightarrow_{tr1}t''\}\cup\Psi^{m-1}(\varnothing)\cup\{\langle t_x;t_z\rangle\mid\langle t_x;t_y\rangle,\langle t_y;t_z\rangle\in\Psi^{m-1}(\varnothing)\}$.

⟨4⟩1. Case: $t_1 \rightarrow_{tr1} t_2$.

⟨5⟩1. Case: $A=B$ and $next_{\mu(A)}(v_{A,i})=prev_{\mu(B)}(v_{B,j'})$.

Proof: by assumption ⟨5⟩, $v_{A,i} \rightarrow_{sd1} next_{\mu(A)}(v_{A,i})$ and $next_{\mu(A)}(v_{A,i}) \rightarrow_{sd1} v_{B,j'}$. Because $\rightarrow_{sd1} \subseteq (\rightarrow_{sd1})^*$, $\langle v_{A,i}; next_{\mu(A)}(v_{A,i})\rangle, \langle next_{\mu(A)}(v_{A,i}); v_{B,j'}\rangle \in (\rightarrow_{sd1})^*$ and because $(\rightarrow_{sd1})^*$ is transitive, $\langle v_{A,i}; v_{B,j'}\rangle \in (\rightarrow_{sd1})^*$. By assumption ⟨5⟩, $v_{B,j'} \neq next_{\mu(A)}(v_{A,i})$.

⟨5⟩2. Case: $\langle\langle v_{A,i}; next_{\mu(A)}(v_{A,i})\rangle; \langle v_{L,i''}; v_{L,j''}; v_{L,k}; v_{L,l}\rangle; \langle prev_{\mu(B)}(v_{B,j'}); v_{B,j'}\rangle\rangle$ is a transmission octet.

Proof: by definition of $\rightarrow_{sd1}$.

⟨5⟩3. Q.E.D.

Proof: by the definition of $\rightarrow_{tr1}$, steps ⟨5⟩1 and ⟨5⟩2 list all cases.

⟨4⟩2. Case: $\langle t_1; t_2\rangle \in \Psi^{m-1}(\varnothing)$.

Proof: by assumption ⟨3⟩:2.

⟨4⟩3. Case: $\exists t_3: \langle t_1; t_3\rangle \in \Psi^{m-1}(\varnothing)$ and $\langle t_3; t_2\rangle \in \Psi^{m-1}(\varnothing)$.

⟨5⟩1. Let $t_3=trans(v_{C,z}, next_{\mu(C)}(v_{C,z}))$. $\langle v_{A,i}; next_{\mu(C)}(v_{C,z})\rangle$, $\langle v_{C,z}; v_{B,j'}\rangle \in (\rightarrow_{sd1})^*$.

Proof: by assumption ⟨3⟩:2.

⟨5⟩2. $\neg(t_1 \rightarrow_{tr1} t_3 \wedge t_3 \rightarrow_{tr1} t_2)$.

Proof: assume that $t_1 \rightarrow_{tr1} t_3$, $t_3 \rightarrow_{tr1} t_2$ and $A \neq B \neq C$. Then Γ cannot be left and right unique as required by Proposition 7.17.

⟨5⟩3. Case: $\neg(t_1 \rightarrow_{tr1} t_3)$.

⟨6⟩1. $\exists t_4$: $t_4=trans(v_{C,z'}, next_{\mu(C)}(v_{C,z'}))$: $t_1 \rightarrow_{tr1} t_4$ and $\langle t_4; t_3\rangle \in \Psi^{m-2}(\varnothing)$.

Proof: by the assumption that Γ is left and right unique and assumptions ⟨4⟩ and ⟨5⟩ and the definition of $\Psi^{m-1}(\varnothing)$.

⟨6⟩2. $\langle v_{A,i}; next_{\mu(C)}(v_{C,z'})\rangle \in \rightarrow_{sd1} \subseteq (\rightarrow_{sd1})^*$.

Proof: by step ⟨6⟩1 and the definitions of $\rightarrow_{tr1}$ and $\rightarrow_{sd1}$.

⟨6⟩3. $v_{C,z}=next_{\mu(C)}(v_{C,z'})$ or $\langle next_{\mu(C)}(v_{C,z'}); v_{C,z}\rangle \in (\rightarrow_{sd1})^*$.

Proof: by assumption ⟨3⟩:2, $\langle v_{C,z'}; next_{\mu(C)}(v_{C,z})\rangle \in (\rightarrow_{sd1})^*$, and the definition of $\rightarrow_{sd1}$.

⟨6⟩4. Q.E.D.

Proof: by transitivity of $(\rightarrow_{sd1})^*$ and steps ⟨5⟩1, ⟨6⟩2 and both cases of step ⟨6⟩3.

⟨5⟩4. Case: $\neg(t_3 \rightarrow_{tr1} t_2)$.
Proof: analogous to proof of previous step.
⟨5⟩5. Q.E.D.
Proof: steps ⟨5⟩3 and ⟨5⟩4 list both cases for step ⟨5⟩2.
⟨4⟩4. Q.E.D.
Proof: by assumption ⟨3⟩:3, steps ⟨4⟩1, ⟨4⟩2 and ⟨4⟩3 list all cases.
⟨3⟩3. Q.E.D.
Proof: by steps ⟨3⟩1 and ⟨3⟩2 and induction to n.
⟨2⟩3. Q.E.D.
Proof: by steps ⟨2⟩1 and ⟨2⟩2.

⟨1⟩2. $(\langle v_{A,i}; v_{B,j'}\rangle \in (\rightarrow_{sd1})^* = \bigcup_{n\geq 0} \Phi^n(\varnothing)$, with $\Phi = \Phi_{sd1}$, and $v_{B,j'} \neq next_{\mu(A)}(v_{A,i}) \Rightarrow \langle t_1; t_2\rangle \in (\rightarrow_{tr1})^* = \bigcup_{n\geq 0} \Psi^n(\varnothing)$, with $\Psi = \Psi_{tr1}$

Assume: $\langle v_{A,i}; v_{B,j'}\rangle \in (\rightarrow_{sd1})^* = \bigcup_{n\geq 0} \Phi^n(\varnothing)$ and $v_{B,j'} \neq next_{\mu(A)}(v_{A,i})$.

Prove: $\langle t_1; t_2\rangle \in (\rightarrow_{tr1})^* = \bigcup_{n\geq 0} \Psi^n(\varnothing)$.

⟨2⟩1. There is an $n \geq 0$ such that $\langle v_{A,i}; v_{B,j'}\rangle \in \Phi^n(\varnothing)$ and $v_{B,j'} \neq next_{\mu(A)}(v_{A,i})$.
Proof: by assumption ⟨1⟩.
⟨2⟩2. $\forall m \geq 0$: if $\langle v_{A,i}; v_{B,j'}\rangle \in \Phi^m(\varnothing)$ and $v_{B,j'} \neq next_{\mu(A)}(v_{A,i})$, then $\langle t_1; t_2\rangle \in (\rightarrow_{tr1})^*$.
⟨3⟩1. Case: $m=0$.
Proof: by definition, $\Phi^m(\varnothing)$ for $m=0$ is the empty set, because for $m=0$, Φ^m is the identity function.
⟨3⟩2. Case: 1. $m>0$,
2. if $\langle v_{A,i}; v_{B,j'}\rangle \in \Phi^{m-1}(\varnothing)$ and $v_{B,j'} \neq next_{\mu(A)}(v_{A,i}))$, then $\langle t_1; t_2\rangle \in (\rightarrow_{tr1})^*$,
3. $v_{B,j'} \neq next_{\mu(A)}(v_{A,i})$
4. $\langle v_{A,i}; v_{B,j'}\rangle \in \Phi^m(\varnothing) = \Phi(\Phi^{m-1}(\varnothing)) =$
$\{\langle v_{A,o}; v_{B,p}\rangle \mid v_{A,o} \rightarrow_{sd1} v_{B,p}\} \cup \Phi^{m-1}(\varnothing) \cup$
$\{\langle v_{A,x}; v_{B,y}\rangle \mid \langle v_{A,x}; v_{C,z}\rangle, \langle v_{C,z}; v_{B,y}\rangle \in \Phi^{m-1}(\varnothing)\}$.
⟨4⟩1. Case: $v_{A,i'} \rightarrow_{sd1} v_{B,j'}$.
Proof: by the definition of $\rightarrow_{sd1}$ and assumption ⟨3⟩:3,
$\langle\langle v_{A,i}; next_{\mu(A)}(v_{A,i})\rangle; \langle v_{L,i''}; v_{L,j''}; v_{L,k}; v_{L,l}\rangle;$
$\langle prev_{\mu(B)}(v_{B,j'}); v_{B,j'}\rangle\rangle$ is a transmission octet. By the definition of $\rightarrow_{tr1}$, $\langle t_1; t_2\rangle \in \rightarrow_{tr1} \subseteq (\rightarrow_{tr1})^*$.
⟨4⟩2. Case: $\langle v_{A,i}; v_{B,j'}\rangle \in \Phi^{m-1}(\varnothing)$.
Proof: by assumptions ⟨3⟩:2 and ⟨3⟩:3.
⟨4⟩3. Case: $\exists v_{C,z}$: $\langle v_{A,i}; v_{C,z}\rangle, \langle v_{C,z}; v_{B,j'}\rangle \in \Phi^{m-1}(\varnothing)$.
⟨5⟩1. Case: $v_{C,z} \neq next_{\mu(A)}(v_{A,i})$ and $v_{B,j'} \neq next_{\mu(C)}(v_{C,z})$.
⟨6⟩1. $\langle t_1; trans(prev_{\mu(C)}(v_{C,z}), v_{C,z})\rangle \in (\rightarrow_{tr1})^*$.
Proof: by assumptions ⟨3⟩:2 and ⟨5⟩.

⟨6⟩2. $\langle trans(v_{C,z}, next_{\mu(C)}(v_{C,z})); t_2\rangle \in (\rightarrow_{tr1})^*$.
Proof: by assumptions ⟨3⟩:2 and ⟨5⟩.

⟨6⟩3. $\langle trans(prev_{\mu(C)}(v_{C,z}), v_{C,z});$
$trans(v_{C,z}, next_{\mu(C)}(v_{C,z}))\rangle \in \rightarrow_{tr1} \subseteq (\rightarrow_{tr1})^*$.
Proof: by definition of $\rightarrow_{tr1}$.

⟨6⟩4. Q.E.D.
Proof: by steps ⟨6⟩1, ⟨6⟩2 and ⟨6⟩3 and transitivity of $(\rightarrow_{tr1})^*$.

⟨5⟩2. Case: $v_{C,z}=next_{\mu(A)}(v_{A,i})$ or $v_{B,j'}=next_{\mu(C)}(v_{C,z})$.
Proof: similar to the proof of the other case. The induction hypothesis is only applied once or not at all. Instead, if e.g. $v_{C,z}=next_{\mu(A)}(v_{A,i})$, then $\langle t_1; trans(v_{C,z}, next_{\mu(C)}(v_{C,z}))\rangle \in (\rightarrow_{tr1})^*$ by clause 1 of the definition of $\rightarrow_{tr1}$.

⟨5⟩3. Q.E.D.
Proof: steps ⟨5⟩1 and ⟨5⟩2 list all cases.

⟨4⟩4. Q.E.D.
Proof: by assumption ⟨3⟩:4, steps ⟨4⟩1, ⟨4⟩2 and ⟨4⟩3 list all cases.

⟨3⟩3. Q.E.D.
Proof: by steps ⟨3⟩1 and ⟨3⟩2 and induction to n.

⟨2⟩3. Q.E.D.
Proof: by steps ⟨2⟩1 and ⟨2⟩2.

⟨1⟩3. Q.E.D.
Proof: by steps ⟨1⟩1 and ⟨1⟩2.

Proposition 7.19. *Let SH be a structure hierarchy and let μ be an atomic computation for SH. Then $v_{A,i} \rightarrow_{sd} v_{B,j'}$ iff $v_{A,i} \rightarrow_{fr} v_{B,j'}$.*

Proof. Proof sketch: The proof consists of straightforward expansions of the definitions involved.

⟨1⟩1. $v_{A,i} \rightarrow_{sd} v_{B,j'} \Rightarrow v_{A,i} \rightarrow_{fr} v_{B,j'}$.
Assume: $v_{A,i} \rightarrow_{sd} v_{B,j'}$.
Prove: $v_{A,i} \rightarrow_{fr} v_{B,j'}$.

⟨2⟩1. Case: $A=B$ and $v_{A,i}=prev_{\mu(A)}(v_{B,j'})$.
Proof: by clause 1 of the definition of Fromentin & Raynal.

⟨2⟩2. Case: $\langle\langle v_{A,i}; next_{\mu(A)}(v_{A,i})\rangle; \langle v_{L,i''}; v_{L,j''}; v_{L,k}; v_{L,l}\rangle; \langle prev_{\mu(B)}(v_{B,j'}); v_{B,j'}\rangle\rangle$ is a transmission octet for LT_A, LT_L and LT_B.
Proof: by Proposition 7.17, $trans(v_{A,i}, next_{\mu(A)}(v_{A,i})) \rightarrow_{tr}$ $trans(prev_{\mu(B)}(v_{B,j'}), v_{B,j'})$. By clause 2 of the definition of Fromentin & Raynal, $v_{A,i} \rightarrow_{fr} v_{B,j'}$.

⟨2⟩3. Case: there is a state $v_{C,m}$ such that $v_{A,i} \rightarrow_{sd} v_{C,m'}$ and $v_{C,m} \rightarrow_{sd} v_{B,j'}$.

Proof: by applying Proposition 7.17 twice and clause 3 of the definition of Fromentin and Raynal.

⟨2⟩4. Q.E.D.
Proof: steps ⟨2⟩1, ⟨2⟩2 and ⟨2⟩3 list all cases.

⟨1⟩2. $\nu_{A,i} \rightarrow_{fr} \nu_{B,j'} \Rightarrow \nu_{A,i} \rightarrow_{sd} \nu_{B,j'}$.
Assume: $\nu_{A,i} \rightarrow_{fr} \nu_{B,j'}$.
Prove: $\nu_{A,i} \rightarrow_{sd} \nu_{B,j'}$.
⟨2⟩1. Case: $A=B$ and $\nu_{A,i}=prev_{\mu(A)}(\nu_{B,j'})$.
Proof: by clause 1 of the definition of strict dependence.
⟨2⟩2. Case: $trans(\nu_{A,i}, next_{\mu(A)}(\nu_{A,i})) \rightarrow_{tr} trans(prev_{\mu(B)}(\nu_{B,j'}), \nu_{B,j'})$.
Proof: by Proposition 7.17, $\nu_{A,i} \rightarrow_{sd} \nu_{B,j'}$.
⟨2⟩3. Q.E.D.
Proof: steps ⟨2⟩1 and ⟨2⟩2 list all cases.

⟨1⟩3. Q.E.D.
Proof: by steps ⟨1⟩1 and ⟨1⟩2.

Proposition 7.25. *Let $SH=(Comp;Lnk;\prec;dom;cdom)$ be a structure hierarchy, let $\gamma=(\gamma_l)_{l \in Lnk}$ be a collection of compatibility relations, let μ be a multitrace compatible for γ and let $cut(SH,\mu,\gamma)$ be a cut for SH with respect to μ and γ. If $cut(SH,\mu,\gamma)$ is a consistent cut, then $fs(cut(SH,\mu,\gamma))$ is a global state.*

Proof. Proof sketch: The contraposition of the proposition is proven. Thus, $fs(cut(SH,\mu,\gamma))$ is assumed not to be a strict global state. Then there must be two components or links S_1,S_2 such that the state of S_1 in $fs(cut(SH,\mu,\gamma))$ is strictly dependent on the state of S_2 or the other way around. Using induction, it is proven that in either case, there are transitions such that $cut(SH,\mu,\gamma)$ cannot be a consistent cut.
Assume: $fs(cut(SH,\mu,\gamma))$ is not a strict global state.
Prove: $cut(SH,\mu,\gamma)$ is not a consistent cut.
⟨1⟩1. Let $\sigma=fs(cut(SH,\mu,\gamma))$ and let $S_1,S_2 \in Comp \cup Lnk$. Then $\sigma(S_1) \rightarrow_{sd} \sigma(S_2)$ or $\sigma(S_2) \rightarrow_{sd} \sigma(S_1)$.
Proof: by assumption ⟨0⟩ and the definition of a strict global state.
⟨1⟩2. Case: $\sigma(S_1) \rightarrow_{sd} \sigma(S_2)$.
Assume: $\langle \sigma(S_1);\sigma(S_2) \rangle \in \rightarrow_{sd} = (\rightarrow_{sd1})^* = \bigcup_{n \geq 0} \Phi^n(\emptyset)$ with $\Phi=\Phi_{sd1}$.
Prove: $cut(SH,\mu,\gamma)$ is not consistent.
⟨2⟩1. $\exists n: \langle \sigma(S_1);\sigma(S_2) \rangle \in \Phi^n(\emptyset)$.
Proof: assumption ⟨1⟩.
⟨2⟩2. $\forall m \geq 0:$ if $\langle \sigma(S_1);\sigma(S_2) \rangle \in \Phi^m(\emptyset)$, then $cut(SH,\mu,\gamma)$ is not consistent.
⟨3⟩1. Case $m=0$.
Proof: by definition, $\Phi^0(\emptyset)$ is the empty set, because for $m=0$, Φ^m is the identity function. Thus, for this case, the claim is trivially true.

⟨3⟩2. Case: 1. m>0,
2. $\langle\sigma(S_1);\sigma(S_2)\rangle\in\Phi^{m-1}(\varnothing) \Rightarrow cut(SH,\mu,\gamma)$ is not consistent.
3. $\langle\sigma(S_1);\sigma(S_2)\rangle\in\Phi^{m}(\varnothing)=\Phi(\Phi^{m-1}(\varnothing))=$
$\{\langle v_{S_1};v_{S_2}\rangle \mid v_{S_1}\rightarrow_{sd1} v_{S_2}\}\cup \Phi^{m-1}(\varnothing)\cup$
$\{\langle\langle v_{S_1};v_{S_2}\rangle \mid \langle v_{S_1};v_{S_3}\rangle,\langle v_{S_3};v_{S_2}\rangle\in\Phi^{m-1}(\varnothing)\}$.

⟨4⟩1. Case: $\langle\sigma(S_1);\sigma(S_2)\rangle\in\{\langle x;y\rangle \mid x\rightarrow_{sd1} y\}$

⟨5⟩1. Case: $S_1=S_2$ and $\sigma(S_2)=next_{\mu(S_1)}(\sigma(S_1))$.

Proof: the case is empty, as by the definition of *fs*, it is impossible that $\sigma(S_2)=next_{\mu(S_1)}(\sigma(S_1))$, while $\sigma=fs(cut(SH,\mu,\gamma)$.

⟨5⟩2. Case: $\langle\langle\sigma(S_1);next_{\mu(S_1)}(\sigma(S_1))\rangle;\langle v_{I,i''};v_{I,j''};v_{I,k};v_{I,'}\rangle;$ $\langle prev_{\mu(S_2)}(\sigma(S_2));\sigma(S_2)\rangle$ is a transmission octet for a link *I* from S_1 to S_2 and traces $\mu(S_1)$, $\mu(I)$ and $\mu(S_2)$.

⟨6⟩1. $t_1=trans(\sigma(S_1),next_{\mu(S_1)}(\sigma(S_1)))\rightarrow_{tr} trans(prev_{\mu(S_2)}(\sigma(S_2));\sigma(S_2))=t_2$.

Proof: by definition of $\rightarrow_{tr}$.

⟨6⟩2. $t_1\notin cut^*(SH,\mu,\gamma)$.

Proof: by the definition of the final state set and of a cut.

⟨6⟩3. Q.E.D.

Proof: $t_2=cut(SH,\mu,\gamma)(S_2)$ by definition of a cut, $t_1\rightarrow_{tr} t_2$ by step ⟨6⟩1 and $t_1\notin cut^*(SH,\mu,\gamma)$ by step ⟨6⟩2.

⟨5⟩3. Q.E.D.

Proof: steps ⟨5⟩1 and ⟨5⟩2 list all cases for assumption ⟨4⟩.

⟨4⟩2. Case: $\langle\sigma(S_1);\sigma(S_2)\rangle\in\Phi^{m-1}(\varnothing)$.

Proof: by assumption ⟨3⟩:2.

⟨4⟩3. Case: $\langle\sigma(S_1);\sigma(S_2)\rangle\in\{\langle x;z\rangle \mid \langle x,y\rangle,\langle y;z\rangle\in\Phi^{m-1}(\varnothing)\}$.

Proof: Let ν be an arbitrary state. By assumption ⟨3⟩:2, if $\langle\sigma(S_1);\nu\rangle\in\Phi^{m}(\varnothing)$, then $cut(SH,\mu,\gamma)$ is not consistent. By assumption ⟨4⟩, there is a ν such that $\langle\sigma(S_1);\nu\rangle\in\Phi^{m}(\varnothing)$.

⟨4⟩4. Q.E.D.

Proof: steps ⟨4⟩1, ⟨4⟩2 and ⟨4⟩3 list all cases for assumption ⟨3⟩:3.

⟨3⟩3. Q.E.D.

Proof: by steps ⟨3⟩1 and ⟨3⟩2 and induction to *m*.

⟨2⟩3. Q.E.D.

Proof: by steps ⟨2⟩1 and ⟨2⟩2.

⟨1⟩3. Case: $\sigma(S_1)\rightarrow_{sd}\sigma(S_2)$.

Proof: analogous to ⟨1⟩2.

⟨1⟩4. Q.E.D.

Proof: steps ⟨1⟩2 and ⟨1⟩3 list all cases for ⟨1⟩1.

Proposition 7.27. *Let $SH=(Comp;Lnk;\prec;dom;cdom)$ be a structure hierarchy, let $\gamma=(\gamma_I)_{I\in Lnk}$ be a collection of compatibility relations, let μ be a multitrace compatible for γ and let $\sigma\in SNAP(SH,\mu,\gamma)$ be a snapshot of SH for μ and γ. If σ is a strict global state of a structure hierarchy SH, then $fts(\sigma)$ is a consistent cut.*

Proof. Proof sketch: The contraposition of the proposition is proven. Thus, $fts(\sigma)$ is assumed not to be a consistent cut. Then there must be two transitions t,t' of components or links S_1,S_2 such that t' depends on t. Using induction, it is proven that in this case, $\sigma(S_1)$ strictly depends on $\sigma(S_2)$ or the other way around. In both cases, σ is not a strict global state.

Assume: $fts(\sigma)$ is not a consistent cut.

Prove: σ is not a strict global state.

⟨1⟩1. There is an $S\in Comp\cup Lnk$ and there is a $t'\in\mathcal{T}_{SH}$ such that $t'\rightarrow_{tr}t=fts(\sigma)(S)$ and $t'\notin fts^*(\sigma)$.

Proof: by assumption ⟨0⟩ and the definition of a consistent cut.

⟨1⟩2. Let $S\in Comp\cup Lnk$ and let $t'\in\mathcal{T}_{SH}$. If $t'\rightarrow_{tr}t=fts(\sigma)(S)$ and $t'\notin fts^*(\sigma)$, then σ is not a strict global state.

Assume: 1. $\langle t';t\rangle\in\rightarrow_{tr}=(\rightarrow_{tr1})^*=\bigcup_{n\geq 0}\Phi^n(\emptyset)$ with $\Phi=\Phi_{\rightarrow_{tr1}}$, 2. $t'\notin fts^*(\sigma)$.

Prove: σ is not a strict global state.

⟨2⟩1. $\exists n:\ \langle t';t\rangle\in\Phi^n(\emptyset)$.

Proof: assumption ⟨1⟩:1.

⟨2⟩2. $\forall m\geq 0:$ if $\langle t';t\rangle\in\Phi^m(\emptyset)$, then σ is not a strict global state.

⟨3⟩1. Case $m=0$.

Proof: by definition, $\Phi^0(\emptyset)$ is the empty set, because for $m=0$, Φ^m is the identity function. Thus, for this case, the claim is trivially true.

⟨3⟩2. Case: 1. m>0,

2. $\langle t';t\rangle\in\Phi^{m-1}(\emptyset)\Rightarrow\sigma$ is not a strict global state.

3. $\langle t';t\rangle\in\Phi^m(\emptyset)=\Phi(\Phi^{m-1}(\emptyset))=$
$\{\langle t';t\rangle\mid t'\rightarrow_{tr1}t\}\cup\Phi^{m-1}(\emptyset)\cup\{\langle\langle t';t\rangle\mid\langle t';t''\rangle,\langle t'';t\rangle\in\Phi^{m-1}(\emptyset)\}$

⟨4⟩1. Case: $\langle t';t\rangle\in\{\langle t';t\rangle\mid t'\rightarrow_{tr1}t\}$.

⟨5⟩1. Let $t'=trans(\langle v_{S_1};next_{\mu(S_1)}(v_{S_1})\rangle)$ and let $t=trans(prev_{\mu(S_2)}(\sigma(S_2)),\sigma(S_2))$.

Case: $S_1=S_2$ and $next_{\mu(S_1)}(v_{S_1})=prev_{\mu(S_2)}(\sigma(S_2))$.

Proof: the case is empty, because by ass. ⟨5⟩ and the definition of $fts^*(\sigma)$, $t'\in fts^*(\sigma)$, which contradicts assumption ⟨1⟩:2.

⟨5⟩2. Case: $\exists l \in Lnk$: $dom(I)=S_1$ and $cdom(I)=S_2$ and $\exists v_{I,i''}, v_{I,j''}, v_{I,k}, v_{I,l}$: $\langle\langle v_{S_1}; next_{\mu(S_1)}(v_{S_1})\rangle; \langle v_{I,i''}; v_{I,j''}; v_{I,k}; v_{I,l}\rangle; \langle prev_{\mu(S_2)}(\sigma(S_2)); \sigma(S_2)\rangle\rangle$ *is a transmission octet for* I, $\mu(S_1)$, $\mu(I)$ and $\mu(S_2)$.

⟨6⟩1. $v_{S_1} \rightarrow_{sd} \sigma(S_2)$.

Proof: by assumption ⟨5⟩ and the definition of strict dependence.

⟨6⟩2. $\sigma(S_1) \rightarrow_{sd} v_{S_1}$ or $\sigma(S_1)=v_{S_1}$.

Proof: by assumption ⟨1⟩:2, $t'=trans(\langle v_{S_1}; next_{\mu(S_1)}(v_{S_1})) \notin fts^*(\sigma)$. By the definition of $fts^*(\sigma)$ and of $\rightarrow_{sd}$, v_{S_1} must be later than $\sigma(S_1)$.

⟨6⟩3. Q.E.D.

Proof: by steps ⟨6⟩1, ⟨6⟩2 and transitivity of $\rightarrow_{sd}$.

⟨5⟩3. Q.E.D.

Proof: steps ⟨4⟩1 and ⟨4⟩2 list all cases for assumption ⟨4⟩.

⟨4⟩2. Case: $\langle t';t\rangle \in \Phi^{m-1}(\varnothing)$.

Proof: by assumption ⟨3⟩:2.

⟨4⟩3. Case: $\langle t';t\rangle \in \{\langle t';t\rangle \mid \langle t',t''\rangle, \langle t'';t\rangle \in \Phi^{m-1}(\varnothing)\}$.

Proof: Let t'' be an arbitrary state. By assumption ⟨3⟩:2, if $\langle t';t''\rangle \in \Phi^{m}(\varnothing)$, then σ is not a strict global state. By assumption ⟨3⟩, there is a t'' such that $\langle t';t''\rangle \in \Phi^{m}(\varnothing)$.

⟨4⟩4. Q.E.D.

Proof: steps ⟨4⟩1, ⟨4⟩2 and ⟨4⟩3 list all cases for assumption ⟨3⟩:3.

⟨3⟩3. Q.E.D.

Proof: steps ⟨3⟩1 and ⟨3⟩2 and induction to m.

⟨2⟩3. Q.E.D.

Proof: steps ⟨2⟩1 and ⟨2⟩2.

⟨1⟩3. Q.E.D.

Proof: by steps ⟨1⟩1 and ⟨1⟩2.

Chapter 8

Control in Compositional Systems and Multi-Agent Systems

This chapter is dedicated to the phenomenon of control in compositional systems and multi-agent systems. The aim of this chapter is to develop constructs for modelling control in multi-agent systems as a refinement of the semantic structure presented in the previous chapters. In other words, in this chapter, new commitments are introduced that further characterise the constructs provided by the semantic structure. These commitments determine how one component in a compositional system can control another component. Furthermore, additional modelling choices are presented that illustrate how control in a compositional system can be used to model control in a multi-agent system.

The previous chapters discuss the influence of information exchange between components on the dynamics of the components. Under specific conditions, this influence enables one component to control other components. Thus, the semantic structure presented in the previous chapters provides necessary constructs to support control. However, to apply the semantic structure, refinements of the constructs provided by the semantic structure are needed that resolve the following issues. First, although the semantic structure presented in the previous chapters ensures *that* one component can influence another component, the question of *what* is influenced and *what* this influence entails, is not addressed. Second, the constructs provided by the semantic structure do not address questions of how control can be exercised in a *domain-independent* and *compositional* way. Third, the *scope* of influence exercised by each individual component in a compositional system is not clear. These issues are addressed in this chapter.

The structure of this chapter is as follows. To provide a strong foundation from which to pursue the aim of this chapter, first, in Section 8.1, the control phenomenon is characterised, the role of control in compositional systems is discussed, and some perspectives from other research areas are presented. The next two sections have a structure similar to Chapter 2 and Chapter 3. Thus,

Section 8.2 presents commitments with respect to how control in compositional systems is represented. This section explains how constructs provided by the semantic structure can exercise control over components. Section 8.3 discusses control in multi-agent systems. Although the autonomy of an agent in a multi-agent system seems to be incompatible with control over that agent, the characterisation of control presented in Section 8.1 is also applicable to an autonomous agent. Section 8.3 presents some modelling choices with respect to how control in multi-agent systems can be represented in a compositional system that models the multi-agent system. Section 8.4 provides an example of the material presented in the previous sections. The final topic is the relationship between controlling components and the components they control. The notion of global state developed in the previous chapter is exploited to investigate this relation. This topic is covered in Section 8.5.

8.1 Control and Compositionality

The phenomenon of control appears in many different guises, both in multi-agents systems as well as in conventional Computer Science and Artificial Intelligence. The control phenomenon is encountered whenever a specific part of a system adopts a goal that requires *another part* of the system to reach a specific state. In other words, a part of a system that tries to constrain the future of another part such that this other part reaches a specific state, has to exercise control over the part that is to be constrained. For instance, in a multi-agent system with robot agents, one of the robots may adopt a goal to change something in the environment (e.g., to paint a car in an assembly line), or to change the state of another robot such that this robot moves out of the way. In both cases, the agent faces a control task: it needs to execute actions in the environment, perform observations or exchange information, to exercise influence with the effect of changing the current state of the environment or agent to the desired state.

The previous paragraph deliberately characterises control in terms of *parts* of a system. Indeed, control is exercised in all kinds of systems, by different kinds of parts. In a compositional system, one component may control other components. In a multi-agent system, one agent may control other agents.

Although the control phenomenon appears in many different guises, it is possible to develop a common characterisation of control tasks, as is indicated in Chandrasekaran (1994). Chandrasekaran studies exercising control from a knowledge-level point of view. His goal is to answer the following question: "What is it that unifies the control task in all its manifestations, from the thermostat to the operator of a nuclear power plant?" (Chandrasekaran, 1999). The answer he derives consists of a process model of control, and provides a general characterisation of what comprises exercising control. The general process model for control consists of the following subprocesses. First, either implicitly or explicitly, by observing the controlled components, a controlling component

constructs a descriptive model of the past and present behaviour of the controlled components. The control component then derives a prescriptive extension of this model, for instance by instantiating a (possibly pre-compiled) plan. This extension describes the intended future behaviour of the controlled components. Using this extension, the control component tries to influence the behaviour of the controlled components by transmitting information to them. After that, the control component may obtain new observations with which the model of the past and present behaviour of the controlled components can be updated, after which the control process is repeated.

As the quotation from Chandrasekaran suggests, the control phenomenon spans a very diverse spectrum of domains, from a thermostat to a nuclear power plant operator or the president of a national bank. According to Chandrasekaran, in all of these domains, exercising control has a similar structure, summarised as follows: a controlling component repeatedly attempts to influence other components by transmitting information to these components, and uses information transmitted from these components as feedback. Thus, the essence of control is information exchange. The constructs provided by the semantic structure are sufficient to enable the construction of compositional systems that execute control tasks as characterised by Chandrasekaran because these constructs support information transmission.

In this chapter, the semantic structure is refined (i.e., additional properties of constructs and relations between constructs are distinguished and committed to) with the aim of providing *better support for control*. The refined semantic structure supports building compositional systems in which control is *separated* and as *domain-independent* as possible. As explained in Section 8.1.1, separated, domain-independent control provides a more maintainable and reusable structure. To separate control, the additional properties and relations for control require that control and control information can be distinguished in a compositional system. As, due to the many different guises of the control phenomenon, it is difficult to distinguish control information from other information by a general definition, another approach is adopted: users of the semantic structure can *designate* specific components and information to be control components and information. The additional properties and relations are applicable to the components and information designated to be specific for control. Designating control components and information is the subject of Section 8.1.2. Perspectives on control from other areas are presented in Section 8.1.3 The additional properties and relations are presented in Section 8.2.

8.1.1 Separate, Domain-independent control

In most branches of engineering, systems are partitioned in components to reduce overall complexity and enhance reuse, modification and maintenance. The extent to which these goals are met depends on how a specific system is partitioned. Therefore, an important research topic in Computer Science and Artificial

Intelligence is the investigation of design principles that guide software engineers in finding a good compositional structure. (Recent developments in this area take place within the study of software architecture (Perry & Wolf, 1992; Bass, Clements, Kazman & Bass, 1998), design patterns (Gamma, Helm, Johnson & Vlissides, 1995) and co-ordination languages (Papadopoulos & Arbab, 1998).)

A design principle applied in Computer Science and Artificial Intelligence is to separate a computation's logic and control, and make both equally explicit. Although this principle appears in many areas of Computer Science and Artificial Intelligence, it is probably most famous as the foundation for logic programming languages as presented in the paper by Kowalski (1979). In the area of knowledge based systems design, Clancey (1983, 1992) and Neches, Swartout and Moore (1985) emphasise the importance of separating domain knowledge (*what* knowledge to apply) from control knowledge (*how* to apply that knowledge). In generic architectures for knowledge based systems, control knowledge is recognised as a separate type of knowledge (Chandrasekaran, Johnson & Smith, 1992; Brazier, Treur & Wijngaards, 1996). In all of these areas, separating control knowledge results in systems that are more maintainable and easier to modify and reuse.

In a compositional system with a recursive structure of subcomponents, subcomponents of subcomponents, and so on, control is separated at each level. This implies that at each level, some components are control components, while other components are controlled components. In the terminology of Kowalski (1979), the controlled components contain the logic of the computation that is carried out collaboratively by the control components and controlled components. In the terminology of knowledge based systems, the controlled components contain domain knowledge such as, for instance, inference relations. In a compositional system, controlled components themselves actively perform computations and may themselves consist of control subcomponents and subcomponents controlled by the control subcomponents. These control subcomponents and controlled subcomponents collaboratively perform the computations of their parent component, which is a controlled component at its level. Thus, although in Kowalski's terms, a controlled component (only) contains the logic of a computation, if the controlled component is composed, there may be control inside the component.

Figure 8.1 illustrates separated control in a compositional system. In this figure, components drawn with thick lines are control components, while the other components are controlled components. All links depicted in Figure 8.1 are used to transmit control information. The figure shows that the control component that is a subcomponent of the controlled component receives control information and provides feedback information for the control component at its parent level. In Figure 8.1, the specific part of each input and output interface that is reserved for control information, is shown in gray.

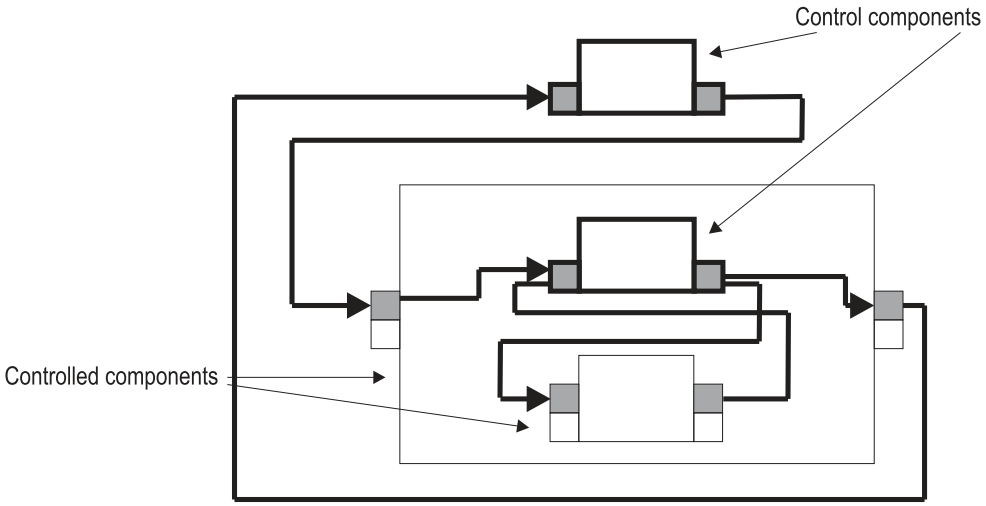

Figure 8.1: Control in a compositional system.

Control and control information separated in a compositional system should be domain independent to support maintainability and ease of modification and reuse. Computations performed by a component are specific for a particular domain. For instance, in a compositional system that represents a nuclear power plant, computations that represent the operator are e.g. co-operation with other operators in the plant and to diagnosis of reactor malfunctioning. These computations process domain specific information, such as e.g. information on the status of the reactor in the nuclear power plant or on the intentions of other operators. Control information should contain as little domain information as possible and should instead consist of domain-independent information on e.g. activating components and enabling information transmission. In other words, control information should be as domain-independent as possible. Note that, apart from control information, other information can also be domain-independent, or *generic*. As an example, consider mathematical knowledge, which is present in almost every domain, but is not control information.

8.1.2 Designating Control Information

The previous subsection presented reasons for separating control in a compositional system. This subsection explains *which* information to separate as control information. As stated at the beginning of Section 8.1, the control phenomenon is encountered whenever part of a system tries to influence another part of the system in an attempt to constrain the future of the other part. In a compositional system, these parts are components. Components use information exchange to influence each other. Therefore, all information exchange has a potential control effect. In other words, it is difficult to distinguish information

transmission for control from other information transmission. Consider, for example, the president of a national bank, who tries to control the economic behaviour of companies and customers. The bank president can influence the behaviour of companies and customers in several ways. One way is to change the official interest rate and transmit information on the new rate to all companies and customers. However, this information does not *force* a change of behaviour, as companies and customers are free to continue their behaviour. Nevertheless, an interest rate change is usually explained as an attempt to control the behaviour of domestic customers and companies. However, the interest rate may have been changed for other reasons, or other information provided by the bank president also changed the behaviour of customers and companies. As every information transmission changes the state of destination components, all information transmission has a potential control effect. (This example domain has been suggested by Chandrasekaran.)

For reasons put forward in the previous subsection, it is important to distinguish information transmission for control from other information transmission. As all information transmission has a potential control effect, an additional criterion has to be developed. There are two possibilities for such a criterion. The first possibility is to require that information transmissions must be performed *intentionally* for control to classify as transmissions of control information. In the example of the bank president, transmitting information on a new interest rate is only a transmission for controlling domestic customers and companies if the president has the intention of changing the economic behaviour of those domestic customers and companies and changed the interest rate because of this intention. This possibility requires insight in the intentions and relationship between intentions and transmitted information to distinguish information transmission for control from other information transmission. This might be suitable in a multi-agent system in which intentions of agents are explicitly modelled. However, in the more general context of compositional systems, this possibility is not realistic.

The other possibility is to avoid committing to a specific characterisation and instead provide a flexible way for users of the semantic structure to *designate* specific information as control information. Specific properties of constructs provided by the semantic structure and specific relations between constructs are only applicable to information designated as control information. Users of the semantic structure can choose which information they wish to subject to these specific properties and relations. Information is designated as control information by associating it, for example, with a reserved part of the input and output interfaces of a component. Consequently, for the state of a component, two substates can be distinguished: called the control state and the domain state.

The specific properties and relations for control provided by the semantic structure enable separating control that is as domain-independent as possible. Moreover, there is an additional property of information designated as control:

control information sent to a controlled component not merely influences this component, it also constrains the behaviour of the controlled component. Control information specifies one or more possible futures for the reserved control information substate of a controlled component. The properties and relations committed to (see Section 8.2) guarantee that the controlled component cannot choose another future but the specified one. This is the most important property of control information and should determine, for a specific application, which information is designated as control information.

As an example, the information on a new interest rate transmitted by the bank president should probably not be designated as information transmission for control: a specific customer might choose to neglect the changed interest rate and continue with his or her behaviour (although this might not be rational). As an example of information transmission that should be designated as a control transmission, consider a nuclear power plant operator. If he or she detects a situation that requires an immediate stop of the reactor, probably by pushing a button he or she commands a specific component of the plant to drop bars of a moderating material into the reactor. The content information of pushing the button (pushing the button is the encapsulated form of the content information, see Section 2.2.4) is: "drop the moderator", or "stop the reactor". The reactor cannot choose to neglect this command: pushing the button breaks a circuit to an electromagnetic device holding the bars containing the moderator. The laws of physics guarantee that the bars fall into the reactor, which slows down or stops. The specified future is not guaranteed: if the button is broken or if someone has removed the moderator bars, the future may not be as expected. However, this is beyond the reactor's control.

8.1.3 Some Perspectives on Control

Different perspectives on the concept of control can be found in related areas:

- In the context of (formal) languages for the specification of algorithms, such as programming languages, the concept of control refers to the sequencing of steps in an algorithm. The characterisation of control developed by Chandrasekaran is also applicable in this context. In his terminology, the specification or (compiled) computer program tries to influence a state space (e.g., a computer memory) to constrain the set of all possible contents of this space. In this context, operations to influence and observe the state space are generally assumed to be infallible (e.g., read and write operations in a computer). The specification or program text itself can be viewed as knowledge provided by a software engineer that is used by the compiled program to, in Chandrasekaran's terminology, derive a prescriptive extension of the model of the behaviour of the state space. This knowledge is formalised in different ways. In imperative programming languages, so-called *control structures* such as 'if … then' and 'do … while' are used. In

logic programming languages, control is fixed and implicit in the semantics of the language. In (Eck, Engelfriet, Fensel, Harmelen, Venema & Willems, in press), formalisms for specification of dynamics in Software Engineering and Information Systems are surveyed from a Knowledge Engineering perspective. They identify two major differences between the formalisms studied. First, there is a distinction between constructive and constraining control specifications. In a constructive specification, control is specified by stating how valid sequences of states are constructed, often in the style of an imperative programming language. In a constraining specification, control is specified by a set of expressions, such as, for instance, temporal logic formulae, that constrain the set of all possible sequences. Second, there is a distinction between step-based and sequence-based specification of control. In a step-based specification, control can only be described in terms of the relation between a begin state and an end state. In a sequence-based specification, control can also be described by reference to intermediate states.

- Control of one agent over another seems to be incompatible with the autonomy of agents in a multi-agent system. Indeed, a property of an autonomous agent is that it is impossible for other agents to unconditionally set the agent's goals in a way that is inescapable for that agent. However, the concept of control sketched at the beginning of Section 8.1 does not assume that this is possible: an agent that controls another agent merely tries to influence the controlled agent to adopt a specific goal. In the area of multi-agent systems, control is related to co-ordination and management. This topic is addressed further in Section 8.3.
- In the CommonKADS approach to Knowledge Engineering (Schreiber, Akkermans, Anjewierden, Hoog, Shadbolt, Velde & Wielinga, 1999), the control phenomenon appears in to some extent in most of the models that constitute the CommonKADS model suite. (The CommonKADS model suite consists of six models, divided over three levels. At the highest level, the organisation, task and agent models describe the environment in which knowledge-intensive systems are applied. The middle level consists of the knowledge model and the communications model, which describe knowledge-intensive tasks and information exchange in a conceptual and implementation-independent way. The lowest level consists of the design model, which describes the (computer) system that is constructed to implement the knowledge and communication models.) First, at the environment level, control is scattered over all three models. The agent model describes various attributes of agents, including competences, responsibilities and communication relations of individual agents. These attributes indicate, implicitly, possible control relations. The organisation model describes formal and informal power relations between agents in the

organisation. In a sense, this is a static view on control, which focuses on the question which agent has the power to control which other agent(s). The task model covers knowledge of time-related aspects of tasks that constitute the process modelled by a CommonKADS model. Thus, this is control *over* tasks. Presumably, specific agents have to exercise control to ensure proper timing and order of tasks. However, the CommonKADS approach does not provide constructs to make explicit which agents control which tasks. Second, control for knowledge-intensive tasks identified in the task model is described in the knowledge model. This is control *within* the task, and describes how the goals of a task can be achieved in terms of subtasks and inferences. This form of control is described using a pseudo-code language in an imperative style. However, control in the knowledge model is only used to describe the task at a conceptual level, in a sense to help characterise the task. In the knowledge model, the entity that is responsible for exercising control to ensure that subtasks and inferences are executed in the order specified, is not distinguished. Third, the design model covers control in the (computer) system that is constructed to implement the knowledge model (and communication model). The design model consists of an architecture design, an application design, and a platform design. There is no commitment to a specific form of control. Instead, properties of control depends on the choices made for the skeletal architecture (e.g., rule-based), the application design (e.g., search), and the platform (i.e., languages and support environment, such as OPS5). However, the preferred choice in the CommonKADS approach is a design model that preserves the structure of the knowledge and communication models. Such a structure-preserving design inherits properties of control from the knowledge and communication models.

8.2 Constructs for Control in Compositional Systems

Chapter 2 presented commitments that characterise the constructs for information transmission in the semantic structure developed in this thesis. In this section, a similar characterisation of additional commitments for support of control is presented. In the next subsections, the following commitments are discussed:

- All control is explicitly represented in the compositional system. If, for any component, control is to be exercised, a component responsible for exercising this control is appointed (Section 8.2.1);
- Control is exercised by components that do not have a domain-dependent function (Section 8.2.2);
- For each component, a part of the state of both the component's input and output is distinguished as the control substate (Section 8.2.3);

- Control information is transmitted using dedicated links, called *control links* (which, apart from their dedication, are normal links). These control links connect control components and (the control part of the interfaces of) the controlled components (Section 8.2.4);
- A control component can directly activate control links to which it is connected (Section 8.2.5);
- Control information constrains the future of controlled components (Section 8.2.6).

8.2.1 Component Responsible for Control

The control phenomenon is often encountered in the following way. During analysis or design of a multi-agent system, the need for a specific order of subprocesses within an agent is identified. Control is needed for this purpose. The question is: which, if any, process is responsible for exercising the control needed?

In a compositional system that represents the multi-agent system, each component may, in principle, be active completely independent from all other components. There is no 'automatic', or default order or execution of processes. Therefore, the execution order determined has to be actively established by either the subcomponents that represent the subprocesses themselves, or by other components. The semantic structure developed in this thesis is not committed to either choice. However, it is not possible to leave the choice implicit.

, In contrast, it is possible to leave implicit which part of a system is responsible for exercising control in many other modelling frameworks. In those frameworks, the execution order determined during analysis and/or design is specified using a notation such as pseudo-code (as in CommonKADS, see Section 8.1.3), some form of dynamic logic, or temporal logic. The semantics of such a notation specifies how this notation is to be interpreted. If the semantics is specified in an operational style (e.g., Plotkin, 1981,1982), the interpretation is given in the form of a mechanism that is able to exercise control as specified. However, this mechanism is only given as a means to find the interpretation of the specification of the proper execution order for the subprocesses. Such frameworks do not explicitly assume the existence of an entity that actually controls the subprocesses and which itself is a part of the system. As an example, consider an algorithm specified in an imperative programming language such as Pascal. The designer of the algorithm specifies the proper execution order of the primitive Pascal statements by means of the control statements provided by the language (e.g., repeat ... until, do ... while). The microprocessor of the computer that executes the program (more or less automatically) carries out the primitive statement in the order specified, and this is well known to all users of the language. However, it is important to note that the component that is actually responsible for the proper execution order is not represented explicitly in the specification of the algorithm. (For a language

designed for sequential computers such as Pascal, it is implicitly assumed that there is only one component that is active at a time, therefore explicit representation of this component is irrelevant.)

In a multi-agent system, there is by definition more than one entity in the system that is active and, at least in principle, able to exercise control over other entities. Nevertheless, it is possible to specify that subprocesses of agents are to be executed in a specific order. In this case, however, confusion may arise as to which component is responsible for exercising the control needed to ensure that subprocesses are executed in the specified order. Usually, a fixed choice is made in the semantics of the notation used for the specification.

For the refinement of the semantic structure developed in this chapter, the situation is slightly different, as there is no specification notation associated with the semantic structure or its refinement. Apart from this, the question of which component is responsible for exercising control is equally relevant. As stated at the beginning of this section, in the refinement of the semantic structure, there is no default choice, and the question has to be answered explicitly.

8.2.2 Dedicated Control Components

The second commitment is the commitment to dedicated control components. This commitment states that, in a compositional system, control is exercised by specifically designated components, called *control components*, that do not have a domain-dependent function. However, apart from their dedication, control components are not special: they are normal components. Control components carry out the process described by Chandrasekaran. In other words, the process of exercising control can be found in these components. A control component can only control components that are subcomponents of the same parent component as the control component itself. As a control component is, apart from its dedication, a normal component, a control component itself may be a composed component. In this case, one of the subcomponents of the control component is itself a control component, which controls the other subcomponents. However, at the level of the parent component, this is not important.

An additional commitment can be distinguished with respect to dedicated control components. This additional commitment is discussed in Section 8.2.2.1.

8.2.2.1 Multiplicity of Control Components

In the semantic structure developed in this thesis, it is assumed that there is exactly one control component in each composed component. As a consequence, all other subcomponents are controlled components, and these controlled components are controlled by the unique control subcomponent of the parent component. This commitment is the most simple option.

An alternative approach is to allow more than one control component within a specific composed component. In this case, the set of controlled components has to

be partitioned in subsets of subcomponents that are controlled by the same control component to avoid conflicting requirements imposed by different control components on the same controlled component. Such a partition could be enforced by the introduction of a new composed component for each partition with all controlled components in the partition and their control component as subcomponent. In this case, the result is that each composed component has exactly one control component.

As each composed component has exactly one control component, all subcomponents that are not control components are controlled components. (If composed components that do not have a control component were allowed, then subcomponents of that component would not be controlled at all: they are able to behave in any conceivable way, without any influence by other components.) As a consequence, all components that are not control components are potentially constrained in their behaviour. However, applications of the semantic structure may choose, for specific composed components, to specify a control component that enforces none, or almost no restrictions on the behaviour of the subcomponents it controls.

8.2.3 Dedicated Control Substates

As explained in Section 2.1.1, the semantic structure distinguishes a state for each component and link. The state of a component consists of three substates: the state of the input interface, the state of the output interface and an internal state. To separate control as described in Section 8.1.1, the substates are partitioned further: each of the three substates is divided in a control part and a domain part. Formally, this can be accomplished by defining the sets $\mathcal{S}_{C,in}$, $\mathcal{S}_{C,int}$ and $\mathcal{S}_{C,out}$ for a component C as unions of sets of control states and domain states. The partition of each substate in a control part and a domain part also holds for the interfaces: each interface is divided in a control part and a domain part. Likewise, the state of a link is divided in a control substate and a domain substate.

As explained in Section 8.1.2, applications of the semantic structure can designate specific information to be control information. The semantic structure uses the following principle for designating information to be control information: by placing information in the control part of an interface, it is designated to be control information. Consequently, control information is the information that determines the control part of each substate. Control is thus separated from domain information in a compositional system and appears in specifically designated control components and in the control part of the substates of all components. The (behaviour of the) control parts is subject to the constraint mentioned in Section 8.1.2: a controlled component behaves as directed. This is made more precise in Section 8.2.6. In Chapter 9, syntactic structures for specifying information as control information are presented. With these structures, control states can be described using domain-independent terms from an ontology for the specification of control (Chandrasekaran, Josephson & Benjamins, 1998).)

Formally, control information is captured as follows:

Definition 8.1. (Control and domain states).

- *Let C be a component with set of states* $\mathcal{S}_C=\langle \mathcal{S}_{in};\mathcal{S}_{int};\mathcal{S}_{out}\rangle$ *such that* $\mathcal{S}_X=\mathcal{S}_{X,c}\times\mathcal{S}_{X,d}$ *for* $X\in\{in,int,out\}$. *Then* $\mathcal{S}_{in,c}$ *is the component's set of* control states *and* $\mathcal{S}_{X,d}$ *is the component's set of* domain states.
- *Let I be a link with set of states* $\mathcal{S}_I$ *such that* $\mathcal{S}_I=\mathcal{S}_{I,c}\times\mathcal{S}_{I,d}$. *Then* $\mathcal{S}_{I,c}$ *is the link's set of* control states *and* $\mathcal{S}_{I,d}$ *is the link's set of* domain states.

8.2.4 Dedicated Control Links

Chandrasekaran characterises control as an attempt (by a control component) to influence another component, based on information on the state of the other component. Thus, control is exercised using information transmission. Similar to control components and control substates, information transmission specific for control is separated from other information transmission. In the semantic structure, control information is transmitted by dedicated control links, which are, apart from their dedication, normal links. There are four kinds of control links. First, there are control links from a control component to a controlled component or link. These control links, called *downward control links*, transmit control information that specifies the initial part of the future of the controlled component or link. Second, there are control links from a controlled component or link to a control component. These control links, called *upward control links*, transmit feedback control information that is used by the control component to evaluate the effect of control it exercised. Third, there are *import control links* which transmit control information from the control part of the input interface of a composed component to the control part of the input interface of the control component in this composed component. Fourth, there are *export control links* which transmit control information from the control part of the output interface of the control component in a composed component to the control part of the output interface of this composed component.

Thus, dedicated control components and dedicated control links are distinguished. As explained in Chapter 5, the structure of a compositional system is described by a structure hierarchy. The designation of dedicated control components and links within a structure hierarchy is formally captured by the definition of a special class of structure hierarchies, called *structure hierarchies with control*. Informally, a structure hierarchy $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ is a structure hierarchy with control if the following requirements hold:

- The set of components *Comp* can be partitioned in a set of control components *Contr* and a set of controlled components *Comp';*
- And the set of links *Lnk* can be partitioned in a set of upward control links *UCLnk*, a set of downward control links *DCLnk*, a set of import control links *ICLnk*, a set of export control links *ECLnk* and a set of controlled links *Lnk';*

- And for each control component, there is a (control or controlled) component of which it is a subcomponent, together with an import control link I_1 from the input interface of the parent component to the control component and an export control link from the control component to the output interface of the parent component;
- And for each controlled component C that is not primitive, there is exactly one control component C' such that C' is a subcomponent of C and for each other subcomponent or link S of C, there is a downward control link I_1 from C' to S and an upward control link I_2 from S to C';
- And for each downward control link I in a composed component C, there is a control component C' and a controlled subcomponent or link S of C such that I is a link from C' to S;
- And for each upward control link I in a composed component C, there is a control component C' and a controlled subcomponent or link S of C such that I is a link from S to C';
- And for each import control link I, there is a control component C and a component P such that C is a subcomponent of P and I is a link from the input interface of P to the input interface of C;
- And for each export control link I, there is a control component C and a component P such that C is a subcomponent of P and I is a link from the output interface of C to the output interface of P;

The eight informal requirements directly correspond with the eight requirements in the following definition:

Definition 8.2. (Structure hierarchy with control). *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy. This structure hierarchy is a* structure hierarchy with control *if:*

- *$Comp=Contr\cup Comp'$ with Contr and Comp' disjoint;*
- *$Lnk=Lnk'\cup UCLnk\cup DCLnk\cup ICLnk\cup ECLnk$ with Lnk', UCLnk, DCLnk, ICLnk and ECLnk pairwise disjoint;*
- *For all $C\in Contr$ there is a component $C'\in Comp$, a link $I_1\in ICLnk$ and a link $I_2\in ECLnk$ such that $C\prec C'$, $I_1\prec C'$, and $I_2\prec C'$, $dom(I_1)=C'$, $cdom(I_1)=C$, $dom(I_2)=C$ and $cdom(I_2)=C'$;*
- *For all $C\in Comp'$ such that $C\notin Prim(SH)$, there is exactly one $C'\in Contr$ such that $C'\prec C$ and for all S in $Comp\cup Lnk$, if $S\prec C$ and $S\neq C'$, then there is an $I_1\in DCLnk$: $dom(I_1)=C'$ and $cdom(I_2)=S$ and there is an $I_2\in UCLnk$: $dom(I_2)=S$ and $cdom(I_2)=C'$;*

- *For all $I \in DCLnk$ such that $I \prec C$ for $C \in Comp$, there is a $C' \in Contr$ and an $S \in Comp \cup Lnk$ such that $dom(I)=C'$ and $cdom(I)=S$ and $S,C' \prec C$;*
- *For all $I \in UCLnk$ such that $I \prec C$ for $C \in Comp$, there is a $C' \in Contr$ and an $S \in Comp \cup Lnk$ such that $dom(I)=S$ and $cdom(I)=C'$ and $S,C' \prec C$;*
- *For all $I \in ICLnk$, there is a component $C \in Contr$ and a component $P \in Comp$ such that $C \prec P$, $dom(I)=P$ and $cdom(I)=C$;*
- *For all $I \in ECLnk$, there is a component $C \in Contr$ and a component $P \in Comp$ such that $C \prec P$, $dom(I)=C$ and $cdom(I)=P$.*

Example 8.3. In Chapter 5 (Example 5.10), a structure hierarchy $sh=\langle Comp;Lnk;\prec;dom;cdom\rangle$ was presented, with

- $Comp=\{$toplevel,user_1,broker,ASP,OPC$\}$;
- $Lnk=\{$user_1_to_broker,broker_to_user_1$\}$;
- $\prec = \{\langle$ASP;broker$\rangle,\langle$OPC;broker$\rangle,\langle$user_1;toplevel$\rangle,\langle$broker;toplevel$\rangle,$ $\langle$user_1_to_broker;toplevel$\rangle,\langle$broker_to_user_1;toplevel$\rangle\}$;
- $dom = \{\langle$user_1_to_broker;user_1$\rangle,\langle$broker_to_user_1;broker$\rangle\}$;
- $cdom = \{\langle$user_1_to_broker;broker$\rangle,\langle$broker_to_user_1;user_1$\rangle\}$;

This structure hierarchy describes the structure of a part of the running example system. (Only one user agent was included in the example structure hierarchy to keep the example concise.) The structure hierarchy with control for this example is the structure hierarchy $sh'=\langle Comp';Lnk';\prec';dom';cdom'\rangle$, with:

- $Comp'=Comp \cup Contr$, *with* $Contr=\{$toplevel_control,broker_control$\}$;
- $Lnk'=Lnk \cup UCLnk \cup DCLnk \cup ICLnk \cup ECLnk$, *with:*
  - $UCLnk=\{$broker_UCL,user_1_UCL,OPC_UCL,ASP_UCL$\}$;
  - $DCLnk=\{$broker_DCL,user_1_DCL,OPC_DCL,ASP_DCL$\}$;
  - $ICLnk=\{$toplevel_ICL,broker_ICL$\}$;
  - $ECLnk=\{$toplevel_ECL,broker_ECL$\}$;
- $\prec' = \prec \cup \{\langle$broker_control;broker$\rangle,\langle$toplevel_control;toplevel$\rangle,\langle$toplevel_ICL;toplevel$\rangle,$ $\langle$toplevel_ECL;toplevel$\rangle,\langle$broker_ICL;broker$\rangle,\langle$broker_ECL;broker$\rangle,$ $\langle$broker_UCL;toplevel$\rangle,\langle$broker_DCL;toplevel$\rangle,\langle$user_1_UCL;toplevel$\rangle,$ $\langle$user_1_DCL;toplevel$\rangle,\langle$OPC_UCL;broker$\rangle,\langle$OPC_DCL;broker$\rangle,$ $\langle$ASP_UCL;broker$\rangle,\langle$ASP_DCL;broker$\rangle\}$;
- $dom' = dom \cup \{\langle$toplevel_ICL;toplevel$\rangle,\langle$toplevel_ECL;toplevel_control$\rangle,$ $\langle$broker_ICL;broker$\rangle,\langle$broker_ECL;broker_control$\rangle,\langle$broker_UCL;broker$\rangle,$ $\langle$broker_DCL;toplevel_control$\rangle,\langle$user_1_UCL;user_1$\rangle,$ $\langle$user_1_DCL;toplevel_control$\rangle,\langle$OPC_UCL;OPC$\rangle,\langle$OPC_DCL;broker_control$\rangle,$ $\langle$ASP_UCL;ASP$\rangle,\langle$ASP_DCL;broker_control$\rangle\}$

- $cdom' = cdom \cup \{\langle$toplevel_ICL;toplevel_control$\rangle, \langle$toplevel_ECL;toplevel$\rangle, \langle$broker_ICL;broker_control$\rangle, \langle$broker_ECL;broker$\rangle, \langle$broker_UCL;toplevel_control$\rangle, \langle$broker_DCL;broker$\rangle, \langle$user_1_UCL;toplevel_control$\rangle, \langle$user_1_DCL;user_1$\rangle, \langle$OPC_UCL;broker_control$\rangle, \langle$OPC_DCL;OPC$\rangle, \langle$ASP_UCL;broker_control$\rangle, \langle$ASP_DCL;ASP$\rangle\}$.

The requirements for a structure hierarchy with control can easily be checked:

- *Comp'* and *Lnk'* are partitioned in pairwise disjoint sets as required;
- $\langle$broker_control;broker$\rangle \in \prec$, $\langle$toplevel_control;toplevel$\rangle \in \prec$, $\langle$broker_ICL;broker$\rangle \in \prec$, $\langle$broker_ECL;broker$\rangle \in \prec$, $\langle$toplevel_ICL;toplevel$\rangle \in \prec$, and $\langle$toplevel_ECL;toplevel$\rangle \in \prec$, so for each $C \in Contr$, there is a $C' \in Comp$, a $I_1 \in ICLnk$ and a $I_2 \in ECLnk$ such that $C \prec C'$, $I_1 \prec C'$ and $I_2 \prec C'$;
- Components toplevel and broker are the only composed components in *sh'*. For both, there is exactly one component $C' \in Contr$ such that C' is a subcomponent of the composed component;
- For each component or link $S \in Comp \cup Lnk$, there are upward and downward control links connected to S and the control component $C' \in Contr$ such that $C' \prec P$ for $S \prec P$;
- All links in *DCLnk, UCLnk, ICLnk* and *ECLnk* are connected as requested.

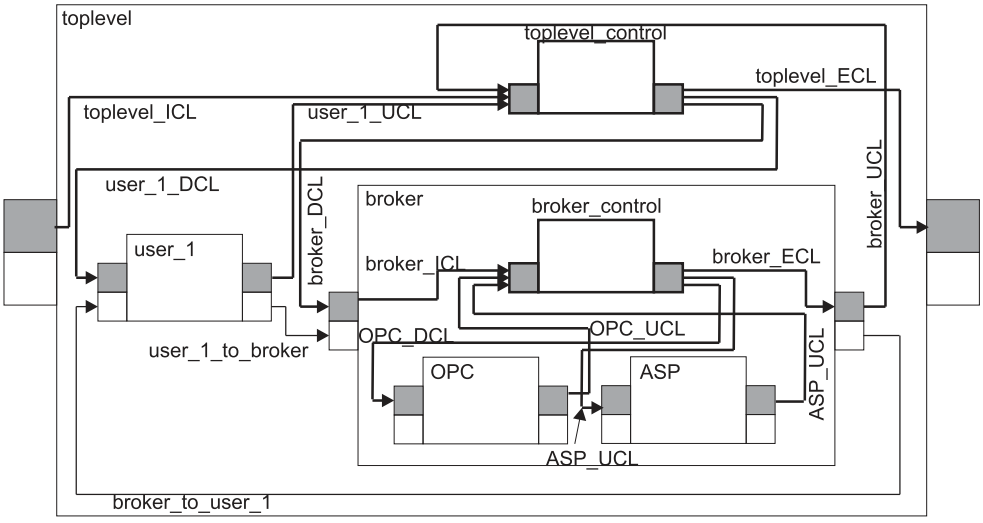

Figure 8.2: Example of a structure hierarchy with control.

The structure hierarchy with control is depicted in Figure 8.2. According to their names, in this example, for the broker agent, there seem to be *two* control components: the dedicated control component broker_control to which all dedicated

control links are connected, and OPC (Own Process Control). However, OPC is not a dedicated control component. Instead, it is a regular component that does not differ from ASP. In the generic agent model presented in Chapter 3, OPC is the component responsible for strategic reasoning, including goal adaptation and priorities. Component OPC executes high-level control which is most often dependent on the domain. The information it processes should therefore *not* be designated as control information. This is illustrated in the example in Section 8.4.■

8.2.5 Channel State and Channel Activation

A control component exercises control over a set of components and links. To exercise control, a control component exchanges information with the controlled components and links using control links. As stated before, control links are, apart from their dedication, normal links. This raises a question: how are control links controlled? The answer consists of a commitment that is specific for control components and links: the state of a control link is fixed and cannot be controlled dynamically. The state of control links is such that they continuously try to transmit control information that appears in the output interface of its source component. This behaviour cannot be influenced by anything in the compositional system. However, if a component wants to delay transmission of control information, it can delay putting this information in its output interface. An advantage of this alternative is that control links are completely independent from the control components to which they are connected.

Another option is to introduce a special kind of link that is used as a control link. This kind of link is special in the sense that it can be activated directly by the control component to which it is connected. This is unlike normal links, which cannot be controlled by the components to which they are connected. (As stated above, normal links are controlled via control links by a control component, similar to components.)

8.2.6 Constraints Imposed by Control

As Chandrasekaran's (1994) abstract characterisation of control indicated, control restricts the future of controlled components and links. The semantic structure therefore requires that a relation between a control component and the components and links it controls is defined that describes in what respect control restricts the future of controlled components and links. However, for reasons explained in Section 8.5, the semantic structure is not committed to a specific relation. Instead, in Section 8.5, a general notion is presented that enables such relations to be defined. Section 8.5 also presents a requirement, the co-ordinated control requirement, that is advised for application of the semantic structure.

8.3 Control in a Multi-Agent System

Section 8.2 presented additional commitments to properties of constructs and relations between constructs provided by the semantic structure that support separate, domain-independent control in a compositional system. The properties to which commitment has been made are encountered in applications of the semantic structure. This section discusses how these properties and relations can be used in the primary application considered in this thesis: modelling multi-agent systems as compositional systems. In Section 8.3.1, the relation between control and autonomy is discussed. Section 8.3.2 presents some perspectives on control in multi-agents system. In Section 8.3.3, modelling choices with respect to control are investigated. In Section 8.3.3, the use of the commitments presented in Section 8.2 for modelling multi-agent systems is discussed.

8.3.1 Control and Autonomy

A multi-agent system consists of a number of agents, each of which is assumed to be autonomous. Even without a philosophically completely satisfactory definition of autonomy, it seems clear that control by one agent over another might conflict with the autonomy of the agents. The central idea put forward in (Castelfranchi, 1995) and (Luck & d'Inverno, 1995) is that to be autonomous, an agent must be able to generate (from its motivations) its own goals, so the agent is not only dependent on the goals of others. This notion of autonomy is called 'goal autonomy' in (Castelfranchi, 1995), who also distinguishes a less strict form of autonomy, called 'executive autonomy'. (For executive autonomy, an agent should be able to accomplish its goals (either generated by itself or given to it by others) free of direct influence of the environment, free to choose among the means needed for the goal.)

Thus, it is impossible for an agent to directly and forcefully change the goal of another agent. However, this does not imply that it is impossible to control an autonomous agent. To control an autonomous agent, instead of forcefully resetting its goals or beliefs (against which it can only complain unsuccessfully), the agent is influenced in such a way that it adopts new goals by itself. This influence can be exercised by communication or by changing the environment of the controlled agent such that this agent changes its beliefs and goals based on new observations. Thus, control over an *autonomous* agent amounts to attempting to influence it such that it changes its state, specifically its goals. This is completely conform the characterisation of control presented at the beginning of Section 8.1. There is no conflict between control as characterised in this chapter and an agent's autonomy. This is in correspondence with the observation that responses of many living systems are "neither caused by, nor independent of the external stimuli." (Chomsky, 1988). There must be no external influences that directly dictate an agent's responses. Under these circumstances, an agent is able to accomplish a goal (either given to it or generated by itself) by the means it chooses.

8.3.2 Some Perspectives on Control in Multi-Agent Systems

Control in a multi-agent system can be related to some other areas:

- In most multi-agent systems, the activities of individual agents in the shared environment introduce interdependencies between the agents and between individual agents and object in the environment. In general, management of these interdependencies by the agents is essential for the agents to achieve their goals. This management of interdependencies, or *co-ordination*, is thus essential in multi-agent systems. Malone and Crowston (1994) study co-ordination from an interdisciplinary point of view, surveying and comparing methods for co-ordination in Computer Science, in biological systems and in Management Science. Malone and Crowston define co-ordination as "management of interdependencies" and argue that co-ordination is a more general notion than co-operation, collaboration and competition. (I.e., co-operation, collaboration and competition are special forms of co-ordination. Co-operation usually is connoted with a shared goal that cannot be achieved individually. Collaboration implies a division of work, and competition usually has the connotation that interdependencies have to be managed in the presence of other agents with conflicting goals.) Control in multi-agent systems as sketched in Section 8.3.1 borders on co-ordination.

- Agents in a multi-agent system can be compared with concurrent objects or actors. Wooldridge (1999) and Briot and Gasser (1998) characterise agents as a special kind of concurrent objects that "decide for themselves whether or not to perform an action on request from another agent" (Wooldridge, 1999, p. 35). This view coincides with the view on autonomy presented in Section 8.3.1. Concurrent, or active, objects, however, do not possess such autonomy. A (concurrent) object makes a set of state operators (methods) available for other objects to invoke. Once a method is made available, the object has no control over invocations of this method: if another object invokes the method, it has to execute that method.

- Contrary to a concurrent object, an agent can decide whether or not to perform an action on request. Consequently, an agent requires a decision procedure that it can use upon receipt of a request to perform an action. In the generic agent model GAM presented in Section 3.2.2, such a decision procedure can be part of the component Own Process Control. A specific decision procedure that has drawn much attention is the BDI (Beliefs, Desires and Intention) architecture developed by Rao and Georgeff (1991). Receipt of a request to perform an action results in a new belief state of the agent. Based on its belief and desires and a theory on the relationships between beliefs, desires and intentions, an agent may adopt an intention to perform the action. From a control point of view, the BDI-approach can be

compared with logic programming languages: an agent is equipped with application-specific, static knowledge that drives a general control mechanism specific for the BDI approach. This control mechanism plays the same role as an interpreter for a logic programming language. However, the control mechanism in the BDI approach handles knowledge represented in terms of beliefs, desires and intentions. The control mechanism in logic programming usually handles Horn clauses.

8.3.3 Modelling Choices for Control in Multi-Agent Systems

In addition to modelling choices for interaction presented in Chapter 3, one modelling choice is made for control. Options for this modelling choice are discussed in the next subsection.

8.3.3.1 Modelling Control

The primary application of the semantic structure developed in this thesis is to model multi-agent systems. Chapter 3 presented a guideline for modelling a multi-agent system as a compositional system, which can be represented by the semantic structure. The guideline assumes that a multi-agent system is viewed as a collection of processes, and that specific relations between processes are identified.

In almost every multi-agent system, some processes have to control other processes in the sense of Section 8.1: they have to influence other processes to achieve one of their own goals. As explained in Section 8.1, the essence of exercising control is information transmission. Thus, the control relationship between processes identified in the multi-agent system is modelled by information transmission in the compositional system that represents the multi-agent system. Modelling choices with respect to information processing are presented in Chapter 3.

To introduce the modelling choice presented in this section, the characterisation of control presented in this chapter is summarised. As stated in Section 8.1, control information should be separated, and as domain-independent as possible. Moreover, information transmission for control is special in the sense that it constrains the future of controlled components. As a consequence, only processes that process domain-independent control information should be modelled as control components, and only information exchange between processes in the multi-agent system that constrain the future of other processes in an inescapable way should be modelled as transmissions for control. Thus, for each control relationship between processes in a multi-agent system, it is determined whether the information used for control is domain-independent and whether the controlled processes cannot escape the future envisioned by the controlling process. Information for which this is the case is designated as control information, and the controlling process is modelled as a control component. Together with the three relations between processes presented in Chapter 3, a structure hierarchy is

found that describes the compositional system that represents the multi-agent system. If, during analysis of the multi-agent system, it is decided that specific processes process domain-independent control information and that they determine an inescapable future for the processes they control, the structure hierarchy that represents the multi-agent system should be a structure hierarchy with control as defined in Definition 8.1.

As stated in Chapter 1, the semantic structure developed in this thesis can be used to provide semantics for multi-agent modelling frameworks or specification languages. In such frameworks, the problem of modelling control in multi-agent systems is approached in a generic way. For instance, in the DESIRE modelling framework presented in Chapter 9, the framework provides a fixed control lexicon and fixed control components for each multi-agent system modelled within the framework. The precise behaviour of control components, however, is not fixed. The user of the DESIRE framework may represent control relationships between processes by specifying the behaviour of specific control components. Alternatively, the user may represent control relationships between processes by normal information transmission. The choice between these alternatives depends on the nature of the processes involved.

On the one hand, if the processes involved represent agents, the control representation is best represented as information transmission. As stated in Chapter 1, an agent is considered to be, among others, autonomous. Therefore, there is almost no information that constrains the future of those processes in an inescapable way. (The only exceptions are probably creation and death of agents.) On the other hand, most processes within agents are not autonomous at all: their future can be completely determined by control information. In this case, the best choice is to provide a description of the precise behaviour of a specific control component.

8.4 An Example

In this section, a comprehensive example is presented to illustrate how control in a multi-agent system is represented in a compositional system.

Example 8.4. Chapter 4 presented a scenario in which an information broker agent matches information provided by provider agents with requests received from user agents. In Chapter 4, two internal processes for the provider agents are distinguished: Own Process Control (OPC) and Agent Specific Processes (ASP). However, in Chapter 4, no further information on these processes was provided.

In this chapter, the scenario presented in Chapter 4 is extended to illustrate how control is represented in a separate, domain-independent way as follows. Upon receipt of a request from one of the user agent, the broker agent first determines whether it is willing to process the request. (This decision is for instance made on the basis of a payment scheme: only if the user's balance maintained by the broker is positive, the broker processes the request.) In the generic model presented in

Chapter 3, knowledge of whether to process incoming information is often modelled as a subprocess of Own Process Control. The actual processing of the request is considered a subprocess of Agent Specific Processes. (Many different choices are possible. The decision whether to process an incoming request can for instance also be considered a subprocess of Cooperation Management.)

The execution order of Own Process Control and Agent Specific Tasks sketched above is determined during design of the example system. As stated in Section 8.2.1, the question which process is responsible for exercising the control needed to ensure the proper execution order of Own Process Control and Agent Specific Tasks has to be answered. There are two possible alternatives: either OPC and ASP themselves are responsible, or the component of which they are subcomponents (in the example, this is the component called broker, which represents the broker agent). In this example, the second alternative is chosen. Thus, broker controls its subcomponents to ensure the proper execution order upon receipt of a request from one of the user agents.

As indicated by the description above, control within the broker can be expressed independently from the domain (information brokering): it is sufficient to use terms such as execution of processes and activation of components. Moreover, it is realistic to expect that the subprocesses of the broker agent are not autonomous at all and will comply exactly with control exercised by the broker agent. Consequently, the information needed to control ASP and OPC is designated to be control information. In conformance with the commitment presented in Section 8.2.2, a dedicated control component is added as a subcomponent of broker. Moreover, dedicated control links are added to broker. To describe the composition structure of broker including the added control component and links, a substructure of the structure hierarchy with control presented in Example 8.3 can be used:

$SH_{\text{broker}} = SS(\text{broker}, sh') = \langle Comp; Lnk; \prec; dom; cdom \rangle$, *with:*

- $Comp$ = {broker,OPC,ASP,broker_control};
- Lnk = {OPC_UCL,ASP_UCL,OPC_DCL,ASP_DCL,broker_ICL,broker_ECL};
- $\prec$ = $\{\langle \text{OPC};\text{broker}\rangle, \langle \text{ASP};\text{broker}\rangle, \langle \text{broker\_control};\text{broker}\rangle\} \cup \{\langle l;\text{broker}\rangle \mid l \in Lnk\}$;
- dom = $\{\langle \text{broker\_ICL};\text{broker}\rangle, \langle \text{broker\_ECL};\text{broker\_control}\rangle, \langle \text{OPC\_UCL};\text{OPC}\rangle, \langle \text{OPC\_DCL};\text{broker\_control}\rangle, \langle \text{ASP\_UCL};\text{ASP}\rangle, \langle \text{ASP\_DCL};\text{broker\_control}\rangle\}$;
- $cdom$ = $\{\langle \text{broker\_ICL};\text{broker\_control}\rangle, \langle \text{broker\_ECL};\text{broker}\rangle, \langle \text{OPC\_UCL};\text{broker\_control}\rangle, \langle \text{OPC\_DCL};\text{OPC}\rangle, \langle \text{ASP\_UCL};\text{broker\_control}\rangle, \langle \text{ASP\_DCL};\text{ASP}\rangle\}$.

This structure hierarchy is itself a structure hierarchy with control.

In conformance with the commitment presented in Section 8.2.3, control states are distinghuished as a part of the state of *each* component. For controlled components, the control state is part of the input and output substates. The

following sets of control state names are used for broker, OPC and ASP (broker is a controlled component of toplevel, as indicated in Example 8.3):

- $S_{in,csub}$ = {own_state(active),own_state(idle)};
- $S_{out,csub}$ = {own_evaluation(succeeded),own_evaluation(failed)}.

The domain part of the input and output substates of OPC and ASP are not specified in this example. The domain part of the input and output substates of broker is given in Example 5.2.

The state of the control component broker_control only consists of control information. Its set of component states is therefore fully specified as follows:

- $S_{\text{broker\_control},out}$ = {component_state(OPC,active),
  component_state(ASP,active),
  own_state(idle)};
- $S_{\text{broker\_control},in}$ = {own_state(active),
  evaluation(OPC,succeeded),evaluation(OPC,failed),
  evaluation(ASP,succeeded),evaluation(ASP,failed)};

Note that the elements of these sets are names of states that do not have an internal structure. Similar to controlled components, the state of a control component is determined by its information contents and changes as a result of information processing. This is illustrated by the set of local component traces Beh_{loc}(broker_control) for the control component:

- $lt_{\text{broker\_control},1}$ = own_state(active)|∅|∅ →
  own_state(active)|∅|component_state(OPC,active) →
  evaluation(OPC,succeeded)|∅|∅ →
  own_state(active)|∅|component_state(ASP,active) →
  evaluation(ASP,succeeded)|∅|∅ →
  ∅|∅|own_state(idle);
- $lt_{\text{broker\_control},2}$ = own_state(active)|∅|∅ →
  own_state(active)|∅|component_state(OPC,active) →
  evaluation(OPC,failed)|∅|∅ →
  ∅|∅|own_state(idle).

The two local component traces in Beh_{loc}(broker_control) represent two different situations. The first state of both traces results from information transmission via broker_ICL, which imports control information from the input interface of broker. (It is assumed that receipt of a request from a user agent triggered the occurrence of an input state with own_state(active) as the control part for the input interface of broker.) The output substate of the second state of both traces indicates the envisioned future for OPC: it has to become active. The input substate of the third state of $lt_{\text{broker\_control},1}$ indicates that feedback from OPC is received. As explained below,

evaluation(OPC,succeeded) indicates that OPC has succeeded in establishing that the balance of the user is positive. Consequently, the request received has to be processed by ASP, and therefore, the envisioned future of ASP is that it becomes active, as indicated by the fourth state in $lt_{broker\_control,1}$. Assuming that there are no other requests to be processed, following receipt of feedback information from ASP, the state of broker_control becomes idle. In the second trace, the user's balance is not positive. The input substate of the third state of $lt_{broker\_control,2}$ also indicates that feedback from OPC has been received. However, evaluation(OPC,failed) indicates that OPC has not succeeded in establishing that the balance of the user is positive (possibly because the balance is negative or zero, or OPC could not determine the state of the account for another reason). Consequently, the request received is not processed by ASP. Assuming that there are no other requests to be processed, the state of broker_control becomes idle. In both situations, the information that broker_control becomes idle is transmitted to the output interface of broker by the export control link broker_ECL. This information may then serve as feedback information for a component that controls broker.

The control component broker_control receives control information from the components it controls and transmits control info to these controlled components via dedicated control links. Apart from their dedication, control links are normal links. Information link mappings as defined in Definition 5.6 are used to specify which states are linked. According to Definition 5.6, an information link mapping is an octet of states. To keep this example concise, the state of the links is not specified. Consequently, the third, fourth, fifth and sixth states of octets in an information link mapping, which are states of the link itself, need not be taken into account. Moreover, it is assumed that. the result of information transmission is not recorded by the domain of the link, and no enabling condition for the co-domain is taken into account. Therefore, the second and seventh state in an information link mapping are not necessary. Thus, the information link mappings are presented as sets of pairs of states: the first state is a state of the domain of the link, and the second state is a state of the co-domain. Such pairs specify that if the first state occurs in a local trace of the domain of the link, then the second state has to occur in a trace of the co-domain of the link. The information link mappings used are:

- $\lambda_{broker_ICL} = \lambda_{broker_ECL} = \{\langle$own_state(active);own_state(active)$\rangle$, $\langle$own_state(idle);own_state(idle)$\rangle\}$;
- $\lambda_{OPC_DCL} = \{\langle$component_state(OPC,active);own_state(active)$\rangle$, $\langle$component_state(ASP,active);own_state(idle)$\rangle\}$;
- $\lambda_{OPC_UCL} = \{\langle$own_evaluation(succeeded);evaluation(OPC,succeeded)$\rangle$, $\langle$own_evaluation(failed);evaluation(OPC,failed)$\rangle\}$;
- $\lambda_{ASP_DCL} = \{\langle$component_state(ASP,active);own_state(active)$\rangle$, $\langle$component_state(OPC,active);own_state(idle)$\rangle\}$;

- λ_{ASP_UCL} = {⟨own_evaluation(succeeded);evaluation(ASP,succeeded)⟩, ⟨own_evaluation(failed);evaluation(ASP,failed)⟩};

The envisioned future of OPC and ASP as determined by broker_control is transmitted to the respective components, where the control substates are affected by the receipt of this information. The commitment presented in Section 8.2.6 requires that a relation between the envisioned future as determined by broker_control and the behaviour of the controlled components are specified. This relation is indicated by the set of local component traces Beh_{loc}(OPC) for OPC. In these traces, only the control part of the input and output substates is shown.

- $lt_{OPC,1}$ = ∅|∅|own_state(idle) →
  own_state(active)|∅|∅ →
  … →
  own_state(active)|user_balance_positive|∅ →
  own_state(idle)|∅|own_evaluation(succeeded);
- $lt_{OPC,2}$ = ∅|∅|own_state(idle) →
  own_state(active)|∅|∅ →
  … →
  own_state(active)|user_balance_not_positive|∅ →
  own_state(idle)|∅|own_evaluation(failed);

The starting state for OPC is a state in which it is idle. The second state in both traces indicates that OPC receives control information from broker_control (via the downward control link OPC_DCL, which is added to the structure hierarchy representing the running example system to form a structure hierarchy with control conform Definition 8.2. The downward control link OPC_DCL is depicted in Figure 8.2, together with other control links). As a result, OPC processes the request (indicated by the dots). After a while, the assessment of the user's balance is finished. In the first trace, the balance is positive, which results in the occurrence in the output state own_evaluation(succeeded). In the second trace, the balance is not positive, which results in the occurrence in the output state own_evaluation(failed). Both evaluations are transmitted to broker_control via the upward control link OPC_UCL. The fact that in both traces in Beh_{loc}(OPC), the occurrence of own_state(active) (caused by receipt of information via the downward control link OPC_DCL) is followed by activity indicates that the informtion transmitted via the downward control link fully determines the future of OPC. Moreover, the traces also show how domain dependent information (represented by the internal substates user_balance_positive and user_balance_not_positive) is related to domain-independent control information (own_evaluation(succeeded) and own_evaluation(failed)).

In this example, for ASP the following local component traces are of interest:

- $lt_{ASP,1}$ = ∅|∅|own_state(idle) →
  own_state(active)|∅|∅;

- $lt_{ASP,2}$ = ∅|∅|own_state(idle).

The starting state for ASP is a state in which it is idle. The second state in the first trace indicates that ASP receives control information from broker_control (via the downward control link ASP_DCL). After receipt of this information, ASP starts processing the request. The second trace indicates the situation in which ASP stays idle.

The behaviour of the broker agent can be analysed using the white box view on the behaviour of broker with respect to SH_{broker}. (As all subcomponents of broker are primitive, the white box view equals the glass box view.) The white box view on the behaviour of broker with respect to SH_{broker} is a set of compatible multitraces for SH_{broker}. A multitrace that represents the situation in which the user's balance is positive is as follows:

$$mt_{broker,1} = \{\langle broker; lt_{broker,1}\rangle, \langle broker\_control; lt_{broker\_control,1}\rangle, \langle OPC; lt_{OPC,1}\rangle, \langle ASP; lt_{ASP,1}\rangle, \langle broker\_ICL;\ldots\rangle, \langle broker\_ECL;\ldots\rangle, \langle OPC\_UCL;\ldots\rangle, \langle OPC\_DCL;\ldots\rangle, \langle ASP\_UCL;\ldots\rangle, \langle ASP\_DCL;\ldots\rangle\}.$$

It is straightforward to check that this multitrace is compatible. A multitrace that represents the situation in which the user's balance is positive is as follows:

$$mt_{broker,2} = \{\langle broker; lt_{broker,1}\rangle, \langle broker\_control; lt_{broker\_control,2}\rangle, \langle OPC; lt_{OPC,2}\rangle, \langle ASP; lt_{ASP,2}\rangle, \langle broker\_ICL;\ldots\rangle, \langle broker\_ECL;\ldots\rangle, \langle OPC\_UCL;\ldots\rangle, \langle OPC\_DCL;\ldots\rangle, \langle ASP\_UCL;\ldots\rangle, \langle ASP\_DCL;\ldots\rangle\}.$$

In fact, there are only two pairwise compatible combinations of local component traces for the subcomponents of broker: (1) $lt_{broker\_control,1}$, $lt_{OPC,1}$ and $lt_{ASP,1}$, and (2) $lt_{broker\_control,2}$, $lt_{OPC,2}$ and $lt_{ASP,2}$,. The traces $lt_{broker\_control,1}$ and $lt_{OPC,2}$ are not compatible: in $lt_{OPC,1}$ a state own_evaluation(succeeded) occurs, but in $lt_{broker\_control,1}$, a state evaluation(OPC,succeeded) does not occur, which violates λ_{OPC_UCL}.

8.5 The Relation between Control Components and Controlled Components

Section 8.1 states that information transmission for control is special in the sense that it constrains the future of the component to which control information is transmitted: this component cannot escape the future specified by the control information. Section 8.5.1 explains which commitment of the semantic structure supports this property of information transmission for control. Section 8.5.2 presents some relations between control components and controlled components associated with these commitments.

8.5.1 Constraints Imposed by Control Information

The commitment of the semantic structure with respect to information transmission for control is that the model of the behaviour of a controlled component maintained by a control component must match the actual behaviour of the controlled component. As described in Section 8.1, according to Chandrasekaran (1994), each control processes can be described as a process in which a model of the past and current behaviour of the controlled component is made, after which this model is extended to include an envisioned future for the controlled component. (As the semantic structure abstracts from the structure of information processed in components, the model maintained by a control component cannot be specified. However, similar to all information maintained by a component, such a model determines a part of the state of the control component. Note that the model maintained by a control component is *not* a construct provided by the semantic structure.)

Although there is a commitment to the existence of a relation between the model maintained by a control component and the controlled components, no commitment to a specific relation is made, for the following reasons:

- The precise definition of such a relation depends on how a control component models its controlled component. However, this differs for different applications of the semantic structure. Thus, as the semantic structure abstracts from the structure of the information maintained within a component, a precise definition fixed for the semantic structure of such a relation cannot be given;
- The extent to which a control component is able to determine the future of a controlled component varies from application to application. A commitment to a specific commitment would probably be too restrictive for a number of applications, and not restrictive enough for other applications. If the commitment is too restrictive for an application, almost no information transmission can be designated as control information. If the commitment is not restrictive enough, almost all information transmission can be designated as control information. In both cases, control cannot be separated to such an extent that it leads to a better, reusable structure of the model.
- The precise definition of such a relation determines, to a large extent, a number of other dynamic properties of a compositional system. Applications of the semantic structure differ with respect to these properties. Consequently, if one specific relation is defined for the semantic structure, this relation is most likely too weak for some applications, and too restrictive for others. The way in which such relations affect dynamic properties is discussed in Section 8.5.1. For the application of the semantic structure to DESIRE presented in Chapter 9, a precise definition is given.

Thus, in this chapter, no commitment to a specific relation is made. However, the problem of how to formalise such a relation is addressed in general.

As stated earlier in this chapter, the essence of control is information transmission, and therefore, the constructs presented in Chapter 5 and Chapter 6 suffice for control in the semantic structure. In fact, the constructs presented in Chapter 5 and Chapter 6 suffice to express the relation between a control component and the component it controls in a basic form, using the sets $Beh_{loc}(S)$ for controlled components and links S and information link mappings as follows.

As stated in Section 8.2, the state of controlled components and links consist of control and domain parts. Information in the control parts is exchanged via control links with the control component that controls the controlled components. The state of control components only have a control part. Information link mappings can be used to specify which control information produced by the control component is to be transmitted to which controlled component or link. This control information is meant to influence the behaviour of the controlled component. For example, assume that a state called component_state(A,active) of the control component is linked to a state with a control part called own_state(active) of a controlled component A. Compatibility ensures that if the state component_state(A,active) occurs in a local component trace of the control component, then a state with a control part own_state(active) occurs in the local component trace for A in a compatible multitrace. As the names of these states suggest, A should become active, which, in this example means that the next state of A contains specific domain information, e.g., the next state should be the state called results_available.

Control information is special in the sense that applications of the semantic framework have to ensure that the future of a controlled component as envisioned by a control component is inescapable. In the example sketched in the previous paragraph, this can be accomplished by restricting the set $Beh_{loc}(A)$ to traces in which each state with control part own_state(active) is followed by state results_available. This requirement is a local requirement: it only restricts the set of local component traces of A. (Example 8.4 in Section 8.4 specified the relation between the envisioned future of OPC as determined by broker_control and the local component traces of OPC in a similar way.)

Thus, the relation between a control component and the components it controls can be expressed using the constructs presented in Chapter 5 and Chapter 6: specific states of the control component are linked to specific states of the controlled components and links, and requirements on the sets of local component traces of the controlled components and links further determine how the behaviour of the controlled components is related to behaviour of the control component. However, although in this way, the future of a controlled component is related to the behaviour of the control component, the key point is that it is not clear when this future begins from the point of view of the control component. When the future begins depends on the properties of information transmission (if information transmission is not lossless, it may not begin at all). For instance, in

Example 8.4, it is not clear how the states of broker_control relate to the states of OPC and ASP in time. In local component trace $lt_{broker\_control,1}$, the state in which feedback information from OPC is received (third state) immediately follows the state in which OPC is made active (second state). However, as trace $lt_{OPC,1}$ indicates, there are several local states of OPC between the receipt of the information that OPC has to become active and the state own_evaluation(succeeded).

For some applications of the semantic structure, more detail than provided by the constructs presented in Chapter 5 and Chapter 6 is not needed. However, for many applications more detail is needed for the following reason. Evaluation of the design of a multi-agent system includes analysis of dynamic properties of the system. In general, the three views on the behaviour of a compositional system can be used to assess dynamic properties by comparing occurrences of state transitions in the individual traces within compatible multitraces. For sets of components and links that consist of a dedicated control component and the components and links it controls, a second alternative is available. As stated in Section 8.2.6, the control component determines the behaviour of the components and links it controls, to a large extent. Therefore, it is possible to assess properties of the controlled components and links by evaluating the behaviour of the control component.

As global time is not assumed to be available, it is not possible to directly express relations between time points in different traces using only the constructs presented in Chapter 5 and Chapter 6. However, with the help of global states as defined in Chapter 7, the relation between a control component and the components and links it controls can be specified more precisely.

As an example of the use of global states, it is possible to request that the envisioned future of controlled components and links in a control component occurs in the same global states as the actual future of the controlled components and links. The envisioned future of controlled components and links is represented by specific states of the control components. These states are linked to states of the controlled components and links by downward task control links, as specified by the information link mappings for these links. The suggested requirement can thus be stated more precisely as follows: given a structure hierarchy with control and a compatible multitrace for this structure hierarchy, for each global state σ of the multitrace, if the output part of a control component C in the global state occurs as the first state in a tuple in the information link mapping of a downward control link from C, then in the same global state, the input part for the co-domain of this link must be the eighth state in that tuple. Formally: let $SH=\langle Comp \cup Contr; Lnk \cup UCLnk \cup DCLnk; \prec; dom; cdom \rangle$ be a structure hierarchy with control, let γ be a collection of compatibility relations and let μ be a multitrace compatible for γ. Then:

$\forall \sigma \in SGS(SH,\mu,\gamma),\ \forall C \in Contr,\ \forall I \in DCLnk$ *and* $\forall v \in \mathcal{S}_C$:
if $dom(I)=C$ *and* $\sigma(C)=v$ *and*

$\langle\langle out(v);out(v_{C,j})\rangle;\langle v_{I,i''};v_{I,j''};v_{I,k};v_{I,l}\rangle;\langle in(v_{cdom(I),i'});in(v_{cdom(I),j'})\rangle\rangle \in \lambda_I$,
then $\sigma(cdom(I)) = v_{cdom(I),j'}$ *for any* j,i'',j'',k,l,i' *and* j'.

This requirement, however, is too strict: there are no global states for which this requirement holds. (A global state for which this requirement holds cannot be a global state for the following reason. On the one hand, for each global state σ and for each component C and link I, if $\langle\langle out(v);out(v_{C,j})\rangle;\langle v_{I,i''};v_{I,j''};v_{I,k};v_{I,l}\rangle;\langle in(v_{cdom(I),i'});in(v_{cdom(I),j'})\rangle\rangle \in \lambda_I$, with $\sigma(C) = v$ then $\sigma(C) \rightarrow_{sd} \sigma(cdom(I))$ by the second clause of the definition of strict dependence. On the other hand, by the definition of a global state σ, for each pair of components C and D, $\sigma(C) \rightarrow_{sd} \sigma(D)$, and therefore, for each global state σ and for each component C and link I, $\sigma(C) \rightarrow_{sd} \sigma(cdom(I))$.)

As an alternative, the future of controlled component A may be required to begin as soon as possible. More precisely: each global state in which the local state of the control component is component_state(A,active) must be followed by a global state in which the local state of A is results_available. Formally: let $SH = \langle Comp \cup Contr; Lnk \cup UCLnk \cup DCLnk; \prec; dom; cdom \rangle$ be a structure hierarchy with control, let γ be a collection of compatibility relations and let μ be a multitrace compatible for γ. Then:

$\forall \sigma \in SGS(SH,\mu,\gamma),\ \forall C \in Contr,\ \forall I \in DCLnk$ *and* $\forall v \in \mathcal{S}_C$:
if $dom(I) = C$ *and* $\sigma(C) = v$ *and*
$\langle\langle out(v);out(v_{C,j})\rangle;\langle v_{I,i''};v_{I,j''};v_{I,k};v_{I,l}\rangle;\langle in(v_{cdom(I),i'});in(v_{cdom(I),j'})\rangle\rangle \in \lambda_I$,
then there is a strict global state $\sigma' \in next_{SGS(SH,\gamma,\mu)}(\sigma)$ *such that*
$\sigma'(cdom(I)) = v_{cdom(I),j'}$ *for any* j,i'',j'',k,l,i' *and* j'.

Contrary to the requirement presented previously, this requirement can be met. However, two issues remain. First, this requirement is not very strong, as it does not constrain the global states between the global state with substate component_state(A,active) for the control component the global state with substate results_available for the controlled component. The second issue is related to the fact that the order of global states is a partial order. This issue is discussed with the help of some additional notions, which are presented in the rest of this chapter. Chapter 9 presents an application of the semantic structure in which the principles presented in this chapter are applied.

8.5.2 Common Global States

Chapter 7 defined a notion of global state relative to a multitrace for a structure hierarchy *SH*. As explained in Chapter 7, global states are partially ordered, even if all local component and link traces in a multitrace are linear. The partiality of the global state order represents the fact that different observers may observe different behaviour of the compositional multitrace represented by *SH*, due to the absence of

global time or fixed duration of information transmission. Each observation corresponds with a different path through the partial order of states.

Most global states for a specific multitrace are only observed by a subset of all possible observers. However, some global states occur on each path in the partial order. In other words, such a state is observed by each possible observer. To formally define common global states, first the notion of an observation is formally defined as a linear extension of the partial order of global states.

Definition 8.5. (Observation). *Let SH be a structure hierarchy, let γ be a collection of compatibility relations and let μ be a multitrace compatible for γ. An* observation *of μ is a sequence of strict global states $\sigma_1,\sigma_2,\ldots \in SGS(SH,\mu,\gamma)$ such that:*

- *if $\sigma_i \in next_{SGS(SH,\mu,\gamma,)}(\sigma_j)$, then $i>j$, and*
- *σ_{i+1} is an immediate successor of σ_i., i.e, there is no $\sigma' \in SGS(SH,\mu,\gamma)$ such that $\sigma' \in next_{SGS(SH,\mu,\gamma,)}(\sigma_i)$ and $\sigma_{i+1} \in next_{SGS(SH,\mu,\gamma,)}(\sigma')$.*

The set of all observations of μ is denoted $OBS(SH,\mu,\gamma)$.

The definition of a common global state is now straightforward:

Definition 8.6. (Common global state). *Let SH be a structure hierarchy, let γ be a collection of compatibility relations and let μ be a multitrace compatible for c. A* common global state *is a strict global state $\sigma \in SGS(SH,\mu,\gamma)$ such that for all sequences $(\sigma_i)_{i \in N}$ in $OBS(SH,\mu,\gamma)$, there is an i such that $\sigma=\sigma_i$. The set of all common global states for a structure hierarchy SH, a multitrace μ, and a collection of compatibility relations γ is denoted $CSGS(SH,\mu,\gamma)$.*

In (Fromentin & Raynal, 1995), a necessary and sufficient criterion for global states to be common is presented. In terms of the semantic structure developed in this thesis, this criterion is as follows:

Proposition 8.7. *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy, let γ be a collection of compatibility relations and let μ be a multitrace compatible for γ. A global state $\sigma \in SGS(SH,\mu,\gamma)$ is a common global state iff:*

$$\forall S_1,S_2 \in Comp \cup Lnk: prev_{\mu(S2)}(\sigma(S_1)) \rightarrow_{sd} next_{\mu(S2)}(\sigma(S_2)). \qquad (1)$$

The formal proof of this proposition can be found in (Fromentin & Raynal, 1995). Informally, the proposition is most easily understood for the case where $Comp \cup Lnk$ has precisely two elements, say A and B. Assume that (1) holds, i.e., for a specific strict global state σ, $prev_{\mu(A)}(\sigma(A)) \rightarrow_{sd} next_{\mu(B)}(\sigma(B))$ and $prev_{\mu(B)}(\sigma(B)) \rightarrow_{sd} next_{\mu(A)}(\sigma(A))$ (note that these two dependencies are symmetric). This situation is depicted in Figure 8.3. The configuration of symmetrically crossing dependencies between state transitions (the thick arrows) prevents specific global states from occurring. For instance, global state σ' cannot occur as for σ', the transition from $\sigma(A)$ to $next_{\mu(A)}(\sigma(A))$ occurred, while the transition from

$prev_{\mu(B)}(\sigma(B))$ to $\sigma(B)$ did not occur. This is prohibited because the transition from $prev_{\mu(B)}(\sigma(B))$ to $\sigma(B)$ precedes the transition from $\sigma(A)$ to $next_{\mu(A)}(\sigma(A))$. Likewise, σ'' cannot occur. Starting from σ_0, in all sequences of global states, σ has to occur before any global state containing $next_{\mu(A)}(\sigma(A))$ or $next_{\mu(B)}(\sigma(B))$ can occur. Therefore, σ is a common global state.

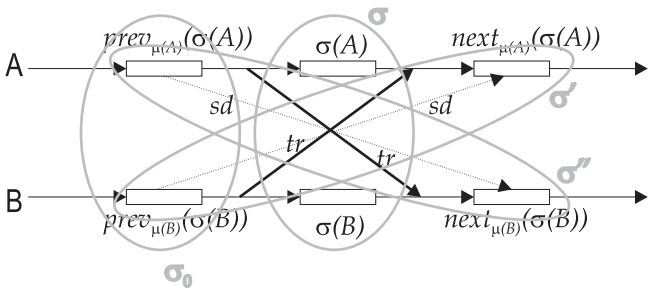

Figure 8.3: Illustration of Proposition 8.7.

Given a partial order of global states $\langle SGS(SH,\mu,\gamma);next_{SGS(SH,\mu,\gamma)}\rangle$, the restriction of this partial order to the subset of common global states is a total order. (If it were not, then some of the common global states would not be in all observations.) The total order of common global states can be used to formulate requirements on the relation between a control component and the components it controls. In the rest of this section, first such a requirement is formally defined. This requirement is then discussed.

8.5.2.1 Co-ordinated Control Requirement

The co-ordinated control requirement expresses a relation between a control component and the components it controls. According to this requirement, control is co-ordinated if control information made available in a specific common global state, is received in the immediate successor common global state of this specific state. (The immediate successor relation is the restriction of the partial order on global states to common global states. As this restriction is a total order, it is possible to require that control information be received in *the* immediate successor common global state.) The co-ordinated control requirement is defined formally as follows:

Definition 8.8. (Co-ordinated Control Requirement). *Let* $SH=\langle Comp \cup Contr; Lnk \cup UCLnk \cup DCLnk; \prec; dom; cdom\rangle$ *be a structure hierarchy with control, let* γ *be a collection of compatibility relations and let* μ *be a multitrace compatible for* γ*. Control in the compositional system represented by SH is* co-ordinated *if:*

- $\forall \sigma \in CSGS(SH,\mu,\gamma)$, $\forall C \in Contr$, $\forall I \in DCLnk$: *if* $dom(I)=C$ *and* $\langle\langle out(\sigma(C));out(v_{C,j})\rangle;\langle v_{I,i''};v_{I,j''};v_{I,k};v_{I,l}\rangle;\langle in(v_{cdom(I),i'});in(v_{cdom(I),j'})\rangle\rangle \in \lambda_I$, *then there is a common global state* $\sigma' \in next_{SGS(SH,\gamma,\mu)}(\sigma)$ *such that:*
  - $\sigma'(cdom(I))=v_{cdom(I),j'}$ *for any* j,i'',j'',k,l,i' *and* j';
  - *there is no* σ'' *such that* $\sigma'' \in next_{SGS(SH,\gamma,\mu)}(\sigma)$ *and* $\sigma' \in next_{SGS(SH,\gamma,\mu)}(\sigma'')$.
- *and* $\forall \sigma \in CSGS(SH,\mu,\gamma)$, $\forall C \in Contr$, $\forall I \in UCLnk$: *if* $cdom(I)=C$ *and* $\langle\langle out(\sigma(dom(I));out(v_{dom(I),j})\rangle;\langle v_{I,i''};v_{I,j''};v_{I,k};v_{I,l}\rangle;\langle in(v_{C,i'});in(v_{C,j'})\rangle\rangle \in \lambda_I$, *then there is a common global state* $\sigma' \in next_{SGS(SH,\gamma,\mu)}(\sigma)$ *such that:*
  - $\sigma'(cdom(I))=v_{C,j'}$ *for any* j,i'',j'',k,l,i' *and* j';
  - *there is no* σ'' *such that* $\sigma'' \in next_{SGS(SH,\gamma,\mu)}(\sigma)$ *and* $\sigma' \in next_{SGS(SH,\gamma,\mu)}(\sigma'')$.

The first part of this requirement resembles the second requirement put forward in the previous subsection. There are two differences. First, the co-ordinated control requirement only formulates a requirement on those global states that are common. (The first universal quantifier ranges over the set of common global states.) Second, the result of control information transmission must be present in the *first* common global state that follows the common global state in which it was available in the control component. The second part of the co-ordinated control requirement co-ordinates feedback of control information from controlled components and links to the control component. To keep the example requirements in Section 8.5.1 concise, co-ordination of feedback information is not included in these requirements.

Common global states and the co-ordinated control requirement thus only involve a subset of all global states associated with a multitrace. It is indeed possible to formally introduce a restriction of the partial order of global states as an abstract perspective on a multitrace, in which only specific global states are represented. This can be appreciated as follows. Suppose that a compositional system is represented by a structure hierarchy with control *SH* and that for multitraces for *SH*, the co-ordinated control requirement must hold. At an early stage in the design, it is sufficient to specify that this requirement must hold. However, at a later stage, mechanisms must be incorporated in the design to ensure that the requirements hold. As an example, exchange of extra information may be needed to ensure that control information is received at the first common global state that follows a common global state in which the control information is made available. Figure 8.4 below depicts (part of) a partial order of eight global states (represented by ovals). Two of the global states (depicted in black) are common global states. The black arrows indicate the total order of these common global states. The small circles inside the global states depict the local component states of a control component. The leftmost common global state is a state in which control is co-ordinated. In the local component state of the control component, new control information is made available. This control information is transmitted to a number of controlled components. This information is not received by all

controlled components at the same time, and moreover, the controlled components possibly exchange information with each other as well. The resulting behaviour is depicted by the grey states in Figure 8.4. The behaviour of the individual components (as represented by sets of local component traces $Beh_{loc}(S)$), however, is designed such that eventually there is a new common global state in which the control information is available to all controlled components. In this new common global state, control is co-ordinated.

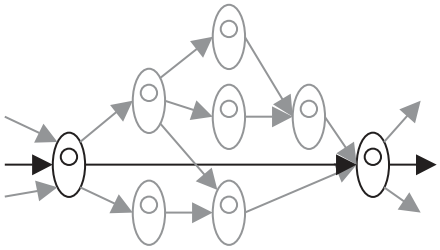

Figure 8.4: Common global states.

8.5.2.2 Some perspectives on the co-ordinated control requirement

Some advantages and disadvantages of the co-ordinated control requirement can be distinguished.

- The set of common global states is totally ordered. If the local component traces used to describe the behaviour of the control component are linear, the 'actual behaviour' of the controlled components as described by the linear order of common global states has the same structure as the behaviour of the control component. Thus, the envisioned future behaviour of controlled components as represented in a control component can be compared to their actual behaviour. (E.g., by viewing the combination of the local component state of a control component and of a controlled component in the same common global state as a bisimulation relation.)
- As stated before, the partial order of global states reflects different observations of the behaviour of a compositional system. Consequently, some properties of the behaviour of a compositional system, such as safety properties, hold for all observations, while other properties do not hold for all observations. However, under specific circumstances, the latter case suffices, as is explained in (Katz & Peled, 1990).
- As Proposition 8.7 indicates, a global state is a common global state iff there is a mutual dependency between local states of each pair of components and links in a compositional system. A mutual dependency between states in essence means that the possible next local state transitions are mutually

constrained. Components in a common global state are thus synchronised: from a global time point of view, they are all in this state at the same time. Consequently, if common global states are required to exist, as is the case for the co-ordinated control requirement, controlled components are synchronised with their control component and with one another. Thus, the co-ordinated control requirement influences to a large extent the possible behaviour of controlled components. In an implementation, common global states are a source of concurrency bottlenecks. A possible solution is to apply the co-ordinated control requirement to a subset of all controlled components in a structure hierarchy. Fromentin and Raynal (1995) offer a similar solution, which they call *partial common global states*.

The paper by Fromentin and Raynal (1995) from which Proposition 8.7 was taken, introduces abstraction of specific states in a different way. The starting point for Fromentin and Raynal is a partial order on a set of events, which is assumed to be given. The set of events is partitioned in 'primitive level' events and 'user level' events. A primitive level event is not a 'sub-event' of a user level event. (This would not be possible, as events are atomic and do not have any internal structure.) The partial order of events is then restricted to the set of user level events. The resulting partial order on (user level) events is then used to define global states in a similar way as presented in Chapter 7 and common global states as defined in Section 8.5.2. The restriction of the partial order on global states to common global states presented in Section 8.5.2.1 is not used by Fromentin and Raynal (1995).

As an aside, in some approaches, events are not atomic. A hierarchical structure of events is used to model levels of event abstraction. Lamport (1986) introduces hierarchies of event abstraction to represent levels of detail of the behaviour of a system. Kshemkalyani (1998) describes all possible precedence relations between events that may have sub-events.

In the semantic structure presented in this thesis, control restricts the future of controlled components and links. The co-ordinated control requirement presented in Section 8.5.2.1 provides a precise expression of the restriction relation between a control component and the components and links it controls. It is advised for applications of the semantic structure to adopt the co-ordinated control requirement. However, applications are free to adopt alternative requirements to express how control restricts the future of controlled components and links.

Chapter 9

Semantics for the DESIRE Multi-Agent Modelling Framework

In this chapter, the semantic structure developed in the previous chapters is applied to the DESIRE multi-agent modelling framework, providing a formal semantics for DESIRE. (See (Brazier, Jonker & Treur, 1998) for an overview of the principles behind DESIRE. A generic agent model modelled in DESIRE is described in (Brazier, Jonker & Treur, 2000). The generic agent model has been applied in many domains including electricity transportation management (Brazier, Dunin-Keplicz, Jennings & Treur, 1997), electricity load balance management (Brazier, Cornelissen, Gustavsson, Jonker, Lindeberg, Polak & Treur, 2000) and as a basis for the co-operative agent model which has been applied, for example, in distributed call centre support (Brazier, Cornelissen, Jonker & Treur, 2000). An earlier version of DESIRE is described in (Langevelde, Philips & Treur, 1992)).

Section 9.1 provides a general overview of the DESIRE modelling framework. In Section 9.2, the main DESIRE constructs are introduced. In Section 9.3, a formal description of the dynamics of DESIRE semantics is given. Section 9.3.2 presents a discussion of some design choices made in the development of the DESIRE modelling framework and their relation with specific commitments of the semantic structure developed in this thesis.

Section 9.1 is a revised version of the introduction of (Brazier, Eck & Treur, 2001a). The design of the DESIRE modelling framework is presented in (Brazier, Treur, Wijngaards & Willems, 1999; Brazier, Jonker & Treur, 1998).

9.1 Compositional Development of Multi-Agent Systems

DESIRE is a modelling framework for the compositional design of multi-agent systems. The modelling framework provides support for the design of multi-agent systems at three levels: conceptual design, detailed design, and (prototype) implementation. The three layers are presented in Figure 9.1. At each layer, a design consists of knowledge of the following three types: process composition,

knowledge composition, and the relation between process composition and knowledge composition. These three types of knowledge are discussed in more detail below.

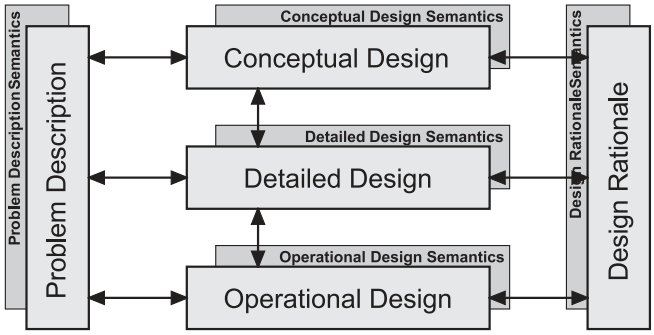

Figure 9.1: The three layers of the DESIRE modelling framework.

The first type of knowledge is knowledge about process composition. In DESIRE, a multi-agent system is viewed to be a collection of processes. Process composition entails the following aspects:

- Process composition identifies the relevant processes at different levels of (process) abstraction, and describes how a process can be defined in terms of (is composed of) lower level processes.
- Processes can be described at different levels of abstraction; for example, the process of the multi-agent system as a whole, processes defined within individual agents and the external world, and processes defined for specific tasks of individual agents. In conformance with the guideline presented in Chapter 3, the identified processes are modelled as *components*. As in Chapter 2, three aspects of a component are represented in DESIRE: state, interfaces and composition structure. The levels of process abstraction identified are modelled as *abstraction/specialisation relations* between components: components may be *composed* of other components or they may be *primitive*.
- The way in which processes at one level of abstraction are composed of processes at the adjacent lower abstraction level is called *composition*. This composition of processes is described by a specification of the possibilities for *information exchange* between processes (*static view* on the composition), and a specification of *task control knowledge* used to control processes and information transmission (*dynamic view* on the composition).

Second, knowledge composition identifies the knowledge structures at different levels of (knowledge) abstraction, and describes how a knowledge structure can be defined in terms of lower level knowledge structures. The knowledge abstraction

levels may correspond to the process abstraction levels, but this is often not the case.

- The two main structures used as building blocks to model knowledge are: *information types* and *knowledge bases*. These knowledge structures can be identified and described at different levels of abstraction. At higher levels details can be hidden. An *information type* defines an ontology (lexicon, vocabulary) to describe objects or terms, their sorts, and the relations or functions that can be defined on these objects. Information types can be represented in order-sorted first-order logic. A *knowledge base* defines a part of the knowledge that is used in one or more of the processes. Knowledge is represented by formulae in order-sorted first-order logic, which can be normalised by a standard transformation into rules.

- Information types can be composed of more specific information types, following the principle of compositionality discussed above. Similarly, knowledge bases can be composed of other knowledge bases. The compositional structure is based on the different levels of knowledge abstraction distinguished, and results in information and knowledge hiding.

Third, each process in a process composition uses knowledge structures. Which knowledge structures are used by which processes is defined by the relation between process composition and knowledge composition.

As stated in Chapter 1, the basic assumption adopted in the development of the semantic structure is that a multi-agent system is represented as a compositional system. Chapter 3 provided a guideline for the design of a compositional system that represents a multi-agent system. The relation between Chapter 3 and this chapter is as follows:

- DESIRE *supports* the guideline put forward in Chapter 3. The modelling framework provides all facilities to *describe* a compositional system that represents a multi-agent system, including the dynamics of such a system. Moreover, the modelling framework provides tools for automatic prototype construction and experimentation.

- DESIRE has a number of additional commitments compared to the commitments put forward in Chapter 2 and Chapter 3. The most important commitment is the commitment to a *knowledge engineering perspective* on the design of multi-agent systems. Chapter 3 abstracts from the way in which individual processes are analysed and/or described, as well as from the way information processed in the multi-agent system is described. The DESIRE modelling framework, however, contains additional facilities, such as a knowledge representation language, to support a knowledge engineering perspective on multi-agent systems design. In DESIRE, in addition to process modelling, the design of a multi-agent system entails *knowledge modelling* (Brazier, Treur & Wijngaards, 1996). Another approach to agent-

based knowledge modelling is presented in (Iglesias, Garijo, González & Velasco, 1998). A survey of such approaches can be found in (Iglesias, Garijo & González, 1999).

- In DESIRE, existing generic agent models can be used to structure specific agents. During design, relevant components in a generic model are refined by (1) more detailed analysis of the tasks of which such components are comprised and/or (2) inclusion of specific domain knowledge.In fact, a generic agent model has been developed (Brazier, Jonker & Treur, 2000) in which all agents have specific knowledge of other agents and of their needs with respect to information exchange with these other agents. The desired behaviour of individual agents and their interaction capabilities is the basis for the design of the system.

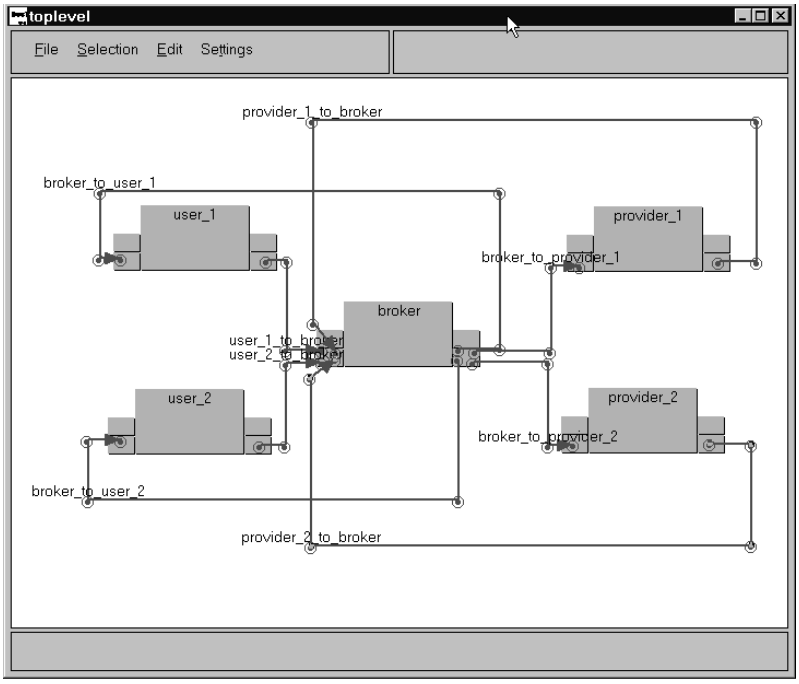

Figure 9.2: The running example multi-agent system in the DESIRE software environment.

The design process is supported by a software environment that includes graphical design tools and automated support for the translation of a specification to an operational system. The software environment includes implementation generators with which formal specifications can be translated into executable code of a prototype system. The representation at the operational level is automatically generated from the representation at the detailed level by the DESIRE software

environment. The software environment is depicted in Figure 9.2. (Note the resemblance with Figure 4.3).

In the next section, the three type of knowledge with which compositional systems are described, are further discussed, providing a static view on a DESIRE model . After that, Section 9.3 provides a dynamic view on a DESIRE model: the dynamic semantics of the model are described in terms of the semantic structure developed in this thesis.

9.2 DESIRE Types of Knowledge

In the DESIRE modelling framework, a model of a multi-agent system is described at three levels as depicted in Figure 9.1: the conceptual level, the detailed level and the operational level. At each level, three types of knowledge are distinguished: process composition, knowledge composition and the relation between process composition and knowledge composition. In this section, these three types of knowledge are described in detail. Section 9.2.1 discusses knowledge representation and composition, Section 9.2.2 describes representation of processes and process composition and Section 9.2.3 describes the relation between process composition and knowledge composition.

At the conceptual level, all three types of knowledge are represented by graphical notations supported by the DESIRE software environment. At the detailed level, sufficient detail is added to the graphical description to allow automatic prototype generation. The DESIRE modelling framework provides two alternatives for the detailed level. First, this level can be specified by filling in parameters of the graphical constructs at the conceptual level. The DESIRE software environment provides different syntax-directed editors for those graphical constructs that need additional detail. Second, a formal syntax is defined to allow specification of the detailed level in textual form. (The software environment provides a tool to automatically generate a textual representation of the graphical notation including the additional details provided via the syntax-directed editors.) The representation of a multi-agent system at the prototype level is automatically generated by the software environment.

Notwithstanding the different representations, each level contains the same types of knowledge. This section describes this knowledge avoiding all details of the precise notation used in the DESIRE modelling framework, although some examples of the notation used in DESIRE are provided.

9.2.1 Knowledge Representation and Composition

At the conceptual level, the building block for knowledge composition is an *information type*. An information type is a definition of data elements (together with the *sort* to which they belong) and relations between such elements that together form a unit of information processed in a specific multi-agent system. The set of

information types used in a specific model is chosen by the designer of the model. Information types are illustrated in Example 9.1 below.

Example 9.1. The internal information maintained by the broker agent as shown in Example 5.2 can be described at the conceptual level by the information type depicted in Figure 9.3.

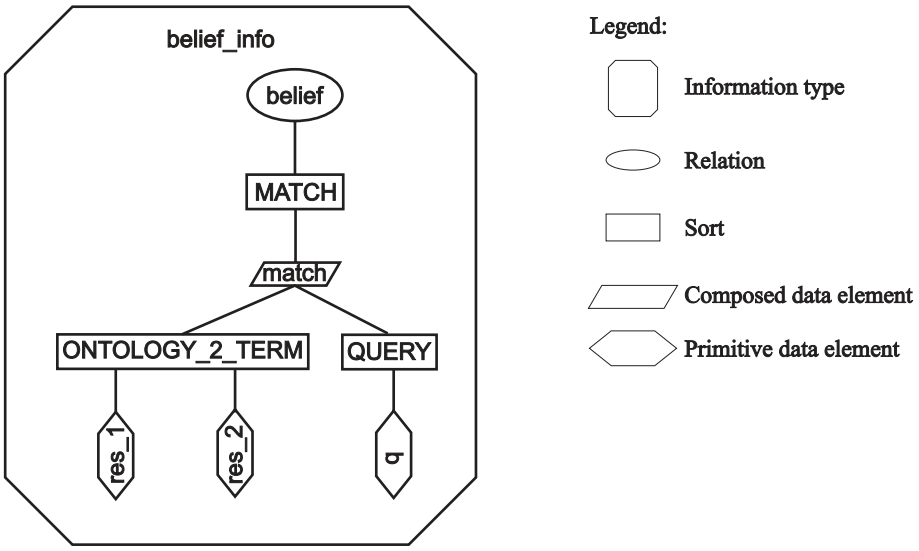

Figure 9.3: Information type for the internal information of the broker agent.

The syntactic description of the information type at the detailed level is as follows:

```
information type belief_info
        sorts
                ONTOLOGY_2_TERM, QUERY, MATCH;
        objects
                res_1, res_2: ONTOLOGY_2_TERM,
                q: QUERY;
        functions
                match: ONTOLOGY_2_TERM * QUERY -> MATCH;
        relations
                belief: MATCH;
end information type
```

Alternatively, the information type depicted in Figure 9.3 can be specified at the detailed level in the DESIRE software environment, with the information type editor shown in Figure 9.4.

As an aside, note that in a DESIRE model, an information type such as *belief_info* that defines sorts and objects and functions for these sorts, as well as relations on these sorts, is usually defined as the composition of a number of simpler sorts. This is illustrated in Section 9.2.1.3. ■

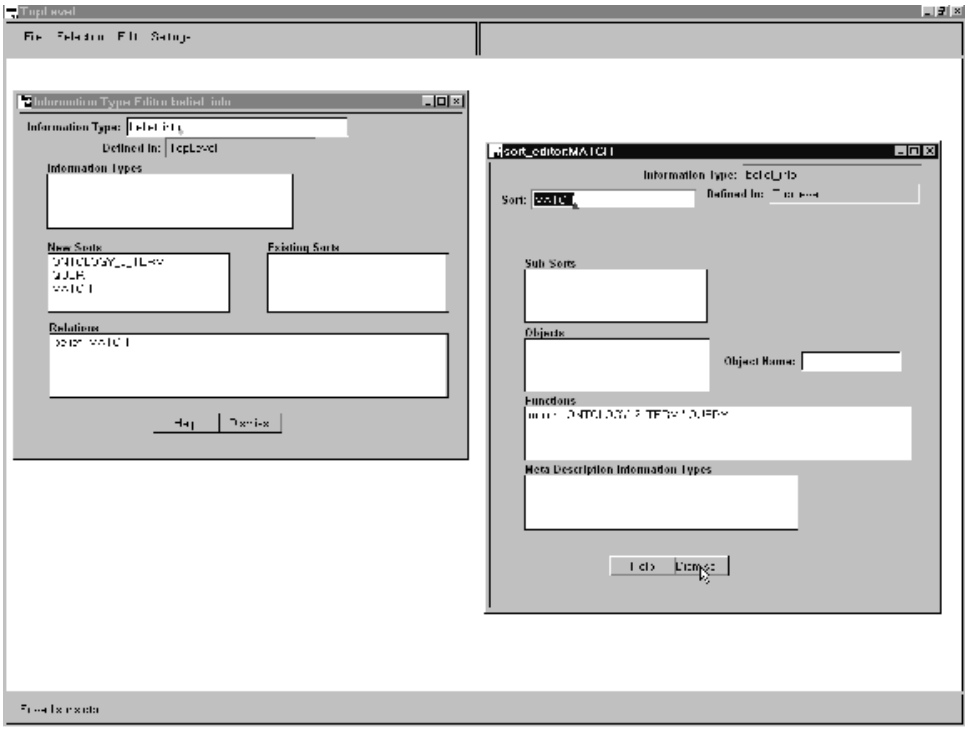

Figure 9.4: Information type editor of the DESIRE software environment.

At the detailed level, knowledge processed in a multi-agent system is described by (a restricted form of) order-sorted first-order languages. To support locality, different languages can be used for different parts of a multi-agent systems. The different order-sorted first-order languages themselves are determined by the information types for which they are used. The information types themselves (and thus, the different order-sorted first-order languages) are specified by *signatures*, which are discussed in Section 9.2.1.1. Processing of knowledge is specified by *knowledge bases*, which are discussed in Section 9.2.1.2. Knowledge composition is discussed in Section 9.2.1.3. In DESIRE, a number of *standard information types* is pre-defined. These information types are introduced in Section 9.2.1.4.

9.2.1.1 Order-Sorted First-Order Signatures

Order-sorted first-order signatures define sets of atoms for order-sorted first-order languages. Sets of atoms are used in knowledge bases (defined in Section 9.2.1.2). An order-sorted first-order signature is defined as follows:

Definition 9.2. (Order-sorted first-order signature). *An* order-sorted first-order signature *is a tuple ⟨⟨S;<⟩;Obj;Func;Pred⟩, where:*

- *⟨S;<⟩ is a finite, partially ordered set of sort names, with < irreflexive;*

- *Obj is an S-indexed family of finite sets of objects;*
- *Func is an S^+-indexed family of finite sets of function symbols;*
- *Pred is an S^*-indexed family of finite sets of predicate symbols.*

(The terminology used to capture partially ordered sorts is based on (Sernadas, Sernadas & Costa, 1995) and the simplified version thereof in (Jungclaus, 1993)). The families *Func* and *Pred* are indexed by infinite index sets (sets of strings over *S*). However, it is assumed that members of *Func* and *Pred* are only defined for a finite number of strings. In other words, if the families of sets are viewed as functions on S^+ and S^*, respectively, these functions are partial functions. An order-sorted first-order signature can be used to define a set of order-sorted first-order *terms* as follows:

Definition 9.3. (Order-sorted first-order terms). *Let $\Sigma=\langle\langle S;<\rangle;Obj;Func;Pred\rangle$ be an order-sorted first-order signature, let Var be an S-indexed family of variables and let $\leq$ be the reflexive closure of <. The family $Term(\Sigma)$ of* order-sorted first-order terms *over Σ is the S-indexed family of smallest sets closed under:*

- *If $o \in Obj_t$ with $t \leq s$ then $o \in Term(\Sigma)_s$;*
- *If $x \in Var_t$ with $t \leq s$ then $x \in Term(\Sigma)_s$;*
- *If $f \in Func_{\langle s0;\ldots;sn;s\rangle}$ and $t_0 \in Term(\Sigma)_{s0'}$ and ... and $t_n \in Term(\Sigma)_{sn'}$ and $s_0' \leq s_0$ and ... and $s_n' \leq s_n$ then $f(t_0,\ldots,t_n) \in Term(\Sigma)_s$.*

Order-sorted first-order terms are building blocks for order-sorted first-order atoms, defined as follows:

Definition 9.4. (Order-sorted first-order atoms). *Let $\Sigma=\langle\langle S;<\rangle;Obj;Func;Pred\rangle$ be an order-sorted first-order signature, let $Term(\Sigma)$ be the S-indexed family of order-sorted first-order terms over Σ, and let $\leq$ be the reflexive closure of <. Then $Atom(\Sigma)$, the set of* order-sorted first-order atoms *over Σ, is the smallest set closed under:*

- *If $p \in Pred_{\langle s0;\ldots;sn\rangle}$ and $t_0 \in Term(\Sigma)_{s0'}$ and ... and $t_n \in Term(\Sigma)_{sn'}$ and $s_0' \leq s_0$ and ... and $s_n' \leq s_n$ then $p(t_0,\ldots,t_n) \in Atom(\Sigma)$.*

An atom is called ground *iff none of the terms $t_0,\ldots,t_n$ is a variable. The set of ground atoms of a signature Σ is denoted $Gratom(\Sigma)$.*

Example 9.5. The information type belief_info, defined in the previous example, is represented at the detailed level by the following signature (as in Example 5.2, two sets of ontology terms OT_1 and OT_2 and a set Q of query terms are assumed to be given): *belief_info=$\langle\langle S;<\rangle;Obj;Func;Pred\rangle$*, with:

- *S*={ONTOLOGY_2_TERM,QUERY,MATCH};
- <=$\varnothing$;

- *Obj* an S-indexed family of sets with the following members: $Obj_{\text{ONTOLOGY\_2\_TERM}}=OT_2$ and $Obj_{\text{QUERY}}=Q$, with OT_2 and Q the sets of ontology terms and query terms introduced in Example 5.2. Assume that $\{\text{res\_1},\text{res\_2}\}\subseteq OT_2$ and $\{\text{q}\}\subseteq Q$;
- *Func* an S^+-indexed family of sets with the following member: $Func_{\langle\text{ONTOLOGY\_2\_TERM;QUERY;MATCH}\rangle}=\{\text{match}\}$, and
- *Pred* an S^*-indexed family of sets with the following member: $Pred_{\langle\text{MATCH}\rangle}=\{\text{belief}\}$.

There are no other sets in the families *Obj*, *Func*, and *Pred*.

Thus, S is a set of sort names, which are, as is customary in DESIRE, written in capitals. The sort named ONTOLOGY_2_TERM corresponds with the set of ontology terms OT_2 used in previous examples. The sort named QUERY corresponds with the set of query terms Q. The sort named MATCH contains matches between ontology terms and queries. The sorts in S are not ordered, therefore, $<$ is empty. The set of terms of the sort MATCH of this signature is $\{\text{match}(t,q) \mid t\in OT_2 \text{ and } q\in Q\}$. As an example, consider the term match(res_1,q). As $\text{res\_1}\in Obj_{\text{ONTOLOGY\_2\_TERM}}$ and $\text{q}\in Obj_{\text{QUERY}}$, $\text{res\_1}\in Term(belief\_info)_{\text{ONTOLOGY\_2\_TERM}}$ and $\text{q}\in Term(belief\_info)_{\text{QUERY}}$, and therefore, $\text{match(res\_1,q)}\in Term(belief\_info)_{\langle\text{MATCH}\rangle}$, because $\text{match}\in Func_{\langle\text{ONTOLOGY\_2\_TERM;QUERY;MATCH}\rangle}$. The set of ground atoms of this signature is $Gratom(belief\_info)=\{\text{belief(match}(t,q))\mid t\in OT_2 \text{ and } q\in Q\}$. As an example, consider belief(match(res_1,q)). As $\text{match(res\_1,q)}\in Term(belief\_info)_{\text{MATCH}}$, $\text{belief(match(res\_1,q))}\in Atom(belief\_info)$ because $\text{belief}\in Pred_{\langle\text{MATCH}\rangle}$. The atom belief(match(res_1,q)) is ground because match(res_1,q) does not contain a variable. ■

9.2.1.2 Knowledge Bases

The relationship between elements of different information types is described by *knowledge bases*, which are sets of *knowledge base rules*. A knowledge base specifies which information elements (conclusions) may be derived from which other information elements (antecedents) according to a specific inference relation. The inference relation used in DESIRE is *chaining*. A knowledge base is defined relative to three signatures: a signature for an information type that contains the input to the chaining process, a signature for an information type that contains the output of the chaining process and an information type for intermediary results of the chaining process.

Definition 9.6. (Knowledge base). *Let* $\Sigma_1=\langle\langle S_1;<_1\rangle;Obj_1;Func_1;Pred_1\rangle$, $\Sigma_2=\langle\langle S_2;<_2\rangle;Obj_2;Func_2;Pred_2\rangle$, *and* $\Sigma_3=\langle\langle S_3;<_3\rangle;Obj_3;Func_3;Pred_3\rangle$ *be three signatures. The set of* knowledge base formulae *for* Σ_1, Σ_2, *and* Σ_3 *is the smallest set of formulae* $For(\Sigma_1,\Sigma_2,\Sigma_3)$ *that comply with the following requirements:*

- *If* $a \in Atom(\Sigma_i)$ *for i=1, i=2 or i=3, then a,* **not** $a \in Lit(\Sigma_i)$;
- *If* $l \in Lit(\Sigma_i)$ *for i=1, i=2 or i=3, then* $l \in Conj$,
- *If* $l \in Lit(\Sigma_i)$ *for i=1, i=2 or i=3, then* $l \in For(\Sigma_1,\Sigma_2,\Sigma_3)$;
- *If* $\varphi_1,\varphi_2 \in Conj$, *then* φ_1 **and** $\varphi_2 \in Conj$;
- *If* $\varphi_1,\varphi_2 \in Conj$, *then* **if** φ_1 **then** $\varphi_2 \in For(\Sigma_1,\Sigma_2,\Sigma_3)$.

A knowledge base $KB(\Sigma_1,\Sigma_2,\Sigma_3)$ *for* Σ_1, Σ_2, *and* Σ_3 *is a set of knowledge base formulae.*

In this definition, Σ_1 is used for input to the chaining process, Σ_2 for intermediary results , and Σ_3 for the output.

At the conceptual level, a graphical notation may be used to represent knowledge bases. Knowledge rules are only represented at the detailed level.

Example 9.7. Suppose that, in addition to the signature *belief_info* presented in Example 9.5, two signatures *match_communication* and *resource_communication* are given, with:

- *Atom(match_communication)*={communicated_by(match(t,q),p) | $t \in OT_2$ and $q \in Q$ and $p \in Providers$};
- *Atom(resource_communication)*={to_be_communicated_to(t,u) | $t \in OT_2$ and $u \in Users$};

The following knowledge base can be defined with these signatures: (A variable $x \in Var_S$ is denoted *x: S*)

```
if      communicated_by( M: MATCH, provider_1 )
then    belief( M: MATCH );

if      belief( match( O: ONTOLOGY_2_TERM, Q: QUERY ))
        and communicated_by( Q: QUERY, user_1 )
then    to_be_communicated_to( O: ONTOLOGY_2_TERM, user_1);
```

This set of rules is a subset of *For(match_communication,belief_info,belief, resource_communication).* ■

9.2.1.3 Knowledge Composition

In DESIRE, knowledge structures can be composed in several ways. Information types can be composed to form larger information types. Operations on signatures are defined to represent composition of information types. The first operation presented in this section defines the *union* of two signatures by taking the union of the sets of sorts, the order on the sorts, and the families of object, function and predicate sets:

Definition 9.8. (Signature union). *Let* $\Sigma_1=\langle\langle S_1;<_1\rangle;Obj_1;Func_1;Pred_1\rangle$ *and* $\Sigma_2=\langle\langle S_2;<_2\rangle;Obj_2;Func_2;Pred_2\rangle$ *be two signatures such that* $<_1\cup<_2$ *is a strict partial order on* $S_1\cup S_2$. *The* union $\Sigma_1\oplus\Sigma_2$ *of* Σ_1 *and* Σ_2 *is the signature* $\langle\langle S_1\cup S_2;<_1\cup<_2\rangle;Obj_1\cup Obj_2;Func_1\cup Func_2;Pred_1\cup Pred_2\rangle$.

In this definition, the union operation for the sets of objects, functions and predicates is the union of indexed families of sets, which takes the union of corresponding sets in a family. Note that union is not assumed to be disjoint. Instead, the fact that union is not disjoint, is used to extend existing sorts. For instance, suppose the following signature is defined: $\Sigma_1=\langle\langle\{\mathsf{S1}\};\varnothing\rangle;Obj;\varnothing;\varnothing\rangle$, with *Obj* a family of sets with only one member, the set $Obj_{\mathsf{S1}}=\{a\}$. To extend the sort S1 with a new object, *b*, a new temporary signature $\Sigma_2=\langle\langle\{\mathsf{S1}\};\varnothing\rangle;Obj';\varnothing;\varnothing\rangle$ is defined, with *Obj'* a family of sets with only one member, the set $Obj'_{\mathsf{S1}}=\{b\}$. Sort S1 is extended with object *b* by taking the signature $\Sigma_1\oplus\Sigma_2=\langle\langle\{\mathsf{S1}\}\cup\{\mathsf{S1}\};\varnothing\rangle;Obj'';\varnothing;\varnothing\rangle$, with *Obj''* a family of sets with only one member, the set $Obj''_{\mathsf{S1}}=Obj_{\mathsf{S1}}\cup Obj'_{\mathsf{S1}}=\{a,b\}$. Thus, the extended signature has only one sort (S1), with two objects (*a* and *b*). If union had been disjoint, the extended signature would have two distinct sorts, one containing object *a* and the other containing object *b*.

An important feature of DESIRE is its support for meta-level architectures. In multi-agent systems (as well as in other knowledge-intensive systems) it is frequently the case that information about other information is needed. Such information is represented by lifting atoms of a specific signature to a meta-level, at which these atoms become terms of some sort *MS*. A signature in which this sort is places is called the *meta-signature of* Σ *with respect to MS*. The resulting signature is called the meta-signature of Σ with respect to *MS*, because statements *about* predicates in Σ can be formally represented using the meta signature by specifying predicates on the sort *MS*. The term to which a specific atom is lifted need not have the same function symbol as the predicate symbol of the atom. A family of 1-1-mappings between function symbols and predicated symbols describes which atom is lifted to which term.

Definition 9.9. (Meta signature). *Let* $\Sigma=\langle\langle S;<\rangle;Obj;Func;Pred\rangle$ *be an order-sorted first-order signature, let MS be a sort not in S, let Func' be an* $(S\cup\{MS\})^+$*-indexed family of sets disjoint from the members of Func, let Obj' be an ({MS})-indexed family of sets disjoint from the members of Obj, let lift be an* S^**-indexed family of 1-1-mappings between* $Pred_{\langle s0;\ldots;sn\rangle}$ *and* $Func'_{\langle s0;\ldots;sn;MS\rangle}$ *for each* $\langle s0;\ldots;sn\rangle$ *for which* $Pred_{\langle s0;\ldots;sn\rangle}$ *is defined and let lift' be a 1-1-mapping between* $Pred_{\langle\rangle}$ *and* Obj'_{MS}. *The* meta-signature $meta_{MS}(\Sigma,lift,lift')$ *of* Σ *with respect to MS, lift and lift' is the signature* $\langle\langle S\cup\{MS\};<\rangle;Obj\cup Obj';Func\cup Func';\varnothing\rangle$ *such that:*

- *For each* $\langle s0;\ldots;sn\rangle$ *and for all* $p\in Pred_{\langle s0;\ldots;sn\rangle}$, *there is an* $f\in Func'_{\langle s0;\ldots;sn;MS\rangle}$ *such that f=lift(p), and*
- *For each* $p\in Pred_{\langle\rangle}$, *there is an* $o\in Obj'_{MS}$ *such that o=lift'(p).*

In DESIRE, the function symbol for the term to which an atom is lifted, is always the same symbol as the predicate symbol of the atom. In other words, in DESIRE, the functions *lift* and *lift'* are identity functions. In the rest of this chapter, it is assumed that *lift* and *lift'* are identity functions, and $meta_{MS}(\Sigma,lift,lift')$ is denoted $meta_{MS}(\Sigma)$.

Both forms of knowledge composition can be used directly by users of the DESIRE modelling framework. (There are some restrictions, e.g. it is forbidden to lift signatures to a sort in the pre-defined, fixed set of so-called *standard sorts* introduced in Section 9.2.1.4.) The following example illustrates knowledge composition.

Example 9.10. As described in Chapter 4, the broker agent transmits knowledge on resources to user agents, i.e., it (only) transmits the name or location of a resource. Assume that, in this example, the broker agent transmits knowledge on its belief about matches between queries and information provided by brokers to user agents. In this example, an information type is used that specifies ground atoms of the form to_be_communicated_to(belief(match(res_1,q)),user_1). This information type is composed of other information types, one of which is the information type *belief_info* presented in the previous example. The information type to describe the broker's communication of its beliefs is specified as follows:

```
information type generic_agent_output
        sorts AGENT, INFO_ELEMENT;
        relations to_be_communicated_to: INFO_ELEMENT * AGENT;
end information type

information type domain_agent_output
        information type generic_agent_output;
        objects user_1, user_2, broker, provider_1, provider_2: AGENT;
end information type

information type belief_meta_info
        sorts BELIEF;
        meta-descriptions belief_info: BELIEF;
end information type

information type belief_output_info
        information type domain_agent_output;
        information type belief_meta_info;
        sub-sorts BELIEF: INFO_ELEMENT;
end information type
```

The last five lines of this specification declare the information type for the broker's communication of its beliefs. It is composed of other information types as follows.

The first four lines, starting with **information type** generic_agent_output, denote the following signature:

generic_agent_output
= $\langle\{$INFO_ELEMENT,AGENT$\};\emptyset\rangle;\emptyset;\emptyset;Pred_{\langle$INFO_ELEMENT;AGENT$\rangle}\rangle$, with
$Pred_{\langle$INFO_ELEMENT;AGENT$\rangle}=\{$to_be_communicated_to$\}$.

As the name of the information type illustrates, this information type is *generic*: it consists of concepts such as the names of agents and information elements that appear not just in the running example, but in almost all multi-agent system. This information type is therefore placed in the generic agent model and can be reused by specific agent models.

The second four lines, starting with **information type** domain_agent_output, denote the following signature:

domain_agent_output
= *generic_agent_output* $\oplus \langle\{$AGENT$\};\emptyset\rangle;Obj_{$AGENT$};\emptyset;\emptyset\rangle$
= $\langle\{$AGENT,INFO_ELEMENT$\};\emptyset\rangle;Obj_{$AGENT$};\emptyset;Pred_{\langle$INFO_ELEMENT;AGENT$\rangle}\rangle$,
with $Obj_{$AGENT$}=\{$user_1,user_2,broker,provider_1,provider_2$\}$ and
$Pred_{\langle$INFO_ELEMENT;AGENT$\rangle}=\{$to_be_communicated_to$\}$.

This information type declaration is an example of knowledge composition: the information type declaration for domain_agent_output references information type generic_agent_output, resulting in an information type that is the composition of the referenced information type and the declared information type. The sort AGENT of signature *generic_agent_output* is extended with the objects user_1, user_2, broker, info_provider_1, and info_provider_2 in signature *domain_agent_output*.

The third group of four lines, starting with **information type** belief_meta_info, is an example of knowledge composition by meta-lifting. Atoms of the signature *belief_info* given in the previous example are lifted to form elements of the sort BELIEF. The information type belief_meta_info denotes the following signature:

belief_meta_info
= $meta_{$BELIEF$}$*(belief_info)*
= $\langle\{$ONTOLOGY_2_TERM,QUERY,MATCH,BELIEF$\};\emptyset\rangle;Obj;Func;\emptyset\rangle$, with *Obj* a family of sets with the following members:
$Obj_{$ONTOLOGY_2_TERM$}=\{$res_1,res_2$\}$ and $Obj_{$QUERY$}=\{$q$\}$,
and *Func* a family of sets with the following members:
$Func_{\langle$ONTOLOGY_2_TERMS;QUERY;MATCH$\rangle}=\{$match$\}$ and $Func_{\langle$MATCH;BELIEF$\rangle}=\{$belief$\}$.

The final group of lines, starting with **information type** belief_output_info, denotes the following signature:

belief_output_info
= *domain_agent_output*$\oplus$*belief_meta_info*$\oplus$
$\langle\{$BELIEF,INFO_ELEMENT$\};\{\langle$BELIEF;INFO_ELEMENT$\rangle\}\rangle;\emptyset;\emptyset;\emptyset\rangle$
= $\langle\langle S;<\rangle;Obj;Func;Pred\rangle$,

with:

- S={ONTOLOGY_2_TERM,QUERY,MATCH,BELIEF,AGENT,INFO_ELEMENT};
- <={⟨BELIEF;INFO_ELEMENT⟩}};
- *Obj* an S-indexed family of sets with the following members: $Obj_{ONTOLOGY\_2\_TERMS}$={res_1,res_2}, and Obj_{QUERY}={q};
- *Func* an S^+-indexed family of sets with the following members: $Func_{⟨ONTOLOGY\_2\_TERMS;QUERY;MATCH⟩}$={match} and $Func_{⟨MATCH;BELIEF⟩}$={belief}, and
- *Pred* an S^*-indexed family of sets with the following member: $Pred_{⟨INFO\_ELEMENT;AGENT⟩}$={to_be_communicated_to}.

As belief(match(res_1,q)) $\in Term_{BELIEF}$(*belief_meta_info*) and BELIEF<INFO_ELEMENT, to_be_communicated_to(belief(match(res_1,q)),user_1) $\in$*Gratom*(*belief_output_info*). ■

It is possible to define multi-level meta signatures by taking the meta-signature of a meta signature, and so forth.

It is often the case that a composite information type is needed that consists of a specific information type, its meta lifting, the meta lifting of this meta lifting, and so forth. In this case, taking the composition of all these levels collapses the multi-level structure if (probably by accident) one of the meta sorts has the same name as a sort on a higher or lower level. To avoid this, a third facility for knowledge composition is introduced, called *level localisation*. The level localisation of a signature takes a signature and adds a level identifier to all sorts, objects, functions and predicates in the signature. The formal definition is as follows:

Definition 9.11. (Level localisation). *Let* Σ=⟨⟨S;<⟩;*Obj*;*Func*;*Pred*⟩ *be a signature and let* L *be a level identifier. The level localisation* $lev_L(\Sigma)$ *of* Σ *is the signature* ⟨⟨S';<'⟩;*Obj'*;*Func'*;*Pred'*⟩ *with:*

- S'={⟨s;L⟩ | $s \in S$};
- <'={⟨⟨s_1;L⟩;⟨s_2;L⟩⟩ | $s_1 < s_2$};
- *Obj' an S'-indexed family of sets with* $Obj'_{⟨s;L⟩}$=Obj_s *for each set* Obj_s *in the family Obj;*
- *Func' an S'*$^+$*-indexed family of sets with* $Func'_{⟨⟨s0;L⟩;\ldots;⟨sn;L⟩;⟨s;L⟩⟩}$=$Func_{⟨s0;\ldots;sn;s⟩}$ *for each set* $Func_{⟨s0;\ldots;sn;s⟩}$ *in the family Func;*
- *Pred' an S'***-indexed family of sets with* $Pred'_{⟨⟨s0;L⟩;\ldots;⟨sn;L⟩⟩}$=$Pred_{⟨s0;\ldots;sn⟩}$ *for each set* $Pred_{⟨s0;\ldots;sn⟩}$ *in the family Pred.*

In the DESIRE modelling framework, this form of knowledge composition is exclusively used for the specification of the relation between knowledge composition and process composition. An example of the use of this form of knowledge composition is therefore postponed.

9.2.1.4 Standard Information Types

The DESIRE framework provides a number of pre-defined, standard information types that are (indirectly) available to the user in every DESIRE model. Standard information types are provided for meta-level information and for control knowledge. There are four standard information types for meta-level information. These information types are used as follows. The DESIRE modelling framework automatically lifts certain user-specified information types to either the sort IA (which stands for *input atoms*) or OA (which stands for *output atoms*), for information types that are used in the input or output interface of a component. Both sorts IA and OA are subsorts of the sort IOA, which stands for *input and output atoms*. These three sorts, as well as a sort to represent the outcome of tests (i.e., positive or negative), are defined in an information type that is referenced by the standard information types:

Standard information type	Standard signature
Information type standard_meta **sorts** SIGN, IA, OA, IOA; **subsorts** IA, OA: IOA; **objects** pos, neg: SIGN; **end information type**	*standard_meta=⟨{S;<};Obj;∅;∅⟩, with:* • *S={*SIGN,IA,OA,IOA*};* • <={⟨IA;IOA⟩,⟨OA;IOA⟩}; • *Obj an S-indexed family of sets with the following member:* Obj_{SIGN}={pos,neg}.

As indicated by the definition of meta-lifting (Definition 9.9), the lifted signatures do not contain predicates. The four standard meta-level information types provide general predicates on the sorts IA, OA, or IOA. These predicates represent (meta-level) statements about user-defined atoms. Each of the four standard meta-level information types provides a different kind of meta-level knowledge: knowledge about which atoms are *assumptions*, which atoms are *targets* (of a reasoning process), which atoms are *required* (to complete a reasoning process), which atoms are *true*, *false*, or *known*. The four standard meta-level information types are presented in Table 9.1.

Note that no objects or functions are defined for the sorts IA, OA, IOA or TCFEC. The sorts IA, OA and IOA are automatically extended to contain lifted user-defined atoms. The sort TCFEF (which stands for *task control foci and evaluation criteria*) is discussed below.

The second type of standard information types provided by the DESIRE framework are *task control* information types. The purpose of these standard information types is to define a control lexicon as introduced in Chapter 8. The task control lexicon used by DESIRE is much more detailed than the example provided in Chapter 8. In this section, the standard task control information types and their corresponding signatures are presented without further discussion of these

information types and their semantics. Section 9.3 presents a detailed, formal semantics of the control lexicon.

Standard information type	Standard signature
Information type assumption_info **information type** standard_meta; **relations** assumption: IA * SIGN; **end information type**	*assumption_info=standard_meta⊕ ⟨{S;∅};∅;∅;Pred⟩, with:* *S={*IA,SIGN*};* *Pred an S*-indexed family of sets with the following member:* *Pred*$_{\langle IA;SIGN\rangle}$*={*assumption*}.*
Information type target_info **information type** standard_meta; **sorts** TCFEC, TARGET_TYPE; **objects** confirm, reject, determine: TARGET_TYPE; **relations** target: TCFEC * OA * TARGET_TYPE; **end information type**	*target_info=standard_meta⊕ ⟨{S;∅};Obj;∅;Pred⟩, with:* *S={*TCFEC,TARGET_TYPE*};* *Obj an S-indexed family of sets with the following member:* *Obj*$_{TARGET\_TYPE}$*={*confirm,reject,determine*};* *Pred an S*-indexed family of sets with the following member:* *Pred*$_{\langle TCFEC;OA;TARGET\_TYPE\rangle}$*={*target*}.*
information type epistemic_info **information type** standard_meta; **relations** true, false, unknown: IOA; **end information type**	*epistemic_info=standard_meta⊕ ⟨{S;∅};∅;∅;Pred⟩, with:* *S={*IOA*};* *Pred an S*-indexed family of sets with the following member:* *Pred*$_{\langle IOA\rangle}$*={*true,false,known*}.*
information type request_info **information type** standard_meta; **relations** required: IA * SIGN; **end information type**	*request_info=standard_meta⊕ ⟨{S;∅};∅;∅;Pred⟩, with:* *S={*IA,SIGN*};* *Pred an S*-indexed family of sets with the following member: Pred*$_{\langle IA;SIGN\rangle}$*={*required*}.*

Table 9.1: Standard meta-signatures.

```
information type TCpredefinedSorts
        sorts
                ActivationType,
                NextActivationType,
                LinkActivationType,
                ExtentType,
                TermType;
        subsorts
```

```
            NextActivationType: ActivationType;
        objects
            busy: ActivationType;
            idle, active, awake: NextActivationType;
            idle, uptodate, awake: LinkActivationType;
            any, any_new, all_p, every: ExtentType;
            succeeded, failed: TermType;
end information type

information type TCparts
        sorts
            Component,
            Link,
            LinkList,
            SubTaskControlFocus,
            SubEvaluationCriterium,
            OwnTaskControlFocus,
            OwnEvaluationCriterium;
        objects
            nil: LinkList;
        functions
            dot: Link * LinkList -> LinkList;
end information type

information type TCcondPrevious
        information types
            TCparts,
            TCpredefinedSorts;
        relations
            previous_evaluation: Component * SubEvaluationCriterium * ExtentType *
                                                    TermType;
            previous_component_state: Component * ActivationType ;
            previous_task_control_focus: Component * SubTaskControlFocus ;
            previous_extent: Component * ExtentType ;
            previous_own_evaluation: OwnEvaluationCriterium * ExtentType * TermType ;
            previous_own_component_state: ActivationType ;
            previous_own_task_control_focus: OwnTaskControlFocus ;
            previous_own_extent: ExtentType ;
end information type

information type TCcondCurrent
        information types
            TCparts,
            TCpredefinedSorts;
        relations
            start;
            evaluation: Component * SubEvaluationCriterium * ExtentType * TermType ;
            component_state: Component * ActivationType ;
            task_control_focus: Component * SubTaskControlFocus ;
            extent: Component * ExtentType ;
            own_evaluation: OwnEvaluationCriterium * ExtentType * TermType ;
            own_component_state: ActivationType ;
            own_task_control_focus: OwnTaskControlFocus ;
            own_extent: ExtentType ;
end information type
```

```
information type TCinput
        information types
                TCcondPrevious,
                TCcondCurrent;
end information type

information type TCoutput
        information types
                TCparts,
                TCpredefinedSorts;
        relations
                stop;
                next_component_state: Component * NextActivationType ;
                next_task_control_focus: Component * SubTaskControlFocus ;
                next_extent: Component * ExtentType ;
                next_link_state: Link * LinkActivationType ;
                next_link_sequence_state: LinkList * LinkActivationType ;
end information type

information type tc_it
        information types
                TCinput,
                TCoutput;
end information type
```

The standard task control information types correspond with the following signatures:

TCpredefinedSorts=⟨⟨S;<⟩;Obj;∅;∅⟩, with:

- *S={*ActivationType,NextActivationType,LinkActivationType,ExtentType,TermType*};*
- *<={⟨*NextActivationType;ActivationType*⟩};*
- *Obj an S-indexed family of sets with the following members:*
  - $Obj_{ActivationType}$ = *{*busy*};*
  - $Obj_{NextActivationType}$ = *{*idle,active,awake*};*
  - $Obj_{LinkActivationType}$ = *{*idle,uptodate,awake*};*
  - $Obj_{ExtentType}$ = *{*any,any_new,all_p,every*};*
  - $Obj_{TermType}$ = *{*succeeded,failed*};*

TCparts=⟨⟨S;∅⟩;Obj;Func;∅⟩, with:

- *S={*Component,Link,LinkList,SubTaskControlFocus,SubEvaluationCriterium,
  OwnTaskControlFocus,OwnEvaluationCriterium*};*
- *Obj an S-indexed family of sets with the following member:*
  - $Obj_{LinkList}$ = *{*nil*};*
- *Func an S^+-indexed family of sets with the following member:*
  - $Func_{⟨Link;LinkList;LinkList⟩}$ = *{*dot*};*

TCcondPrevious=TCparts⊕TCpredefinedSorts⊕⟨⟨S;∅⟩;∅;∅;Pred⟩, with:

- S={ActivationType,NextActivationType,LinkActivationType,ExtentType,TermType, Component,Link,LinkList,SubTaskControlFocus,SubEvaluationCriterium, OwnTaskControlFocus,OwnEvaluationCriterium};
- *Pred an S^*-indexed family of sets with the following members:*
  - $Pred_{\langle Component;SubEvaluationCriterium;ExtentType;TermType\rangle}$ = {previous_evaluation};
  - $Pred_{\langle Component;ActivationType\rangle}$ = {previous_component_state};
  - $Pred_{\langle Component;SubTaskControlFocus\rangle}$ = {previous_task_control_focus};
  - $Pred_{\langle Component;ExtentType\rangle}$ = {previous_extent};
  - $Pred_{\langle OwnEvaluationCriterium;ExtentType;TermType\rangle}$ = {previous_own_evaluation};
  - $Pred_{\langle ActivationType\rangle}$ = {previous_own_component_state};
  - $Pred_{\langle OwnTaskControlFocus\rangle}$ = {previous_own_task_control_focus};
  - $Pred_{\langle ExtentType\rangle}$ = {previous_own_extent};

TCcondCurrent=TCparts⊕TCpredefinedSorts⊕⟨⟨S;∅⟩;∅;∅;Pred⟩, with:

- S={ActivationType,NextActivationType,LinkActivationType,ExtentType,TermType, Component,Link,LinkList,SubTaskControlFocus,SubEvaluationCriterium, OwnTaskControlFocus,OwnEvaluationCriterium};
- *Pred an S^*-indexed family of sets with the following members:*
  - $Pred_{\langle\rangle}$ = {start};
  - $Pred_{\langle Component;SubEvaluationCriterium;ExtentType;TermType\rangle}$ = {evaluation};
  - $Pred_{\langle Component;ActivationType\rangle}$ = {component_state};
  - $Pred_{\langle Component;SubTaskControlFocus\rangle}$ = {task_control_focus};
  - $Pred_{\langle Component;ExtentType\rangle}$ = {extent};
  - $Pred_{\langle OwnEvaluationCriterium;ExtentType;TermType\rangle}$ = {own_evaluation};
  - $Pred_{\langle ActivationType\rangle}$ = {own_component_state};
  - $Pred_{\langle OwnTaskControlFocus\rangle}$ = {own_task_control_focus};
  - $Pred_{\langle ExtentType\rangle}$ = {own_extent};

TCinput=TCcondPrevious⊕TCcondCurrent.

TCoutput=TCparts⊕TCpredefinedSorts⊕⟨⟨S;∅⟩;∅;∅;Pred⟩ with:

- S={ActivationType,NextActivationType,LinkActivationType,ExtentType,TermType, Component,Link,LinkList,SubTaskControlFocus,SubEvaluationCriterium, OwnTaskControlFocus,OwnEvaluationCriterium};
- *Pred an S^*-indexed family of sets with the following members:*
  - $Pred_{\langle\rangle}$ = {Stop};
  - $Pred_{\langle Component;NextActivationType\rangle}$ = {next_component_state};
  - $Pred_{\langle Component;SubTaskControlFocus\rangle}$ = {next_task_control_focus};
  - $Pred_{\langle Component;ExtentType\rangle}$ = {next_extent};
  - $Pred_{\langle Link;LinkActivationType\rangle}$ = {next_link_state};
  - $Pred_{\langle LinkList;LinkActivationType\rangle}$ = {next_link_sequence_state};

tc_it=TCinput⊕TCoutput.

Note that there are no objects or functions for the sorts in information type TCparts. The objects for these sorts (as well as for the sort TCFEC in the target info standard meta-level information type) are determined by the composition structure of a component, as discussed in Section 9.2.2.1.

9.2.2 Processes and Process Composition in DESIRE

In the DESIRE modelling framework, processes distinguished in a multi-agent system are represented by components. The notion of a component in DESIRE is similar to the notion introduced in Chapter 2. In Section 9.2.2.1, the composition structure of DESIRE components is described formally.The representation of component state in DESIRE is described in Section 9.2.2.2. In DESIRE, as well as in the semantic structure developed in this thesis, information links form the glue between components, the building blocks of compositional systems. DESIRE information links are described in Section 9.2.2.3.

9.2.2.1 DESIRE Components and Composition Structure

Components are the main constructs with which compositional systems are built in DESIRE. The conceptual level focuses on the compositional structure of components and on their interfaces. Figure 9.5 shows a graphical representation of a DESIRE component. In this figure, the following parts of a component are depicted:

- The two boxes on the left side represent the input interface of the component. The interface consists of two parts, one for control input and one for domain-specific input;
- The two boxes on the right side represent the output interface of the component. The interface consists of two parts, one for control output and one for domain-specific output;
- Inside the component, a task control part and a kernel part are distinguished. In contrast to the semantic structure, task control in DESIRE is viewed as an integral part of a component. The kernel part of the component contains the internal knowledge structures of the component if the component is primitive, or subcomponents and links if the component is composed;
- In the graphical representation of a DESIRE component, the boxes that represent the task control input and kernel input interfaces are often drawn as one box, which represents the input interface of the component, and likewise for the output interfaces;
- The kernel input and output interfaces consist of (at least two) levels of object/meta-information. These levels are indicated in the graphical

representation by horizontal division of the boxes that represent the (kernel) input and output interfaces.

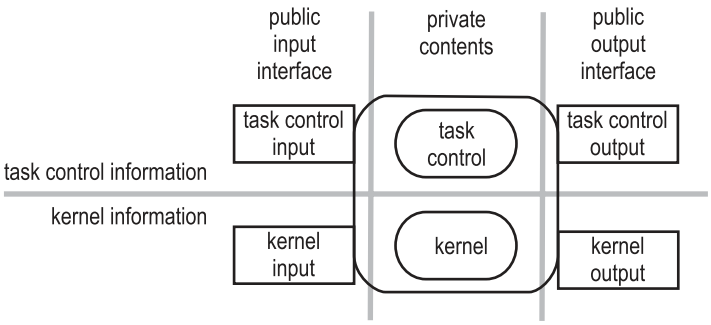

Figure 9.5: Graphical representation of a DESIRE component.

In DESIRE, primitive components may be either reasoning components (i.e., based on a knowledge base), or, components capable of performing tasks such as calculation, information retrieval, optimisation.

The composition structure of components is indicated at the conceptual level by placing components inside other components, and by drawing information links between components. (Figure 9.2 shows a number of link between components. All components in Figure 9.2 are subcomponents of a component called toplevel, as displayed in the title bar of the window shown in Figure 9.2.) At the detailed level, the compositional structure of components is represented in the textual form of the detailed level by the block structure of the syntax. In the DESIRE software environment, the compositional structure at the detailed level is automatically taken from the conceptual level and is not represented separately.

The composition structure of DESIRE components thus specified resembles the notion of a structure hierarchy as developed in Chapter 5. There are two differences:

- In DESIRE, only private links, import mediating links, export mediating links and cross-mediating links are allowed. (In a structure hierarchy, two additional types of links: link monitoring links and link modifier links, are also allowed);
- In DESIRE, there is always exactly one component, the toplevel component, that is not a subcomponent of any other component.

The four types of information links that are allowed in the DESIRE framework share a property that the other two types of links do not have: both the domain and the co-domain are components. The composition structure of DESIRE components can be described by a restricted form of a structure hierarchy, called a

DESIRE structure hierarchy, in which for each link it is required that both the domain and co-domain are components. Formally:

Definition 9.12. (DESIRE structure hierarchy). *A structure hierarchy SH=⟨Comp;Lnk;≺;dom;cdom⟩ for a component C is a* DESIRE structure hierarchy *for C if for each link I∈Lnk, dom(I),cdom(I) ∈Comp.*

Example 9.13. The example structure hierarchy presented in Example 5.10 is also a DESIRE structure hierarchy. (This structure hierarchy is used in Example 8.3 to illustrate structure hierarchies with control.) The example structure hierarchy presented in Example 5.10 is: *sh=⟨Comp;Lnk;≺;dom;cdom⟩* with

- *Comp={*toplevel,user_1,broker,ASP,OPC*};*
- *Lnk={*user_1_to_broker,broker_to_user_1*};*
- *≺ = {⟨*ASP;broker*⟩,⟨*OPC;broker*⟩,⟨*user_1;toplevel*⟩,⟨*broker;toplevel*⟩,*
  *⟨*user_1_to_broker;toplevel*⟩,⟨*broker_to_user_1;toplevel*⟩};*
- *dom = {⟨*user_1_to_broker;user_1*⟩,⟨*broker_to_user_1;broker*⟩};*
- *cdom = {⟨*user_1_to_broker;broker*⟩,⟨*broker_to_user_1;user_1*⟩};*

This DESIRE structure hierarchy represents only part of the running example multi-agent system. For brevity, and to comply with the examples presented in Chapter 8 and the rest of this chapter, only this limited DESIRE structure hierarchy is used. ■

As indicated by the definition, a DESIRE structure hierarchy is always a structure hierarchy *for a specific component*. A structure hierarchy *for a component* is a structure hierarchy that defines a forest of exactly one tree, or, in other words, a structure hierarchy in which there is exactly one component that is not a subcomponent of any other component (see Definition 5.8).

In DESIRE, at the detailed level, both in the textual representation, as well as in the software environment, additional information for each component and link is specified by the user (the intended use of these attributes is discussed in Section 9.2.2.2 and Section 9.3):

- A list of *task control focus* names. (A task control focus is a set of ground atoms on which a reasoning process focuses. The task control focus names automatically appear as objects of the sort TCFEC of the target standard meta-level information type);
- A list of *evaluation criterion* names. (An evaluation criterion is a set of ground atoms with which a reasoning process is evaluated. The evaluation criterion names automatically appear as objects of the sort TCFEC of the target standard meta-level information type);
- The name of the *initial task control focus*;

- The *initial extent type* of the component. (The extent type of a component determines to what extent reasoning processes inside the component try to fulfil their current task control focus);
- A discrete, totally-ordered set of *level identifiers*;
- Knowledge structures (Information types used internally by the component and as input and output, and knowledge bases).

For each link, the user defines a domain and co-domain information type and the *reflection type* of the link. The domain and co-domain information types determine the information that is to be transmitted if no further constraints are specified. (This is discussed in Section 9.2.2.3.) The reflection type of a link establishes object/meta relations between interface levels of *different* component. Reflection types are discussed in Section 9.2.2.3.

The additional attributes presented above are represented in the semantic structure by introducing a number of functions on the set of component or links in a structure hierarchy. Given a DESIRE structure hierarchy $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$, the following functions represent the additional attributes, where $C \in Comp$ and $I \in Lnk$:

- *TCF(C,SH)* denotes the set of task control focus names;
- *EC(C,SH)* denotes the set of evaluation criterion names;
- $TCF_{initial}(C,SH)$ denotes the name of the initial task control focus;
- $EXT_{initial}(C,SH)$: $Comp \rightarrow \{\mathsf{any},\mathsf{any\_new},\mathsf{all\_p},\mathsf{every}\}$ denotes the name of the initial extent;
- *LEV(C,SH)* denotes the set of level identifiers together with its partial order;
- $\Sigma\Sigma_{in}(C,SH)$, $\Sigma\Sigma_{int}(C,SH)$ and $\Sigma\Sigma_{out}(C,SH)$ denote *LEV(C,SH)*-indexed families of sets of signatures;
- $MF_{initial}(C,SH)$ denotes an *LEV(C,SH)*-indexed family of sets of initial meta facts, where $(MF_{initial}(C,SH))_l$ is a subset of:

  $Gratoms(target\_info \oplus assumption\_info \oplus \langle\{S;\varnothing\};Obj;\varnothing;\varnothing\rangle \oplus$
  $meta_{\langle \mathsf{IA},l\rangle}((\Sigma\Sigma_{in}(C,SH))_{l'}) \oplus meta_{\langle \mathsf{OA},l\rangle}((\Sigma\Sigma_{out}(C,SH))_{l'}))$,

  with l' the predecessor of l and:

  - $S=\{\mathsf{TCFEC}\}$;
  - Obj an S-indexed family of sets with one member: $Obj_{\langle \mathsf{TCFEC}\rangle}=TCF(C,SH) \cup EC(C,SH)$;
- *KB(C,SH)* denotes a knowledge base for the following signatures:
  - If C is a primitive component: $(\Sigma\Sigma_{in}(C,SH))_\perp$, $(\Sigma\Sigma_{int}(C,SH))_\perp$ and $(\Sigma\Sigma_{out}(C,SH))_\perp$, where $\perp$ is the bottom element of the set *LEV(C,SH)*;

  - If C is a composed component: *TCinput*, *⟨(∅;∅);∅;∅;∅⟩*, and *TCoutput*, as presented in Section 9.2.1.4.
- *ΛΛ(I,SH)* denotes an information link mapping description for *I*. (Information link mapping descriptions are defined in Section 9.2.2.3.)

In the current version of DESIRE, primitive components use exactly two levels of knowledge: one object level and one meta-level. A knowledge base can only reason about object level information. Thus, the function *KB(C,SH)* denotes a single knowledge base (not a set of knowledge bases) for the input, internal and output signatures of the bottom level of *C*. Composed components may have more than two levels. A composed component has only one knowledge base, which is the task control knowledge base. (Other knowledge bases are part of subcomponents of the composed component.)

In DESIRE, task control is considered to be an integral part of a component. Task control of a composed component is specified by the user and consists of a task control knowledge base. Task control for primitive components is standard and is not explicitly represented. (Task control for primitive components determines how a primitive component carries out its reasoning tasks. It is discussed in detail in Section 9.3.) However, as discussed in Chapter 8, in the semantic structure developed in this thesis, there is no concept of control as an integral part of components. Instead, control is introduced as a refinement of the semantic structure: specific components are dedicated to exercising control over other components, and specific links are dedicated to transmitting control information. Apart from their dedication, these components and links are normal components and links.

The gap between task control in DESIRE and control in the semantic structure is bridged by using structure hierarchies with control (see Definition 8.2). The definition of a structure hierarchy with control is repeated for ease of reference.

Definition 8.2. (Structure hierarchy with control). *Let SH=⟨Comp;Lnk;≺;dom;cdom⟩ be a structure hierarchy. This structure hierarchy is a* structure hierarchy with control *if:*

- *Comp=Contr∪Comp' with Contr and Comp' disjoint;*
- *Lnk=Lnk'∪UCLnk∪DCLnk∪ICLnk∪ECLnk with Lnk', UCLnk, DCLnk, ICLnk and ECLnk pairwise disjoint;*
- *For all C∈Contr there is a C'∈Comp, a link I_1∈ICLnk and a link I_2∈ECLnk such that C≺C', I_1≺C', and I_2≺C', dom(I_1)=C', cdom(I_1)=C, dom(I_2)=C and cdom(I_2)=C';*
- *For all C∈Comp' such that C∉Prim(SH), there is exactly one C'∈Contr such that C'≺C and for all S in Comp∪Lnk, if S≺C and S≠C', then there is an I_1∈DCLnk: dom(I_1)=C' and cdom(I_2)=S and there is an I_2∈UCLnk: dom(I_2)=S and cdom(I_2)=C';*

- *For all $I \in DCLnk$ such that $I \prec C$ for $C \in Comp$, there is a $C' \in Contr$ and an $S \in Comp \cup Lnk$ such that $dom(I)=C'$ and $cdom(I)=S$ and $S,C' \prec C$;*
- *For all $I \in UCLnk$ such that $I \prec C$ for $C \in Comp$, there is a $C' \in Contr$ and an $S \in Comp \cup Lnk$ such that $dom(I)=S$ and $cdom(I)=C'$ and $S,C' \prec C$;*
- *For all $I \in ICLnk$, there is a component $C \in Contr$ and a component $P \in Comp$ such that $C \prec P$, $dom(I)=P$ and $cdom(I)=C$;*
- *For all $I \in ECLnk$, there is a component $C \in Contr$ and a component $P \in Comp$ such that $C \prec P$, $dom(I)=C$ and $cdom(I)=P$.*

Structure hierarchies with control are used to represent control in DESIRE as follows:

- Assume that a description of the composition of a DESIRE model by a DESIRE structure hierarchy is given. A structure hierarchy with control is associated with this DESIRE structure hierarchy. There is a dedicated primitive control component for each composed DESIRE component in this structure hierarchy with control together with control links to all other subcomponents. This DESIRE structure hierarchy *with control*, and not the DESIRE structure hierarchy (without control), is the representation of the DESIRE model in the semantic structure;
- The standard task control associated with a primitive component C determines the set $Beh_{loc}(C)$.

Given a DESIRE structure hierarchy, the DESIRE structure hierarchy with control is defined as follows:

Definition 9.14. (DESIRE structure hierarchy with control). *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a DESIRE structure hierarchy. The* DESIRE structure hierarchy with control *for SH is a structure hierarchy with control $\langle Comp';Lnk';\prec';dom';cdom'\rangle$ with:*

- *$Comp'=Comp \cup Comp_{ctr}$, where $Comp_{ctr}$ is a set of component names disjoint with Comp such that for each composed component $C \in Comp$, there is a unique component name $C_{ctr} \in Comp_{ctr}$;*
- *$Lnk'=Lnk \cup Lnk_{ctr}$, where Lnk_{ctr} is a set of link names disjoint with Lnk such that:*
  - *For each component or link $S \in Comp \cup Lnk$ except the toplevel component, there are two link names $S_{UTCL}, S_{DTCL} \in Lnk_{ctr}$, and*
  - *For each composed component $C \in Comp$, there are two link names $C_{ITCL}, C_{ETCL} \in Lnk$.*
- *$\prec'=\prec \cup \prec''$, with $\prec''$ such that for each composed component $P \in Comp$:*
  - *$P_{ctr} \prec'' P$, $P_{ITCL} \prec'' P$, and $P_{ETCL} \prec'' P$, and;*
  - *For all $S \prec P$: $S_{UTCL} \prec'' P$ and $S_{DTCL} \prec'' P$.*

- *dom′=dom∪dom″, with dom″ such that for each composed component P∈Comp:*
  - *For all S≺P, dom″(S_UTCL)=S and dom″(S_DTCL)=P_ctr;*
  - *dom″(P_ITCL)=P and dom″(P_ETCL)=P_ctr;*
- *cdom′=cdom∪cdom″, with cdom″ such that for each composed component P∈Comp:*
  - *For all S≺P, cdom″(S_UTCL)=P_ctr and cdom″(S_DTCL)=S;*
  - *cdom″(P_ITCL)=P_ctr and cdom″(P_ETCL)=P;*

It is straightforward to verify that a DESIRE structure hierarchy with control complies with the requirements on a structure hierarchy presented in the definition of a structure hierarchy with control (see Chapter 8, Definition 8.2).

Example 9.15. As stated in Example 9.13, the structure hierarchy *sh* presented in Example 5.10 and re-used in Example 8.3 is also a DESIRE structure hierarchy. In Example 8.3, this structure hierarchy is extended with control components to constitute a structure hierarchy with control. The resulting structure hierarchy with control *sh′=⟨Comp′;Lnk′;≺;dom′;cdom′⟩*, which is depicted in Figure 8.2, is also a DESIRE structure hierarchy with control. (*Comp′*, *Lnk′*, *≺*, *dom′*, and *cdom′* as specified in Example 8.3.)

- *Comp′* and *Lnk′* are partitioned in pairwise disjoint sets as required by the definition of a DESIRE structure hierarcy with control. For each of the two composed components in *sh*, there are two links in *ICLnk* and *ECLnk*, respectively, and a control component in *sh′*. For each component in *sh* except the toplevel component, there are upward and downward control links;
- ⟨broker_control;broker⟩∈≺, ⟨toplevel_control;toplevel⟩∈≺, ⟨broker_ICL;broker⟩∈≺, ⟨broker_ECL;broker⟩∈≺, ⟨toplevel_ICL;toplevel⟩∈≺, and ⟨toplevel_ECL;toplevel⟩∈≺, so for each composed component *P* in *sh*, the control component P_{ctr} of *P* is a subcomponent of *P*, and the import and export control links are also links of *P*. Moreover, all upward and downward control links are links of the proper composed components;
- All links in *DCLnk, UCLnk, ICLnk* and *ECLnk* are connected as requested.

Therefore, the extended structure hierarchy is a structure hierarchy with control. ■

For the additional components in a DESIRE structure hierarchy with control, attributes such as the set of task control foci *TCF(C,SH)* have default values, as explained in Section 9.3.

9.2.2.2 The State of DESIRE Components

As explained in Chapter 2 and Chapter 5, a component (and a link) has a state, which is determined by its information contents. This state changes over time. Chapter 5 assumes that three sets of states are given for each component: one for

the input substate of the component, one for the internal substate and one for the output substate. For the application of the semantic structure to describe the semantics of multi-agent systems modelled using DESIRE, these sets are defined as valuations to keep the semantics of DESIRE as simple as possible.

The propositional language used to describe the state of a component is a local language: the set of atomic propositions of such a language only contains propositions about this component. For each component, three sets of atomic propositions are defined:

Definition 9.16. (Component information state description signature). *Let C be a component. A* component information state description signature Σ_C *of C is a triple* $\langle Prop_{in};Prop_{int};Prop_{out}\rangle$, *where* $Prop_{in}$, $Prop_{int}$ *and* $Prop_{out}$ *are sets of atomic proposition symbols used to describe the input state, internal state and output state of component C, respectively.*

It is assumed that only one information state description signature is used for each component *C*. Based on the component information state description signature of a component, a local language for the description of component states is defined. This language enables the description of relations between the input, output and internal substates that comprise the component state. As the three sets $Prop_{in}$, $Prop_{int}$ and $Prop_{out}$ in a component information state description signature are not assumed to be disjoint, in the component information state description language, proposition symbols from these sets are coloured.

Definition 9.17. (Component information state description language). *Let* $\Sigma_C=\langle Prop_{in};Prop_{int};Prop_{out}\rangle$ *be an information state description signature. The set of component information state description formulae* $For(\Sigma_C)$ *of a component C is the smallest set closed under the following restrictions:*

- *If* $p \in Prop_X$ *then* $p_X \in For(\Sigma_C)$ *for* $X \in \{in,int,out\}$;
- *If* $\varphi \in For(\Sigma_C)$ *then* $\neg\varphi \in For(\Sigma_C)$;
- *If* $\varphi,\psi \in For(\Sigma_C)$ *then* $\varphi\wedge\psi \in For(\Sigma_C)$.

The connectives $\vee$ and $\rightarrow$ are defined as usual. For each state of a component, certain propositions about (the information contents of) this state are true, while others are not, and probably for yet other propositions, the truth value is not determined. The state of a DESIRE component *C* is therefore defined as a three-valued valuation for Σ_C as follows.

Definition 9.18. (DESIRE component state). *Let C be a component with component state description signature* Σ_C. *The set of input states* $\mathcal{S}_{C,in}$ *is defined as a set of functions* $\mathcal{S}_{C,in}=\{ \nu_{C,in} \mid \nu_{C,in}: Prop_{in} \rightarrow \{0,1,u\}\}$. *The set of internal states* $\mathcal{S}_{C,int}$ *is defined as a set of functions* $\mathcal{S}_{C,int}=\{ \nu_{C,int} \mid \nu_{C,int}: Prop_{int} \rightarrow \{0,1,u\}\}$. *The set of output states* $\mathcal{S}_{C,out}$ *is defined as a set of functions* $\mathcal{S}_{C,out}=\{ \nu_{C,out} \mid \nu_{C,out}: Prop_{out} \rightarrow \{0,1,u\}\}$.

By Definition 5.1, the set of DESIRE component states for a component C is $\mathcal{S}_C=\mathcal{S}_{C,in}\times\mathcal{S}_{C,int}\times\mathcal{S}_{C,out}$. Relations $\models_3^+\subseteq\mathcal{S}_{C,X}\times Prop_X$ and $\models_3^-\subseteq\mathcal{S}_{C,X}\times Prop_X$ which precisely specify which propositional symbols are true or false in a state, respectively, can now be defined as follows (for $X\in\{in,int,out\}$): $\nu_{C,X}\models_3^+p$ if and only if $\nu_{C,X}(p)=1$ and $\nu_{C,X}\models_3^-p$ if and only if $\nu_{C,X}(p)=0$. The interpretation of formulae $\varphi\in For(\Sigma_C)$ is then defined by extending the relations $\models_3^+$ and $\models_3^-$ for formulae according to some partial semantics, e.g. strong Kleene semantics. (see (Langholm, 1988; Turner, 1990) for a discussion of such relations).

As an alternative for partial semantics, it is possible to use epistemic logic (Fagin, Halpern, Moses & Vardi, 1995) to describe the state of a component. However, as explained in the next subsection, information link mappings are characterised directly in terms of truth values. Therefore, partial semantics seems to be a better choice.

The state of components is thus described by propositional languages, which are generated from sets of atomic proposition symbols assumed to be given in Definition 9.16. To describe the semantics of DESIRE, these sets of atomic proposition symbols are themselves defined as ground atoms of specific signatures. These signatures are composed of user-defined signatures that represent domain-dependent or generic information types processes by a component, and of standard signatures provided by the DESIRE modelling framework for task control and meta reflection. The state of a DESIRE component is thus described by a propositional language, the atomic propositions of which are ground atoms of an extensive, composed signature. A separate signature is defined for each substate. The structure of these signatures enables differentiation between parts of the interfaces of a component as follows:

- The input and the output interface of a component each consist of two parts: the *kernel interface* and the *task control interface*;
- For the task control parts, fixed signatures are defined, based on the DESIRE standard task control input signature and standard task control output signature presented in Section 9.2.1.4;
- The kernel parts of both the input and output interface are *levelled interfaces*: they consist of an arbitrary number (at least two) of (meta) levels (the number of levels of a component C in a DESIRE structure hierarchy SH as well as the order of the levels is given by $LEV(C,SH)$);
- Each level of a levelled interface, except the lowest level, consists of two parts: the user part, described by a user-supplied signature, and the standard meta part, described by a standard meta signature. The lowest level only contains the user part (the user-supplied signatures of a component C in a DESIRE structure hierarchy SH are given by $\Sigma\Sigma_{in}(C,SH)$ and $\Sigma\Sigma_{out}(C,SH)$, respectively);

- The meta part consists of the meta lifting of the *user part* of the level immediately below, together with standard meta relations on the lifted user part. In terms of signatures, suppose that the user part of the *output* interface at level 1 is described by a signature Σ_1. The meta part of the output interface at level 2 is then described by $meta_{\langle OA;2\rangle}(lev_1(\Sigma_1)) \oplus \Sigma_2$, where OA is a sort in Σ_2, and Σ_2 is the standard output meta signature. According to the definition of $meta_{\langle OA;2\rangle}(lev_1(\Sigma_1))$, $meta_{\langle OA;2\rangle}(lev_1(\Sigma_1))$ does not contain any predicates. Signature Σ_2 contains predicates on the sort *OA* that represent statements on the atoms defined by Σ_1. Figure 9.6 illustrates levelled interfaces;
- The signature that describes the input or output substate of a component is the composition of all user-supplied and standard signatures, where the kernel signatures are level localised to avoid collapsing the stack of levels;
- The internal substate of a component is described by a user-supplied signature, given by $\Sigma\Sigma_{int}(C,SH)$.

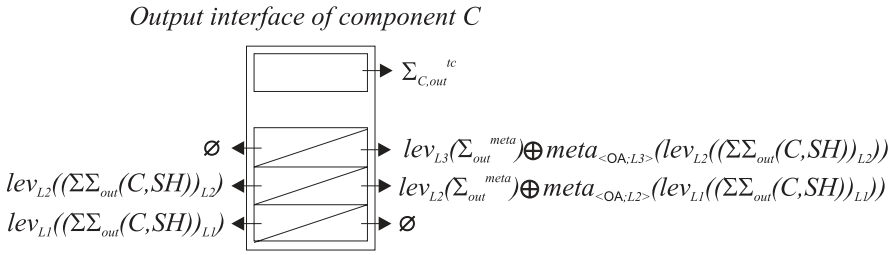

Figure 9.6: Signatures for a levelled output interface.

First, the standard signatures are defined. There are four standard signatures: the *standard input meta signature* Σ_{in}^{meta}, the *standard output meta signature* Σ_{out}^{meta}, the *task control input signature* $\Sigma_{C,in}^{tc}$, and the *task control output signature* $\Sigma_{C,out}^{tc}$. The standard input and output meta signatures are defined as follows:

Definition 9.19. (Standard input meta signature). *The* standard input meta signature *is the signature* Σ_{in}^{meta}*=assumption_info⊕target_info, where assumption_info and target_info are the signatures presented in Table 9.1.*

Definition 9.20. (Standard output meta signature). *The* standard output meta signature *is the signature* Σ_{out}^{meta}*=epistemic_info⊕request_info, where epistemic_info and request_info are the signatures presented in Table 9.1.*

Definition 9.21. (Standard task control input signature). *Let SH=⟨Comp;Lnk;≺;dom;cdom⟩ be a DESIRE structure hierarchy and let C∈Comp be a component. The* standard task control input signature *for C is the signature*

$\Sigma_{C,in}{}^{tc}$=*TCpredefinedSorts*⊕*TCparts*⊕⟨⟨*S*;∅⟩;*Obj*;∅;*Pred*⟩, *where TCpredefinedSorts and TCparts are the standard signatures presented in Section 9.2.1.4 and:*

- *S*={Link,Component,ExtentType,ActivationType,OwnTaskControlFocus};
- *Obj an S-indexed family of sets with the following members:*
  - $Obj_{\langle Link \rangle} = \{L \in Lnk \mid L \prec C\}$;
  - $Obj_{\langle Component \rangle} = \{C' \in Comp \mid C' \prec C\}$;
- *Pred an S*-indexed family of sets with the following members:*
  - $Pred_{\langle\rangle}$ = {start};
  - $Pred_{\langle ActivationType \rangle}$ = {own_component_state,};
  - $Pred_{\langle OwnTaskControlFocus \rangle}$ = {own_task_control_focus};
  - $Pred_{\langle ExtentType \rangle}$ = {own_extent};

Definition 9.22. (Standard task control output signature). *Let SH=⟨Comp;Lnk;≺;dom;cdom⟩ be a DESIRE structure hierarchy and let C be a component. The* standard task control output signature *for C is the signature* $\Sigma_{C,out}{}^{tc}$= *TCpredefinedSorts*⊕*TCparts*⊕⟨⟨*S*;∅⟩;*Obj*;∅;*Pred*⟩, *where TCpredefinedSorts and TCparts are the standard signature presented in Section 9.2.1.4 and:*

- *S*={Link,Component,OwnEvaluationCriterium,ExtentType,TermType,ActivationType, OwnTaskControlFocus};
- *Obj an S-indexed family of sets with the following members:*
  - $Obj_{\langle Link \rangle} = \{L \in Lnk \mid L \prec C\}$;
  - $Obj_{\langle Component \rangle} = \{C' \in Comp \mid C' \prec C\}$;
- *Pred an S*-indexed family of sets with the following members:*
  - $Pred_{\langle OwnEvaluationCriterium;ExtentType;TermType \rangle}$ = {own_evaluation};
  - $Pred_{\langle ActivationType \rangle}$ = {own_component_state,};
  - $Pred_{\langle OwnTaskControlFocus \rangle}$ = {own_task_control_focus};
  - $Pred_{\langle ExtentType \rangle}$ = {own_extent};
  - $Pred_{\langle\rangle}$ = {stop}.

Given the four standard signatures, the composed signatures that are used to describe the state of a DESIRE component are defined as follows:

Definition 9.23. (DESIRE component signatures). *Let SH be a DESIRE structure hierarchy, let C be a component and let prev be the immediate predecessor function for LEV(C,SH). Let* $\Sigma_{C,in}{}^{tc}$ *be the standard task control input signature for C and let* $\Sigma_{C,out}{}^{tc}$ *be the standard task control output signature for C. Then:*

- *The* level *l* kernel input interface for *C is the signature* $\Sigma_{C,in,l}{}^{ker}=lev_l((\Sigma\Sigma_{in}(C,SH))_l) \oplus lev_l(\Sigma_{in}{}^{meta}) \oplus meta_{\langle IA;l \rangle}(lev_{prev(l)}((\Sigma\Sigma_{in}(C,SH))_l))$, *if* $l>\bot$, *and* $\Sigma_{C,in,l}{}^{ker}=lev_l((\Sigma\Sigma_{in}(C,SH))_l)$ *if* $l=\bot$;
- *The* kernel input interface for *C is the signature* $\Sigma_{C,in}{}^{ker}=\bigoplus_{l \in LEV(C,SH)} \Sigma_{C,in,l}{}^{ker}$;

- *The* input signature for *C is the signature* $\Sigma_{C,in}=\Sigma_{C,in}^{tc} \oplus \Sigma_{C,in}^{ker}$;
- *The* internal signature for *C is the signature* $\Sigma_{C,int}=\Sigma\Sigma_{int}(C,SH)$;
- *The* level *l* kernel output interface for *C is the signature* $\Sigma_{C,out,l}^{ker}=lev_l((\Sigma\Sigma_{out}(C,SH))_l) \oplus lev_l(\Sigma_{out}^{meta}) \oplus meta_{\langle OA;l\rangle}(lev_{prev(l)}((\Sigma\Sigma_{out}(C,SH))_l))$ *, if* $l>\bot$*, and* $\Sigma_{C,out,l}^{ker}=lev_l((\Sigma\Sigma_{out}(C,SH))_l)$ *if* $l=\bot$;
- *The* kernel output interface for *C is the signature* $\Sigma_{C,out}^{ker}=\bigoplus_{l \in LEV(C,SH)} \Sigma_{C,out,l}^{ker}$;
- *The* output signature for *C is the signature* $\Sigma_{C,out}=\Sigma_{C,out}^{tc} \oplus \Sigma_{C,out}^{ker}$.

Definition 9.23 precisely specifies the signatures used to describe the state of those DESIRE components that are available to users. However, a DESIRE model is represented in the semantic structure by a structure hierarchy with control, which not only contains these components, but also contains links and additional components. The following signatures are used to describe the state of the additional components. Signatures to describe the state of links are described in Section 9.2.2.3.

Definition 9.24. (DESIRE control component state description signatures). *Let SH be a structure hierarchy with control for a DESIRE structure hierarchy SH'. Let C be a control component. The input signature for C is the signature* $\Sigma_{C,in}=TCinput\Sigma_{in}^{tc}$*, where* Σ_{in}^{tc} *is the standard task control input signature given in Defintion* 9.21. *The output signature for C is the signature* $\Sigma_{C,out}=\Sigma_{out}^{tc}$*, where* Σ_{out}^{tc} *is the standard task control output signature given in Defintion* 9.22.

Ground atoms of the input, internal and output signatures for a component *C* are used as proposition symbols to describe the state of *C* as follows:

Definition 9.25. (DESIRE component state description signature). *Let* $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ *be a structure hierarchy with control for a DESIRE structure hierarchy SH' and let* $C \in Comp$ *be a component with input signature* $\Sigma_{C,in}$*, internal signature* $\Sigma_{C,int}$*, and output signature* $\Sigma_{C,out}$*. The* DESIRE component state description signature *for component C is a triple* $\Sigma_C=\langle Prop_{C,in};Prop_{C,int};Prop_{C,out}\rangle$ *with*

- $Prop_{C,in}=Gratom(\Sigma_{C,in})$;
- $Prop_{C,out}=Gratom(\Sigma_{C,out})$;
- *If C is a composed component:* $Prop_{C,int}=\varnothing$;
- *If C is a primitive component:* $Prop_{C,int}=Gratom(\Sigma_{C,int})$.

All signatures used to describe the state of a component are depicted in Figure 9.5.

Example 9.26. Suppose that a user of the DESIRE modelling framework has defined a component *C* with a two-level output interface. The levels are identified

with the identifiers *L1* and *L2*, with *L1*<*L2*. The user-supplied signatures for the two levels are: $\Sigma_{C,out,L1}{}^{user}$=*belief_info* (defined in Example 9.5) and $\Sigma_{C,out,L2}{}^{user}$=$meta_{\text{BELIEF}}$(*belief_info*)⊕⟨⟨*S*;<⟩;*Obj*;∅;*Pred*⟩, with:

- *S*={BELIEF,INFO_ELEMENT,AGENT};
- <={⟨BELIEF,INFO_ELEMENT⟩};
- *Obj an S-indexed family of sets with the following member:*
  - Obj_{AGENT}={user_1,user_2,broker,provider_1,provider_2};
- *Pred an S*-indexed family of sets with the following member:*
  - $Pred_{\langle \text{INFO\_ELEMENT;AGENT} \rangle}$={to_be_communicated_to}.

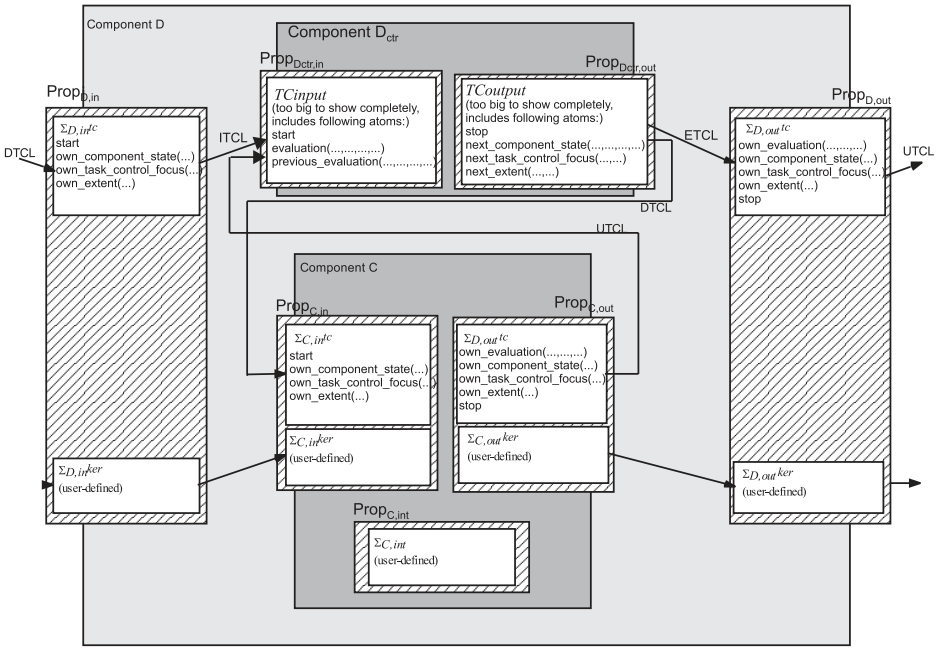

Figure 9.7: DESIRE signatures.

The complete signature for level *L2* of the output interface of *C* is the following signature:

$$\Sigma_{C,out,L2}{}^{ker} = lev_{L2}(\Sigma_{C,out,L2}{}^{user}) \oplus lev_{L2}(\Sigma_{out}{}^{meta}) \oplus meta_{\langle \text{OA};L2 \rangle}(lev_{L1}(\Sigma_{C,out,L1}{}^{user}))$$
$$= \langle\langle S;< \rangle;Obj;Func;Pred \rangle,$$

with:

- *S*={⟨SIGN;*L2*⟩;⟨IA;*L2*⟩;⟨OA;*L2*⟩;⟨IOA;*L2*⟩;⟨ONTOLOGY_2_TERM;*L1*⟩; ⟨ONTOLOGY_2_TERM;*L2*⟩;⟨QUERY;*L1*⟩;⟨QUERY;*L2*⟩;⟨MATCH;*L1*⟩;⟨MATCH;*L2*⟩; ⟨BELIEF;*L2*⟩;⟨INFO_ELEMENT;*L2*⟩;⟨AGENT;*L2*⟩};

- $<=\{\langle\langle IA;L2\rangle;\langle IOA;L2\rangle\rangle;\langle\langle OA;L2\rangle;\langle IOA;L2\rangle\rangle;\langle\langle BELIEF;L2\rangle;\langle INFO\_ELEMENT;L2\rangle\rangle\}$;
- *Obj an S-indexed family of sets with the following members:*
  - $Obj_{\langle ONTOLOGY\_2\_TERM;L1\rangle}=\{res\_1,res\_2\}$;
  - $Obj_{\langle ONTOLOGY\_2\_TERM;L2\rangle}=\{res\_1,res\_2\}$;
  - $Obj_{\langle QUERY;L1\rangle}=\{q\}$;
  - $Obj_{\langle QUERY;L2\rangle}=\{q\}$;
  - $Obj_{\langle SIGN;L2\rangle}=\{pos,neg\}$;
  - $Obj_{\langle AGENT;L2\rangle}=\{user\_1,user\_2,broker,provider\_1,provider\_2\}$;
- *Func an S^+-indexed family of sets with the following members:*
  - $Func_{\langle\langle ONTOLOGY\_2\_TERM;L1\rangle;\langle QUERY;L1\rangle;\langle MATCH;L1\rangle\rangle}=\{match\}$;
  - $Func_{\langle\langle ONTOLOGY\_2\_TERM;L2\rangle;\langle QUERY;L2\rangle;\langle MATCH;L2\rangle\rangle}=\{match\}$;
  - $Func_{\langle\langle MATCH;L1\rangle;\langle OA;L2\rangle\rangle}=\{belief\}$;
  - $Func_{\langle\langle MATCH;L2\rangle;\langle BELIEF;L2\rangle\rangle}=\{belief\}$;
- *Pred an S^*-indexed family of sets with the following members:*
  - $Pred_{\langle\langle IOA;L2\rangle\rangle}=\{true,false,unknown\}$;
  - $Pred_{\langle\langle IA;L2\rangle;\langle SIGN;L2\rangle\rangle}=\{required\}$;
  - $Pred_{\langle\langle INFO\_ELEMENT;L2\rangle;\langle AGENT;L2\rangle\rangle}=\{to\_be\_communicated\_to\}$;

Thus, the sorts ONTOLOGY_2_TERM, QUERY and MATCH appear twice as level-localised versions of this signature. ■

9.2.2.3 Information Links

In DESIRE, as well as in the semantic structure developed in this thesis, information links are the glue between components in a compositional system. In this section, the representation of DESIRE links in the semantic structure is discussed. According to Chapter 5, the state, the domain and co-domain, and the information link mapping of a link have to be described. The domain and co-domain of an information link are described by the notion of a structure hierarchy. As discussed in Chapter 5, in the semantic structure for each information link I, a set of information link states $\mathcal{S}_I$ is distinguished. For each information link state, certain propositions about (the information contents of) this state are true, while others are not, and probably for yet other propositions, the truth value is not known. The state of an information link is described by the *information link state description signature* for the link, which is assumed to be unique:

Definition 9.27. (Information link state description signature). *Let I be an information link. The information link state description signature Σ_I is a set of atomic proposition symbols.*

In DESIRE, the information link state description signature is fixed and cannot be specified by users of the DESIRE modelling framework. The precise set of atomic proposition symbols chosen for DESIRE is defined as follows:

Definition 9.28. (DESIRE link state description signature). *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a structure hierarchy with control for a DESIRE structure hierarchy SH' and let $I \in Lnk$ be an information link. The* DESIRE link state description signature *for information link I is the set $Gratoms(\Sigma_I)$, where $\Sigma_I=\langle\langle S;\varnothing\rangle;Obj;\varnothing;Pred\rangle$, with:*

- *S={*LinkActivationType,Link*};*
- *Obj an S-indexed family of sets with the following members:*
  - $Obj_{\text{LinkActivationType}}$*={*idle,uptodate,awake*};*
  - Obj_{Link}*={I};*
- *Pred an S^*-indexed family of sets with the following member: $Pred_{\langle \text{Link;LinkActivationType}\rangle}$={*link_state*}.*

This definition is analogous to Definition 9.25, which defines the component state description signatures used in DESIRE. The sort LinkActivationType in Σ_I contains exactly the same elements as the sort LinkActivationType in information type TCpredefinedSorts given in Section 9.2.1.4.

Using the information link state description signature, states of an information link are described by the following language:

Definition 9.29. (Information link state description language). *Let Σ_I be an information link state description signature. The set of information link state description formulae $For(\Sigma_I)$ of a link I is the smallest set closed under the following restrictions:*

- *If $p \in Gratoms(\Sigma_I)$ then $p \in For(\Sigma_I)$;*
- *If $\varphi \in For(\Sigma_I)$ then $\neg\varphi \in For(\Sigma_I)$;*
- *If $\varphi, \psi \in For(\Sigma_I)$ then $\varphi \wedge \psi \in For(\Sigma_I)$.*

The connectives $\vee$ and $\rightarrow$ are defined as usual. In the semantic structure, as set of link states is assumed to be given. Thus, such a set has to be defined to fully describe the semantics of DESIRE. Similar to component states, the set of link states $\mathcal{S}_I$ for a link *I* defined as a set of valuations of the atomic propositions generated by Σ_I as follows: $\mathcal{S}_I=\{ v_I \mid v_I: Gratoms(\Sigma_I) \rightarrow \{0,1\}\}$. As this definition indicates, in DESIRE, links states are not partial, therefore a link state is a mapping into *{0,1}* instead of *{0,1,u}*. To keep the semantics of components and links uniform, relations $\models_3^+ \subseteq \mathcal{S}_I \times For(\Sigma_I)$ and $\models_3^- \subseteq \mathcal{S}_I \times For(\Sigma_{I,m})$ are used to interpret the information link state description language. These relations are defined as follows (for $p \in Gratoms(\Sigma_I)$): $v_I \models_3^+ p$ if and only if $v_I(p)=1$ and $v_I \models_3^- p$ if and only if $v_I(p)=0$. The interpretation of formulae $\varphi \in For(\Sigma_I)$ is then defined by extending the relations $\models_3^+$ and $\models_3^-$ for formulae according to some partial semantics, similar to component states.

The information link mapping is described by a *information link mapping description*. Before this notion is defined, first the definition of an information link mapping is repeated for ease of reference.

Definition 5.6. (Information link mapping). *Let I be an information link. An information link mapping for I is a relation defined as follows:*

- $\lambda_I \subseteq (\mathcal{S}_{dom(I),out} \times \mathcal{S}_{dom(I),out}) \times \mathcal{S}_I^4 \times (\mathcal{S}_{cdom(I),in} \times \mathcal{S}_{cdom(I),in})$, *if I is a private link, or*
- $\lambda_I \subseteq (\mathcal{S}_{dom(I),in} \times \mathcal{S}_{dom(I),in}) \times \mathcal{S}_I^4 \times (\mathcal{S}_{cdom(I),in} \times \mathcal{S}_{cdom(I),in})$, *if I is an import mediating link, or*
- $\lambda_I \subseteq (\mathcal{S}_{dom(I),out} \times \mathcal{S}_{dom(I),out}) \times \mathcal{S}_I^4 \times (\mathcal{S}_{cdom(I),out} \times \mathcal{S}_{cdom(I),out})$, *if I is an export mediating link, or*
- $\lambda_I \subseteq (\mathcal{S}_{dom(I),in} \times \mathcal{S}_{dom(I),in}) \times \mathcal{S}_I^4 \times (\mathcal{S}_{cdom(I),out} \times \mathcal{S}_{cdom(I),out})$, *if I is a cross-mediating link, or*
- $\lambda_I \subseteq (\mathcal{S}_{dom(I),out} \times \mathcal{S}_{dom(I),out}) \times \mathcal{S}_I^4 \times (\mathcal{S}_{cdom(I)} \times \mathcal{S}_{cdom(I)})$, *if I is a link modifier link, or*
- $\lambda_I \subseteq (\mathcal{S}_{dom(I)} \times \mathcal{S}_{dom(I)}) \times \mathcal{S}_I^4 \times (\mathcal{S}_{cdom(I),in} \times \mathcal{S}_{cdom(I),in})$, *if I is a link monitoring link.*

An information link mapping is a set of octets of states. Such a set can be determined by propositions about the states that constitute (the octets in) the set. An *information link mapping description* consists of octets of proposition symbols, together with a truth value for each proposition symbol. The truth value further specifies which states are to be part of the information link mapping as follows: if an information link mapping description Λ_I for a link I contains a proposition symbol p together with truth value *1*, then all states for which p is true have to be part of the information link mapping described by Λ_I. If Λ_I contains a proposition symbol p together with truth value *0*, then all states for which p is false have to be part of the information link mapping described by Λ_I. If Λ_I contains a proposition symbol p together with truth value u, then all states for which p is neither true nor false have to be part of the information link mapping described by Λ_I. A pair consisting of a proposition symbol (from a specific set of propositions) together with a truth value is called a *basic information element* and is defined as follows:

Definition 9.30. (Basic information element). *A* basic information element *IE(P) of a set of proposition symbols P is an element of* $(P \cup \{\mathsf{T}\}) \times \{0,1,u\}$. *The set of all basic information elements of a set of proposition symbols P is denoted* $IE(P) = (P \cup \{\mathsf{T}\}) \times \{0,1,u\}$. *For notational convenience, a basic information element such as e.g.* $\langle p;1 \rangle$ *is denoted [p:1].*

The special proposition symbol T, which stands for 'true', can be used to specify that any state from a set has to be part of an information link mapping.

Basic information elements denote states. The following definition precisely specifies which states are denoted by a basic information element:

Definition 9.31. (Satisfaction of information elements). *A (sub)state* σ satisfies *an information element IE, denoted* $\sigma \models_{ie} IE$, *iff:*

- $IE=[p:1]$ *and* $\sigma \models_3^+ p$, *or*
- $IE=[p:0]$ *and* $\sigma \models_3^- p$, *or*
- $IE=[p:u]$ *and* $\sigma \not\models_3^+ p$ *and* $\sigma \not\models_3^- p$, *or*
- $\sigma \models_{ie} [\top:t]$ *for any* σ *and* $t \in \{0,1,u\}$.

The stage is now set to define the notion of an *information link mapping description*. An information link mapping description is a set of octets of basic information elements for specific sets of proposition symbols, which depends on the type of the information link:

Definition 9.32. (Information link mapping description). *Let I be an information link. An* information link description *for I is a relation defined as follows:*

- $\Lambda_I \subseteq (IE(Prop_{dom(I),out}) \times IE(Prop_{dom(I),out})) \times IE(Prop_I)^4 \times (IE(Prop_{cdom(I),in}) \times IE(Prop_{cdom(I),in}))$, *if I is a private link, or*
- $\Lambda_I \subseteq (IE(Prop_{dom(I),in}) \times IE(Prop_{dom(I),in})) \times IE(Prop_I)^4 \times (IE(Prop_{cdom(I),in}) \times IE(Prop_{cdom(I),in}))$, *if I is an import mediating link, or*
- $\Lambda_I \subseteq (IE(Prop_{dom(I),out}) \times IE(Prop_{dom(I),out})) \times IE(Prop_I)^4 \times (IE(Prop_{cdom(I),out}) \times IE(Prop_{cdom(I),out}))$, *if I is an export mediating link, or*
- $\Lambda_I \subseteq (IE(Prop_{dom(I),in}) \times IE(Prop_{dom(I),in})) \times IE(Prop_I)^4 \times (IE(Prop_{cdom(I),out}) \times IE(Prop_{cdom(I),out}))$, *if I is a cross-mediating link, or*
- $\Lambda_I \subseteq (IE(Prop_{dom(I),out}) \times IE(Prop_{dom(I),out})) \times IE(Prop_I)^4 \times (IE(Prop_{cdom(I)}) \times IE(Prop_{cdom(I)}))$, *if I is a link modifier link, or*
- $\Lambda_I \subseteq (IE(Prop_{dom(I)}) \times IE(Prop_{dom(I)})) \times IE(Prop_I)^4 \times (IE(Prop_{dom(I),in}) \times IE(Prop_{dom(I),in}))$, *if I is a link monitoring link.*

An information link mapping description describes the information link mapping that *satisfies* the description. An information link mapping satisfies an information link mapping description if for each octet of states in the information link mapping there is at least one octet of basic information elements in its description such that (i) the first, third and seventh states in the octet of states satisfies the first, third and seventh basic information elements in the octet of basic information elements, and (ii) if the first, third and seventh state satisfy the corresponding basic information elements, then the other states satisfy their corresponding basic information elements. The second requirement shows that in an information link mapping, the first, third and seventh states are states in which enabling conditions for information transmission must hold, while in the other states, results of

transmission are present. Satisfaction of an information link mapping description as described above is formalised as follows:

Definition 9.33. (Satisfaction of an information link mapping description). *An information link mapping λ_L for a link L* satisfies *an information link mapping description Λ_L, denoted $\lambda_L \models_{Lnk} \Lambda_L$, if each octet $\langle\langle\sigma_1;\sigma_2\rangle;\langle\sigma_3;\sigma_4;\sigma_5;\sigma_6\rangle;\langle\sigma_7;\sigma_8\rangle\rangle \in \lambda_L$ fulfills the following requirements:*

- *There is an octet $\langle\langle IE_1;IE_2\rangle;\langle IE_3;IE_4;IE_5;IE_6\rangle;\langle IE_7;IE_8\rangle\rangle \in \Lambda_L$ such that $\sigma_1 \models_{ie} IE_1$, $\sigma_3 \models_{ie} IE_3$ and $\sigma_7 \models_{ie} IE_7$;*
- *If $\sigma_1 \models_{ie} IE_1$, $\sigma_3 \models_{ie} IE_3$ and $\sigma_7 \models_{ie} IE_7$, then $\sigma_2 \models_{ie} IE_2$, $\sigma_4 \models_{ie} IE_4$, $\sigma_5 \models_{ie} IE_5$, $\sigma_6 \models_{ie} IE_6$ and $\sigma_8 \models_{ie} IE_8$.*

Example 9.34. In Chapter 5, example 5.2 presents an information link mapping for the link broker_to_user_1 from broker to user_1 in the running example. For ease of reference, the definition of the information link mapping is repeated (where *trans* is a function from OT_2 to OT_1 that translates ontology terms in OT_2 to OT_1, which is assumed to be given):

$\lambda_{\text{broker\_to\_user\_1}}$ = {⟨⟨to_be_communicated_to(t,user_1);just_communicated_to(t,user_1)⟩;
⟨awake_and_empty;active_and_contents(t);active_and_contents(t);
awake_and_empty⟩;⟨ready_for_information;
communicated_by(t',broker)⟩⟩ | $t \in OT2$, $t' \in OT_1$ *and* $t'=trans(t)$}.

As $\lambda_{\text{broker\_to\_user\_1}}$ is a private link, an information link mapping description for $\lambda_{\text{broker\_to\_user\_1}}$ contains basic information elements to describe the output state of the broker, the link itself and the input of user_1. The following sets of proposition symbols are used to describe these states. As in Example 5.2, two sets of ontology terms OT_1 and OT_2 and a set Q of query terms are assumed to be given. The sets *Users* and *Providers* are defined as follows: *Users*={user_1,user_2} and *Providers*={provider_1,provider_2}.

- $Prop_{\text{broker},out}$ = {to_be_communicated_to(t,u) | $t \in OT_2$ *and* $u \in Users$} ∪
{just_communicated_to(t,u) | $t \in OT_2$ *and* $u \in Users$}.
- $Prop_{\text{broker\_to\_user\_1}}$ = {awake_and_empty,active_and_contents(t) | $t \in OT_2$}.
- $Prop_{\text{user\_1},in}$ = {communicated_by(t,broker) | $t \in OT_1$} ∪
{ready_for_information}.

A relation $\models_3^+$ is defined such that $\sigma \models_3^+ p$ if and only iff the name of state σ exactly matches p, i.e., for all $t \in OT_2$ and for all $u \in User$, to_be_communicated_to(t,u)$\models_3^+$to_be_communicated_to(t,u), and likewise for the other states and propositions. A relation $\models_3^-$ is defined such that $\sigma \models_3^- p$ if and only iff the name of state σ differs from p, i.e., for all $t,t' \in OT_2$ and for all $u,u' \in User$,

to_be_communicated_to(t,u)$\models_3^-$to_be_communicated_to(t',u') for $t \neq t'$ and $u \neq u'$, and likewise for the other states and propositions.

The information link mapping given above can be described by the following information link mapping description:

$\Lambda_{\text{broker\_to\_user\_1}} = \{\langle\{[\text{to\_be\_communicated\_to}(t,\text{user\_1}):1];[\text{just\_communicated\_to}(t,\text{user\_1}):1]\rangle;$
$\langle[\text{awake\_and\_empty}:1];[\text{active\_and\_contents}(t):1];[\text{active\_and\_contents}(t):1];$
$[\text{awake\_and\_empty}:1]\rangle;\langle[\text{ready\_for\_information}:1];$
$[\text{communicated\_by}(t',\text{broker}):1]\rangle\rangle \mid t \in OT2,\ t' \in OT_1 \text{ and } t' = trans(t)\}.$

It is straightforward to check that $\lambda_{\text{broker\_to\_user\_1}} \models_{Lnk} \Lambda_{\text{broker\_to\_user\_1}}$. ■

Example 9.35. As stated in Example 5.2 in Chapter 5, the broker agent maintains beliefs about matches between ontology terms from the set OT_2 and query terms from the set Q. The set $\mathcal{S}_{\text{broker},int}$ of internal states of the broker agents is defined in Example 5.2 as: $\mathcal{S}_{\text{broker},int} = \{\text{belief}(\text{match}(t,q)) \mid t \in OT_2 \text{ and } q \in Q\}$. The set *Gratom(belief_info)*, with *belief_info* as defined in Example 9.5, is used to describe the internal state of the broker agent. The relations $\models_3^+$ and $\models_3^-$ are defined as follows:

- $\models_3^+$ is the smallest relation such that for all $t \in OT_2$ and for all $q \in Q$, $\text{belief}(\text{match}(t,q)) \models_3^+ \text{belief}(\text{match}(t,q))$.
- $\models_3^-$ is the smallest relation such that for all $t,t' \in OT_2$ and for all $q,q' \in Q$, $\text{belief}(\text{match}(t,q)) \models_3^- \text{belief}(\text{match}(t',q'))$ if $t \neq t'$ and $q \neq q'$.

Thus, this example assumes that the broker agent can only believe one match at a time. (This assumption enables a direct connection between proposition symbols and names of states.)

Suppose that there is a sixth component, called exernal_world, in the running example multi-agent system, and that this component contains a database of matches between ontology terms and query terms. The state of the external world can (partially) be described by proposition symbols match(t,q) for $t \in OT_2$ and $q \in Q$. Moreover, suppose that there are links from external_world to the broker such that (a subcomponent of) the broker agent can consult the database in external_world to maintain beliefs about matches. (The precise configuration of links from external_world to (a subcomponent of) broker is deliberately omitted to keep the example concise.) The broker agent believes that there is a match between an ontology term t and a query term q if and only if there is a state of external_world for which match(t,q) is true. Thus, the broker does not believe that there is a match between t and q either if there is a state of external_world for which match(t,q) is false, or if there is a state of external_world for which match(t,q) is not true and not false.

In this example, in which the exact structure of the links is omitted, focus is on the first and eighth state in an information link mapping that connects the database to the internal beliefs of the broker agent. (As stated in Chapter 5, the first and

eighth state of an information link mapping specify which information is transmitted, while the other states specify enabling conditions, results and details of the transmission process.) Consequently, in this example the information link mapping description is presented as a binary relation as follows:

$$\begin{aligned}\Lambda \;=\; & \{\, \langle [\mathsf{match}(t,q):1];[\mathsf{belief}(\mathsf{match}(t,q)):1]\rangle \mid t \in OT_2,\, q \in Q \,\} \cup \\ & \{\, \langle [\mathsf{match}(t,q):0];[\mathsf{belief}(\mathsf{match}(t,q)):0]\rangle \mid t \in OT_2,\, q \in Q \,\} \cup \\ & \{\, \langle [\mathsf{match}(t,q):u];[\mathsf{belief}(\mathsf{match}(t,q)):0]\rangle \mid t \in OT_2,\, q \in Q \,\}.\end{aligned}$$

This information link mapping description is satisfied by the following information link mapping: (In this example, also for information link mappings focus is on the first and eighth factors. Consequently, the information link mapping is presented as a binary relation.)

$$\begin{aligned}\lambda \;=\; & \{\, \langle \sigma;\mathsf{belief}(\mathsf{match}(t,q))\rangle \mid \sigma \models_3^{+} \mathsf{match}(t,q) \text{ for } t \in OT_2,\, q \in Q \text{ and} \\ & \sigma \models_3^{-} \mathsf{match}(t',q') \text{ or } \sigma \not\models_3^{+} \mathsf{match}(t',q') \text{ and } \sigma \not\models_3^{-} \mathsf{match}(t',q') \text{ for } t \neq t' \in OT_2 \\ & \text{and } q \neq q' \in Q \,\}.\end{aligned}$$

Thus, $\lambda \models_{Lnk} \Lambda$, which can be proven as follows. Let $\langle \sigma;\mathsf{belief}(\mathsf{match}(t,q))\rangle$ be an element of λ, for arbitrary $t \in OT_2$ and $q \in Q$. Take $IE_1 = [\mathsf{match}(t,q):1]$ and $IE_2 = [\mathsf{belief}(\mathsf{match}(t,q)):1]$. Then $\langle IE_1;IE_2\rangle \in \Lambda$, $\sigma \models_{ie} IE_1$ because by the definition of λ given above, $\sigma \models_3^{+} \mathsf{match}(t,q)$, and $\mathsf{belief}(\mathsf{match}(t,q)) \models_{ie} IE_2$ because by the definition of $\models_3^{+}$ at the beginning of this example, $\mathsf{belief}(\mathsf{match}(t,q)) \models_3^{+} \mathsf{belief}(\mathsf{match}(t,q))$. Thus, for each pair of states $\langle \sigma_1;\sigma_2\rangle$ in λ, there is a pair of basic information elements $\langle IE_1;IE_2\rangle$ in Λ such that $\sigma_1 \models_{ie} IE_1$, and if $\sigma_1 \models_{ie} IE_1$, then $\sigma_2 \models_{ie} IE_2$. ■

In DESIRE, at the conceptual level, for a link only the name, domain, co-domain and the object/meta level of both the domain and co-domain to which the link connects, is represented. At the detailed level, the reflection type of the link is represented, as well as, for some reflection types, the names of the signatures to which it is constrained, either in textual form or by fill-in forms in the software environment. Additionally, a restricted form of an information link mapping is provided in textual form (The software environment uses a segment of the syntax for the textual form of the detailed level here.)

As stated in Section 9.2.2.1, the reflection type of a link establishes object/meta relations between levels of the interfaces of the domain and co-domain of a link. The reflection type also partly determines to which signatures a link is restricted. (Restriction of a link to a specific information type is used in the specification of the information link mapping description, discussed below.) The table below lists examples of link types for all reflection types. Some reflection types are only allowed for specific links, i.e. private links or mediating links. The reflection types differ in the object/meta relation between the levels of the domain and co-domain they specify. Some reflection types, e.g. links of the O-O type, specify that the specific object/meta level of the domain of the link is at the same level as the object/meta level of the co-domain. Other reflection types, e.g. links of the type O-

A, specify *downward reflection*: the object/meta level of the domain of the link is at a higher level than the object/meta level of the co-domain. Yet other reflection types, e.g. links of the type E-O, specify *upward reflection*: the object/meta level of the domain of the link is at a lower level than the object/meta level of the co-domain. The signature names for the domain and co-domain refer to the standard signatures introduced in Section 9.2.1.4.

Reflection Type	Link type	Domain signature	Co-domain signature	Reflection
O-O	Private, mediating	(user-selected)	(user-selected)	None
O-A	Private, import mediating	(user-selected)	*assumption_info*	Downward
O-T	Private, import mediating	(user-selected)	*target_info*	Downward
E-O	Private, export mediating	*epistemic_info*	(user-selected)	Upward
E-A	Private	*epistemic_info*	*assumption_info*	None
E-T	Private	*epistemic_info*	*target_info*	None
R-O	Private, export mediating	*request_info*	(user-selected)	Upward
R-A	Private	*request_info*	*assumption_info*	None
R-T	Private	*request_info*	*target_info*	None

Table 9.2: reflection types.

The DESIRE framework does not support a notion of state for information links available for users. Moreover, the user cannot specify if and how the domain records the result of information transmission or enabling conditions for receipt of information to be possible. As a consequence, information link mapping descriptions for a DESIRE link only need to contain two basic information elements (instead of eight): a basic information element that describes the state of the domain in which information is to be transmitted, and a basic information element that describes the state of the co-domain in which the information is received. (These two basic information elements correspond with the first and the eighth basic information element in an information link mapping description as defined by Definition 9.32.)

In DESIRE, the state of components is described by ground atoms that act as proposition symbols of a propositional language. According to the definition, an

information link mapping description for a DESIRE link is a set of tuples of basic information elements, where each basic information element is a ground atom together with a truth value. In the DESIRE framework, the user does not have to directly enumerate such sets of tuples of ground atoms. (Such sets may have a large number of elements.) Instead, the user may describe the set by giving a sufficient number of constraints on all possible information link mapping descriptions. The constraints should specify a unique information link mapping. For both basic information elements in a DESIRE information link mapping description, one constraint is always present: for both the domain and co-domain of the link, a specific signature in the domain interface and co-domain interface, respectively, is given. (Table 9.1 lists which signatures are given for a specific reflection type.) Basic information elements in the information link mapping description are at least constrained to ground atoms of these signatures.

To present the facilities offered by the DESIRE framework for describing additional constraints, an example of the DESIRE description of a link of type O-A is given:

```
sort links
        (OA,IA)
        rest identity
object links identity
term links identity
atom links
        (known(X: OA),assumption(X: IA, pos)): <<false, true>>;
```

This description is best read from bottom to top. The last line states that all basic information elements in the description are of the form *⟨[*known*(…):0];[*assumption*(…,*pos*):1]⟩*. The other lines specify what terms may appear at the positions indicated by the dots. The second line specifies the constraint that for terms of the sort OA, corresponding terms of the sort IA have to be found. The constraints specified are then applied recursively to the structure of terms in the sorts OA and IA to determine which terms of the sort OA correspond to which terms of the sort IA. Assume that OA and IA are defined as $meta_{OA}(belief\_info)$ and $meta_{IA}(belief\_info)$, respectively. (The signature belief_info is defined in Example 10.5.) Thus, the term belief*(M:* MATCH*)* is a term of both OA and IA. The outermost symbol is the symbol belief, which is a function symbol. Constraints for function symbols are listed in the line starting with **term links**. The expression **identity** specifies that (only) lexically equivalent function symbols (i.e., equal strings) in the sorts OA and IA match. This is the case for the terms belief*(M:* MATCH*)* in the sorts OA and IA, as these terms all start with the same symbol, belief. The process then continues. The argument of belief is a term of the sort MATCH. According to the expression **rest identity** on the third line, all matching sort names except the ones listed on the second line have to be lexically equivalent. This is the case for the name MATCH, after which the process continues with the function and object symbols in the sort MATCH, and so on. The information link mapping description specified by the

constraints given above is the set $\{\langle[\text{known}(\text{belief}(\text{match}(t,q))):0];[\text{assumption}(\text{belief}(\text{match}(t,q)),\text{pos}):1]\rangle \mid t \in OT_2$ and $q \in Q\}$. As an aside, this description is an example of how a *closed world assumption* is represented in DESIRE: if the truth of atoms $\text{belief}(\text{match}(t,q))$ is not known at the domain side of the link, then at the co-domain side it is assumed that such atoms are positive (pos), i.e., they are true.

The process outlined above only establishes an information link mapping description for two of the eight basic information elements in an information link mapping description. A pair of basic information elements $\langle[p:s];[q,s']\rangle$ as defined by a user denotes the following complete information link mapping description:

$\{\langle\langle[p:s];[\mathbf{T}:1]\rangle;\langle[\text{link\_state}(\text{uptodate}):1];[\mathbf{T}:1];[\mathbf{T}:1];[\text{link\_state}(\text{idle}):1]\rangle;\langle[\mathbf{T}:1];[q,s']\rangle\rangle,$.
$\langle\langle[p:s];[\mathbf{T}:1]\rangle;\langle[\text{link\_state}(\text{awake}):1];[\mathbf{T}:1];[\mathbf{T}:1];[\mathbf{T}:1]\rangle;\langle[\mathbf{T}:1];[q,s']\rangle\rangle\}$.

Thus, there is no constraint on the second and seventh states and most of the link states. The constraints on the third state indicate that information is only transmitted if a link is in state uptodate or awake. After information transmission has finished, a link that was in state uptodate at the beginning of the transmission becomes idle. This is not the case for links that are awake.

9.2.3 The Relation between Process Composition and Knowledge Composition

Section 9.2.1 and Section 9.2.2 presented knowledge structures and process composition, respectively. In the current section, the relations between knowledge structures and process composition mentioned in Section 9.2.1 and Section 9.2.2 are summarised.

- DESIRE knowledge structures are defined almost independently of process composition. However, some of the standard information types given in Section 9.2.1.4 represent knowledge of process composition. This is the case for the standard meta information type *target_info* and for the task control information type *TCparts*.
- For the individual components that represent processes in a process composition, knowledge structures are specified for the input, output and internal state of components. Moreover, for each component a knowledge base is given that relates the input, output and internal state of a component. Functions $\Sigma\Sigma_{in}(C,SH)$, $\Sigma\Sigma_{int}(C,SH)$, $\Sigma\Sigma_{out}(C,SH)$, and $KB(C,SH)$, specify which knowledge structures are used by which components in a structure hierarchy *SH*.

9.3 DESIRE dynamics

The goal of this chapter is to describe the dynamics of multi-agent systems modelled in DESIRE using the semantic structure developed in this thesis. In

conformance with the basic assumption adapted in this thesis (presented in Chapter 1), a multi-agent system modelled using DESIRE is represented as a compositional system. The previous sections showed how the (static) compositional structure of a DESIRE model of a multi-agent system is represented using the semantic structure. The three views on the behaviour of a compositional system defined in Chapter 5 are used to describe the dynamics of such a compositional system. As explained in Chapter 5, these views are relative to:

- A composition structure (in the case of the white box view) or a structure hierarchy (in the case of the black box view or the glass box view). A structure hierarchy with control (as defined in Definition 9.14) can be used to describe the dynamics of the corresponding DESIRE model;
- A family of sets of local component and link traces $Beh_{loc}(S)$ for specific components and links S is the DESIRE model. (As indicated in Chapter 5, the specific components and links for which sets of local component and link traces have to be specified, differ for the three views on the behaviour of a compositional system);
- A collection of compatibility relations.

Thus, to describe the dynamics of DESIRE models using the three views defined in Chapter 5, it suffices to define sets of local behaviour and collections of compatibility relations for these models. Sets of local component and link traces for DESIRE components are defined in Section 9.3.1. Collections of compatibility relations for DESIRE are defined in Section 9.3.2.

9.3.1 Local Component and Link Traces

This section uses temporal logic as a means to describe sets of local component and link traces. In Section 9.3.1.1, a temporal logic is defined for this purpose as an extension of the (non-temporal) logic language used for the description of individual component states (Definition 9.17) and for link states (Definition 9.29). Local component *traces* of a component C consist of local component *states*. Definition 9.25, presented in Section 9.2.2.2, defines signatures that describe the sets of local components states that constitute local component traces. The standard information types *TCinput* and *TCoutput* (presented in Section 9.2.1.4) are part of the state of all components in a DESIRE model. Moreover, the state of control components in a DESIRE structure hierarchy with control is fully described by these information types. Temporal logic formulae based on these information types describe behaviour of DESIRE components in terms of the predicates, sorts, objects and functions defined by *TCinput* and *TCoutput*. In Section 9.3.1.2, the intended use of these information types is described by presenting an informal description of the dynamics of a DESIRE model. The stage is then set to define local behaviour of DESIRE components. Section 9.3.1.3 defines, for each component C in a DESIRE structure hierarchy with control, a set of formulae that specify the

local behaviour of the component. The set of local component traces $Beh_{loc}(C)$ for C is defined as the set of local component traces that satisfy the set of formulae. Section 9.3.1.4 defines a set of local link traces for each link in a DESIRE structure hierarchy with control in a similar way.

9.3.1.1 Specification of Local Component and Link Traces

In this section, the specific temporal language used to describe sets of local component and link traces is defined. Section 9.2.2.2 presented a (non-temporal) logic language for the description of individual component states (Definition 9.17) and for link states (Definition 9.29). The (temporal) language for description of local component and link traces is defined as an extension of these languages To describe the semantics of DESIRE, discrete time frames are used. This enables the use of 'previous' and 'next' modalities in the temporal language. As described in Section 9.3.1.3, 'previous' and 'next' modalities are important in the formal description on the behaviour of control components. The temporal language is formally defined as follows:

Definition 9.36. (Local behaviour specification language). *Let S be a component or link and let $For(\Sigma_S)$ be the set of information state description formulae for S (Definition 9.17). The set of* local behaviour specification formulae *$Spec(\Sigma_S)$ is the smallest set closed under the following restrictions:*

- *If S is a component and $\varphi \in For(\Sigma_S)$ then $\boldsymbol{C_{S,in}}\varphi, \boldsymbol{C_{S,int}}\varphi, \boldsymbol{C_{S,out}}\varphi \in Spec(\Sigma_S)$;*
- *If S is a link and $\varphi \in For(\Sigma_S)$ then $\boldsymbol{C_S}\varphi \in Spec(\Sigma_S)$;*
- *If $\varphi, \psi \in Spec(\Sigma_S)$ then $\neg\varphi, \varphi \wedge \psi \in Spec(\Sigma_S)$;*
- *If $\varphi \in Spec(\Sigma_S)$ then $\boldsymbol{P}\varphi \in Spec(\Sigma_S)$, $\boldsymbol{X}\varphi \in Spec(\Sigma_S)$, and $\boldsymbol{F}\varphi \in Spec(\Sigma_S)$.*

The intended meaning of the connectives is as follows. A formula $\boldsymbol{C_{S,in}}\varphi$ is true at a time point t iff the formula φ is true at time point t for the input state of component S, and is to be read as 'currently φ'. The intended meaning of $\boldsymbol{C_{S,int}}\varphi$, $\boldsymbol{C_{S,out}}\varphi$ and $\boldsymbol{C_S}\varphi$ is similar. A formula $\boldsymbol{C_S}\varphi$ is true at a time point t iff the formula φ is true at time point t for the state of link S. A formula $\boldsymbol{P}\varphi$ is true at a time point t iff the formula φ is true at the previous point in time. A formula $\boldsymbol{X}\varphi$ is true at a time point t iff the formula φ is true at the next time points. A formula $\boldsymbol{F}\varphi$ is true at a time point t iff a time point $t' > t$ exists such that formula φ is true at time point t'. The connectives $\vee$, $\bot$, $\top$ and $\rightarrow$ are defined as usual.

Definition 9.37. (Non-temporal subformulae). *Let $\varphi \in Spec(\Sigma_S)$ be a local specification formula. A subformula φ' of φ is called* non-temporal *iff φ' nor the subformulae of φ' contain **B** or **W** connectives.*

Sets of local component and link traces for DESIRE are described by *local behaviour specifications*, which are specific sets of local component specification formulae

from the set $Spec(\Sigma_S)$. These sets are defined in Section 9.3.1.3. An example of a local behaviour specification formula is presented below.

Example 9.38. As an example of such temporal formulae, consider the following formula, which formalises the requirement given in Chapter 4: once a broker agent receives a query from a user, information matching the query has to be communicated to the user at the next moment in time if this information was already available to the broker, or some time in the future in all other cases. (In this formula, $q \in Q$, $t \in T_2$, $u \in Users$ and $p \in Providers$ as in Example 5.2):

$$(\text{communicated\_by}(q,u) \wedge \text{belief}(\text{match}(t,q))) \rightarrow$$
$$((\mathbf{P}\text{communicated\_by}(t,p) \rightarrow \mathbf{X}\text{to\_be\_communicated\_to}(t,u)) \vee$$
$$(\mathbf{F}(\text{communicated\_by}(t,p) \rightarrow \mathbf{F}\text{to\_be\_communicated\_to}(t,u))))$$ ■

The local behaviour specification language of a component or link S is interpreted relative to a local trace LT_S of S as follows:

Definition 9.39. (Local satisfaction and local behaviour). *Let $LT_S=\langle\langle T_S;<_S\rangle;V_S\rangle$ be a local component or link trace, let $\models_3^+,\models_3^- \subseteq \mathcal{S}_S \times For(\Sigma_S)$ be the satisfaction relations for formulae $\varphi \in For(\Sigma_S)$ defined in Section 9.2.2.2 and let $\varphi \in Spec(\Sigma_S)$ be a local specification language formula. Then* local satisfaction *of φ by LT_S at state $v_{S,t}$, state at point $t \in T_S$, denoted $LT_S, v_{S,t} \models_l \varphi$, is defined by induction on the structure of φ as follows:*

- $LT_S, v_{S,t} \models_l \mathbf{C}_{S,X}\phi \quad \Leftrightarrow \quad (v_{S,t})_X \models_3^+ \phi$ *for $X \in \{in,int,out\}$, if S is a component;*
- $LT_S, v_{S,t} \models_l \mathbf{C}_S\phi \quad \Leftrightarrow \quad v_{S,t} \models_3^+ \phi$, *if S is a link;*
- $LT_S, v_{S,t} \models_l \neg\varphi \quad \Leftrightarrow \quad LT_S, v_{S,t} \not\models_l \varphi$;
- $LT_S, v_{S,t} \models_l \varphi \wedge \psi \quad \Leftrightarrow \quad LT_S, t \models_l \varphi$ *and* $LT_S, t \models_l \psi$.
- $LT_S, v_{S,t} \models_l \mathbf{P}\varphi \quad \Leftrightarrow \quad LT_S, prev_{LT_S}(v_{S,t}) \models_l \varphi$;
- $LT_S, v_{S,t} \models_l \mathbf{X}\varphi \quad \Leftrightarrow \quad LT_S, next_{LT_S}(v_{S,t}) \models_l \varphi$;
- $LT_S, v_{S,t} \models_l \mathbf{F}\varphi \quad \Leftrightarrow \quad$ *there is a t' with $t'>t$ such that* $LT_S, v_{S,t'} \models_l \varphi$.

A local component trace satisfies a formula φ, denoted $LT_S \models_l \varphi$, if for all t, $LT_S, v_{S,t} \models_l \varphi$. A local component trace satisfies a set of formulae Spec if for all $\varphi \in Spec$, $LT_S \models_l \varphi$.

The notion of satisfaction is used to define sets of local component and link traces for DESIRE as follows:

Definition 9.40. (DESIRE local behaviour). *Let S be a component or link and let Spec(S) be the local behaviour specification of S, (i.e., the set of local behaviour specification formulae defined for S in Section 9.3.1.3 below). The set of DESIRE local behaviour traces is the set $Beh_{loc}(S)=\{LT \in \mathcal{LT}_S \mid LT \models_l \varphi$ for all $\varphi \in Spec(S)\}$.*

9.3.1.2 Control Knowledge in DESIRE

Control in DESIRE is centred around the control lexicon consisting of the information types *TCinput* and *TCoutput* presented in Section 9.2.1.4. These information types are used for both types of components in a DESIRE model. A knowledge base containing knowledge rules for the information types *TCinput* and *TCoutput* is associated with each composed component. Thus, these knowledge rules express control knowledge completely in terms of *TCinput* and *TCoutput*. A knowledge base for (only) user-supplied information types is associated with each primitive component. The information types *TCinput* and *TCoutput* are part of the full signatures $\Sigma_{C,out}$ and $\Sigma_{C,in}$ (see Definition 9.23) that describe the state of primitive components and are used to control how the knowledge base rules associated with a primitive component are used. This section presents an informal description of the behaviour of DESIRE components and links in terms of the predicates, sorts, objects and functions defined by *TCinput* and *TCoutput* to provide insight in their intended meaning.

Each component in a DESIRE model represents a process in a multi-agent system. The activity of such a process results in state changes of the component. As the state of a DESIRE process is described by a propositional language (explained in Section 9.2.2.2), the truth values of propositions that describe the state of a component change as well. Truth values of propositions that describe the state of a component thus form a means to describe and evaluate the activity of a process in DESIRE. A description in terms of truth values is generic for DESIRE: each DESIRE model is committed to the use of propositional logic to describe the state of a component.

A DESIRE model consists of primitive and composed components. There is one composed component, the toplevel component, that is not a subcomponent of any other component. A knowledge base for the information types *TCinput* and *TCoutput* is associated with each composed component. As an example, consider the following rule that may be specified for the toplevel component:

if **start**
then next_component_state(user_1, active)
and next_task_control_focus(user_1, default_focus)
and next_extent(user_1, all_p);

The process represented by the toplevel component is the first component that is active in a DESIRE model. It starts by evaluating its own current state. In this state, the formula **start** is true. Therefore, the toplevel component applies the knowledge rule given above and derives next_component_state(user_1,active), next_task_control_focus(user_1,default_focus), and next_extent(user_1,all_p) which are atoms of *TCoutput*.

The atom next_component_state(user_1,active) states that the next *activation type* of component user_1 is active. At each moment in time, a component is either idle, active, or awake. As indicated by the definitions of *TCinput* and *TCoutput*, a fourth

activation type is distinguished: busy. The activation type busy indicates whether the component is actually performing anything. If the activation type of a component is idle, it is not performing anything. To indicate this, in this case, the activation type busy is not true for the component. Activation type active means that the component is busy to apply its knowledge base in a way explained below. It will do so *once*. Activation type awake means that the component, whether it is currently busy or not, will become busy whenever its domain substate changes. In this case, it is guaranteed to apply its knowledge base *at least once*. By default, all components and links are idle. The toplevel component automatically becomes active at the first moment in time. Note that the predicate symbols in the conclusion of the example control rule begin with next. The component behaviour described by the activation type is realised at the next moment in time.

A downward control link to component user_1 links the atoms derived by the toplevel component to atoms own_component_state(active), own_task_control_focus(default_focus), and own_extent(all_p), which are atoms of *TCinput*. (*TCinput* is part of the full information type $\Sigma_{C,in}$ (see Definition 9.23) that describes the input substate of the primitive component user_1.) As a result, component user_1 becomes active.

Component user_1 is a primitive component and has an associated knowledge base for user-supplied information types. In the running example, a knowledge rule for this component is, for example:

if **not** communicated_by(t, broker)
then to_be_communicated_to(q, broker);

As soon as component user_1 becomes active, it starts to derive conclusions from information present in its input interface by forward chaining. The chaining process is controlled by *task control foci* and *extents*, which are described using several predicates defined in *TCinput* and *TCoutput*. A task control focus is a set of pairs consisting of an output atom and a *target type*. Each task control focus has a name, for instance, default_focus in the example control rule presented in this section. A target type is either confirm, reject or determine. The contents of a task control focus is specified by the standard information type *target_info* presented in Table 9.1. Atoms of this information type are transmitted via information links or specified as *initial meta-facts*.

For component user_1, the initial meta-fact target(default_focus,to_be_communicated_to(q,broker), confirm) specifies that the pair ⟨to_be_communicated_to(q,broker);confirm⟩ is an element of task control focus default_focus. As a result of information transmission by the downward control link to user_1, default_focus is the current task control focus of user_1. The chaining process is executed with the goal to confirm to_be_communicated_to(q,broker), i.e., to derive that to_be_communicated_to(q,broker) is true. (If the target type of to_be_communicated_to(q,broker) were reject, the goal would have been to derive that to_be_communicated_to(q,broker) is false. If the target type of

to_be_communicated_to(q,broker) were determine, the goal would have been to determine which of the two truth values for to_be_communicated_to(q,broker).)

As a result of information transmission by the downward control link to user_1, the current extent of user_1 is all_p. The extent controls the effort spent by the chaining process to derive truth values for the atoms in the current task control focus as follows:

- Extent any: the process stops after a truth value is found for at most one atom in the current task control focus;
- Extent any_new: the process stops after a truth value is found for at most one atom in the current task control focus that has not been derived before;
- Extent all_p: the process stops after a truth value is found for as many atoms in the current task control focus as possible;
- Extent every: the process stops after a truth value is derived that makes it impossible to find a truth value as specified by the current task control focus. I.e., if the current task control focus is to confirm truth of an atom a*(T)*, and for some term *t*, truth value false is derived for a*(t)*, then the process stops in the case of extent every.

After the chaining process stops, the result of the process is evaluated. For this purpose, *evaluation criteria* are defined. An evaluation criterion is a set of pairs consisting of an output atom and a target type, similar to a task control focus. Each evaluation criterion has a name. Suppose that for user_1, an evaluation criterion default_ec is defined, which contains the pair ⟨to_be_communicated_to(q,broker);confirm⟩. The *termination type* for user_1, which is either succeeded or failed, with respect to default_ec, is now determined as follows:

- If the current extent is any, the termination type is succeeded if for at least one atom in default_ec, truth value true (because the target type is confirm) has been derived;
- If the current extent is any_new, the termination type is succeeded if for at least one atom in default_ec, truth value true (because the target type is confirm) has been derived, and for this atom, the truth value was not known before the chaining process started;
- If the current extent is every, the termination type is succeeded if for all atoms in default_ec, truth value true (because the target type is confirm) has been derived;
- If the current extent is all_p, the termination type is always succeeded: for as many atoms as possible in default_ec, truth value true (because the target type is confirm) need to be derived.
- For each extent, the termination type is failed if the above condition is not fulfilled, while an exhaustive effort to fulfil the condition has been

performed. E.g., if the extent is any, and for every atom in the current task control focus, derivation of the truth value given by the task control focus did not produce the truth value requested, then the termination type is failed. As a consequence, in the case of extent type all_p, the termination type cannot be failed, as it is always the case that as many truth values are derived 'as possible'.

The control part of the output substate contains the result of the evaluation, described by, for instance, the atom own_evaluation(default_ec,all_p,succeeded). This information is transmitted by an upward control link to the task control knowledge base of the toplevel component (as user_1 is a subcomponent of the toplevel component). The toplevel component may contain the following example rule:

if evaluation(user_1,default_ec,all_p,succeeded)
and not previous_evaluation(user_1,default_ec,all_p,succeeded)
then next_component_state(broker, active)
and next_link_state(user_1_to_broker, uptodate);

The conjunctive premise of this rule evaluation(user_1,default_ec,all_p,succeeded) **and not** previous_evaluation(user_1,default_ec,all_p,succeeded) is true at the moment at which the evaluation of user_1 becomes succeeded. Thus, at that moment, the toplevel component derives next_component_state(broker,active) and next_link_state(user_1_to_broker, uptodate). These activation types are transmitted to broker and user_1_to_broker, respectively, by downward control links. The activation type of a link is either idle, awake, or uptodate. Activation type uptodate means that information transmission as specified by the information link mapping description has been carried out (once). Activation type awake means that the link is continuously able to carry out information transmission as specified by the information link mapping description, as soon as the state of the domain component changes.

In this scenario, link user_1_to_broker links the newly derived atom to_be_communicated_to(q,broker) to input atom communicated_by(q,user_1) of component broker. This component broker is in state active, which means that it will become busy (once) as soon as the truth value of input atoms change (reflecting a new input substate). As link user_1_to_broker is in state uptodate, it updates the input substate of its co-domain (component broker) according to its information link description, possibly resulting in a state change of the input substate of broker. Thus, broker becomes busy. However, as broker is a composed component (as opposed to user_1), there is no knowledge base for which a chaining process has to be executed. Instead, the task control knowledge base associated with broker is applied in the same way as the task control knowledge base of the toplevel component. In other words, the general DESIRE dynamics described informally in this section is recursively applicable.

9.3.1.3 Local Behaviour

As stated at the beginning of Section 9.3.1, sets of local component and link traces have to be defined to describe the dynamics of DESIRE models. Such sets are defined as those sets of local component or link traces that satisfy a local behaviour specification (Definition 9.40). Consequently, to describe the dynamics of DESIRE models, local behaviour specifications have to be defined for all components. This is the topic of this section.

In the specification of the behaviour of a DESIRE model, *three* different kinds of components have to be considered, for the following reason. As stated before, a DESIRE model contains *two* kinds of components: primitive and composed. A DESIRE model is described by a DESIRE structure hierarchy *SH*. A knowledge base for $(\Sigma\Sigma_{in}(C,SH))_{\perp}$, $(\Sigma\Sigma_{int}(C,SH))_{\perp}$ and $(\Sigma\Sigma_{out}(C,SH))_{\perp}$, where $\perp$ is the bottom element of the set *LEV(C,SH)* of level identifiers, is associated with each primitive component *C*. (However, a primitive component in DESIRE may be described in some other way, e.g. in a specification language such as Z or in a programming language. In such cases, there is no knowledge base associated with the primitive component.) A knowledge base for *TCinput*, $\langle(\emptyset;\emptyset);\emptyset;\emptyset;\emptyset\rangle$, and *TCoutput* is associated with each composed component *C*. A DESIRE model is represented in the semantic structure by a DESIRE structure hierarchy *with control SH'*, which introduces a third kind of component: control components. In a structure hierarchy with control, there is a control component C_{ctr} for each composed component *C*. In a structure hierarchy with control that represents a DESIRE model, no knowledge base is associated with composed components: control knowledge bases associated with composed components in a DESIRE model are moved to the corresponding control component in a structure hierarchy with control, i.e. $KB(C_{ctr},SH')=KB(C,SH)$ for *C* a composed component.

Primitive and control components share some common characteristics: neither can have subcomponents, and knowledge bases may be associated with both. (Moreover, primitive and control components are the only kinds of components in a DESIRE structure hierarchy with control with which knowledge bases can be associated.) Some differences can also be distinguished. For primitive components, initial meta-facts, task control foci, extents, and kernel input and output information types are defined by the user and are therefore in general not the same for all models. For control components, initial meta-facts, task control foci, extents, as well as the kernel input and output information types are pre-defined and the same for all DESIRE models. These pre-defined initial meta-facts, task control foci, and extents are defined as follows:

- Each control component has one task control focus, called default_focus. This task control focus is always the current task control focus. It contains all possible output atoms of the control component, i.e., target(default_focus,A_i,determine) is always true for each output atom A_i;

- The extent of a control component is always all_p. Therefore, the evaluation is always succeeded for any possible evaluation criterion.
- The kernel part of the input and output substate of a control component consists of *gratoms(TCinput)* and *gratoms(TCoutput)*, respectively.

Notwithstanding the differences between composed on the one hand, and primitive and control components on the other hand, important similarities can also be observed. These similarities consist of the use of delegation to perform tasks and the use of control signatures:

- A composed component delegates the determination of truth values for its kernel output atoms to its subcomponents;
- A primitive component or control component delegates the determination of truth values for its kernel output atoms to a reasoning engine, the behaviour of which is specified either by a knowledge base or by some alternative means;
- All three kinds of components use the same input and output control signatures, which influence the delegation of functions to subcomponents or a reasoning engine and describe the result of the delegation.

These similarities are employed to specify the behaviour of DESIRE components as follows:

- As stated in Section 9.2.2.2, the input and output substates of a DESIRE component (either primitive or composed) consists of two parts: a kernel part and a control part. The kernel part itself consists of levels, and each level consists of a user-defined part and a part described by the standard meta-signatures. (See Figure 9.6). The user-defined parts of the output substates in a local component trace for a composed component are unconstrained from a local point of view. In other words, all possible sequences of user-defined output substates may occur in $Beh_{loc}(C)$. Consequently, no specification for this part is given. However, from a more global point of view, the kernel part of output substates is determined by the subcomponents via export mediating links. Thus, determination of this part is delegated to the subcomponents;
- The part described by the standard meta-signatures of the output substate is completely determined locally, as it contains information lifted from user-defined parts. The relation between the user-defined parts and the lifted standard information is static: it can be fully defined per state. It is not necessary to consider states in the context of traces in which they occur. The relation between user-defined parts and the lifted standard information is described in (Brazier, Treur, Wijngaards & Willems, 1999). As an aside, please take note that on the input side, the state of the user-defined part is

partly determined by the truth values of assumption(*A*,*S*) atoms (where *A* is an atom used to describe a user-defined part, and *S* is either pos or neg). However, these truth values are themselves unconstrained from a local point of view, similar to the user-defined parts.

- The kernel part of the output substates in a local component trace for a primitive component or control component is determined by a reasoning engine. As many reasoning engines are conceivable, no specification of this part is given. (For primitive components and control components with which a knowledge base is associated, a specification is sketched in Section 9.3.5.) Thus, determination of this part is delegated to a reasoning engine that is assumed to be available. In other words, a primitive component or control component is viewed as if it were a composed component with one subcomponent that contains the reasoning engine;
- A general specification (set of formulae) is developed to describe the control output part of local component traces for all three kinds of components. This specification describes how the control part of the output substates makes information on the results of the delegation available to other components in a standard way;
- The input substates in local component traces (both the control part and the kernel part) are unconstrained from a local point of view for all three kinds of components. (The input substates are determined by information transmission.) Therefore, no specification is needed for the input substates.

To summarise, the specification of the standard behaviour of DESIRE components only needs to describe the control part of the output substate of a component. The specification describes the result of the delegation to either subcomponents or a reasoning engine in relation to control information received as input. This control information is represented by the signature $\Sigma_{C,in}^{tc}$ defined in Section 9.2.2.2 (Definition 9.21), which is repeated here for ease of reference:

Definition 9.21. (Standard task control input signature). *Let* $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ *be a DESIRE structure hierarchy and let* $C\in Comp$ *be a component. The* standard task control input signature *for C is the signature* $\Sigma_{C,in}^{tc}=TCpredefinedSorts\oplus TCparts\oplus\langle S;\varnothing\rangle;Obj;\varnothing;Pred\rangle$, *where TCpredefinedSorts and TCparts are the standard signatures presented in Section 9.2.1.4 and:*

- $S=\{$Link,Component,ExtentType,ActivationType,OwnTaskControlFocus$\}$;
- *Obj an S-indexed family of sets with the following members:*
  - $Obj_{\langle \text{Link}\rangle} = \{L\in Lnk \mid L\prec C\}$;
  - $Obj_{\langle \text{Component}\rangle} = \{C'\in Comp \mid C'\prec C\}$;
- *Pred an* S^**-indexed family of sets with the following members:*
  - $Pred_{\langle\rangle} = \{$start$\}$;

- $Pred_{\langle \text{ActivationType} \rangle}$ = {own_component_state,};
- $Pred_{\langle \text{OwnTaskControlFocus} \rangle}$ = {own_task_control_focus};
- $Pred_{\langle \text{ExtentType} \rangle}$ = {own_extent};

The control part of the output substate is described by the signature $\Sigma_{C,out}{}^{tc}$ defined in Section 9.2.2.2 (Definition 9.22), which is also repeated here for ease of reference:

Definition 9.22. (Standard task control output signature). *Let* $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ *be a DESIRE structure hierarchy and let C be a component. The* standard task control output signature *for C is the signature* $\Sigma_{C,out}{}^{tc}= TCpredefinedSorts \oplus TCparts \oplus \langle\langle S;\varnothing\rangle;Obj;\varnothing;Pred\rangle$, *where TCpredefinedSorts and TCparts are the standard signature presented in Section 9.2.1.4 and:*

- $S=\{$Link,Component,OwnEvaluationCriterium,ExtentType,TermType,ActivationType, OwnTaskControlFocus$\}$;
- *Obj an S-indexed family of sets with the following members:*
  - $Obj_{\langle \text{Link} \rangle} = \{L \in Lnk \mid L \prec C\}$;
  - $Obj_{\langle \text{Component} \rangle} = \{C' \in Comp \mid C' \prec C\}$;
- *Pred an* S^**-indexed family of sets with the following members:*
  - $Pred_{\langle \text{OwnEvaluationCriterium;ExtentType;TermType} \rangle}$ = {own_evaluation};
  - $Pred_{\langle \text{ActivationType} \rangle}$ = {own_component_state,};
  - $Pred_{\langle \text{OwnTaskControlFocus} \rangle}$ = {own_task_control_focus};
  - $Pred_{\langle \text{ExtentType} \rangle}$ = {own_extent};
  - $Pred_{\langle\rangle}$ = {stop}.

The stage is now almost set to present the specification of the standard behaviour of DESIRE components. However, first the temporal language presented in Section 9.3.1.1 that will be used to specify the behaviour of a composed component has to be precisely specified itself. This language is defined as an extension of the (non-temporal) *propositional* languages defined in Definition 9.17 and Definition 9.29. Consequently, sets of proposition symbols are needed for such specifications. The standard task control input and outpus signatures are likely candidates, as they consist of terms suited for the description of behaviour. (They constitute a *control lexicon* in the terms of Chapter 8.) However, $\Sigma_{C,out}{}^{tc}$ and $\Sigma_{C,in}{}^{tc}$ are first-order signatures, as opposed to sets of proposition symbols. Thus, they cannot be used directly. Definition 9.17 and Definition 9.29 do not require the use of specific sets of propositional symbols for the atomic formulae. Thus, to describe the behaviour of DESIRE components, any set of propositional symbols can be used. Therefore, sets of ground atoms determined by $\Sigma_{C,out}{}^{tc}$ and $\Sigma_{C,in}{}^{tc}$ can be used. In this way, the relatively complex formulae are more readable for the human reader, while use of first-order temporal logic is avoided. In other words, the order-sorted first-order signatures used to describe the state of DESIRE components are also used to provide structure for formulae that describe the

behaviour of DESIRE. (Formulae that describe the behaviour of DESIRE can be considered to be meta-level formulae with respect to DESIRE itself.)

According to Definition 9.40, the set $Beh_{loc}(C)$ for a component C is defined as the set of local component traces that satisfy the set of formulae defined as follows:

Definition 9.41. (DESIRE local behaviour specification). *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a DESIRE structure hierarchy with control and let $C \in Comp$ be a component. Let:*

- *E range over the set EC(C,SH) of evaluation criteria of C;*
- *A_i range over the set of terms generated by the signature $\bigoplus_{l \in LEV(C,SH)} (meta_{\langle OA,l'\rangle}(\Sigma_{C,out,l}^{ker}))$, where l' is the successor of l;*

The DESIRE local behaviour specification for components *is a specification Spec(C) with the following formulae:*

- *If C is a composed component:* $\text{finished} \leftrightarrow \bigwedge_{i=1}^{n} \text{stopped}(C_i)$ *for $C_i \in Subc(C)$;*
- *For each evaluation criterion E in EC(C,SH), the specification of C contains the following formulae:*
  - $(\mathbf{PC}(\text{own\_extent}(\text{any}) \wedge \text{own\_task\_control\_focus}(E)) \wedge$
    $\bigvee_{i=1}^{n} (\mathbf{C}(\text{target}(E,A_i,\text{confirm}) \wedge$
    $\text{true}(A_i))))$
    $\rightarrow \mathbf{C}(\ \text{own\_evaluation}(E,\text{any},\text{succeeded}) \wedge$
    $\neg\text{own\_evaluation}(E,\text{any},\text{failed}));$
  - $(\mathbf{PC}(\text{own\_extent}(\text{any}) \wedge \text{own\_task\_control\_focus}(E)) \wedge$
    $\bigvee_{i=1}^{n} (\mathbf{C}(\text{target}(E,A_i,\text{reject}) \wedge$
    $\text{false}(A_i))))$
    $\rightarrow \mathbf{C}(\ \text{own\_evaluation}(E,\text{any},\text{succeeded}) \wedge$
    $\neg\text{own\_evaluation}(E,\text{any},\text{failed}));$
  - $(\mathbf{PC}(\text{own\_extent}(\text{any}) \wedge \text{own\_task\_control\_focus}(E)) \wedge$
    $\bigvee_{i=1}^{n} (\mathbf{C}(\text{target}(E,A_i,\text{determine}) \wedge$
    $\text{known}(A_i))))$
    $\rightarrow \mathbf{C}(\ \text{own\_evaluation}(E,\text{any},\text{succeeded}) \wedge$
    $\neg\text{own\_evaluation}(E,\text{any},\text{failed}));$
  - $(\mathbf{PC}(\text{own\_extent}(\text{any}) \wedge \text{finished} \wedge \text{own\_task\_control\_focus}(E)) \wedge$
    $\bigwedge_{i=1}^{n} (\mathbf{C}(\text{target}(E,A_i,\text{confirm}) \wedge$
    $\neg\text{true}(A_i))))$
    $\rightarrow \mathbf{C}(\ \text{own\_evaluation}(E,\text{any},\text{failed}) \wedge$
    $\neg\text{own\_evaluation}(E,\text{any},\text{succeeded}));$

– $(\mathbf{PC}(\text{own\_extent}(\text{any}) \wedge \text{finished} \wedge \text{own\_task\_control\_focus}(E)) \wedge$
$\bigwedge_{i=1}^{n}(\mathbf{C}(\text{target}(E,A_i,\text{reject}) \wedge$
$\neg\text{false}(A_i))))$
$\rightarrow \mathbf{C}(\ \text{own\_evaluation}(E,\text{any},\text{failed}) \wedge$
$\neg\text{own\_evaluation}(E,\text{any},\text{succeeded}));$

– $(\mathbf{PC}(\text{own\_extent}(\text{any}) \wedge \text{finished} \wedge \text{own\_task\_control\_focus}(E)) \wedge$
$\bigwedge_{i=1}^{n}(\mathbf{C}(\text{target}(E,A_i,\text{determine}) \wedge$
$\neg\text{true}(A_i) \wedge \neg\text{false}(A_i))))$
$\rightarrow \mathbf{C}(\ \text{own\_evaluation}(E,\text{any},\text{failed}) \wedge$
$\neg\text{own\_evaluation}(E,\text{any},\text{succeeded}));$

– $(\mathbf{PC}(\text{own\_extent}(\text{any\_new}) \wedge \text{own\_task\_control\_focus}(E)) \wedge$
$\bigvee_{i=1}^{n}(\mathbf{C}(\text{target}(E,A_i,\text{confirm}) \wedge$
$\text{true}(A_i)) \wedge \neg\mathbf{PC}(\text{true}(A_i)))$
$\rightarrow \mathbf{C}(\ \text{own\_evaluation}(E,\text{any\_new},\text{succeeded}) \wedge$
$\neg\text{own\_evaluation}(E,\text{any\_new},\text{failed}));$

– $(\mathbf{PC}(\text{own\_extent}(\text{any\_new}) \wedge \text{own\_task\_control\_focus}(E)) \wedge$
$\bigvee_{i=1}^{n}(\mathbf{C}(\text{target}(E,A_i,\text{reject}) \wedge$
$\text{false}(A_i)) \wedge \neg\mathbf{PC}(\text{false}(A_i)))$
$\rightarrow \mathbf{C}(\ \text{own\_evaluation}(E,\text{any\_new},\text{succeeded}) \wedge$
$\neg\text{own\_evaluation}(E,\text{any\_new},\text{failed}));$

– $(\mathbf{PC}(\text{own\_extent}(\text{any\_new}) \wedge \text{own\_task\_control\_focus}(E)) \wedge$
$\bigvee_{i=1}^{n}(\mathbf{C}(\text{target}(E,A_i,\text{determine}) \wedge$
$\text{known}(A_i)) \wedge \neg\mathbf{PC}(\text{known}(A_i)))$
$\rightarrow \mathbf{C}(\ \text{own\_evaluation}(E,\text{any\_new},\text{succeeded}) \wedge$
$\neg\text{own\_evaluation}(E,\text{any\_new},\text{failed}));$

– $(\mathbf{PC}(\text{own\_extent}(\text{any\_new}) \wedge \text{finished} \wedge \text{own\_task\_control\_focus}(E)) \wedge$
$\bigwedge_{i=1}^{n}(\mathbf{C}(\text{target}(E,A_i,\text{confirm})) \wedge$
$(\neg\mathbf{C}\text{true}(A_i) \vee (\mathbf{C}\text{true}(A_i) \wedge \mathbf{PC}\text{true}(A_i)))))$
$\rightarrow \mathbf{C}(\ \text{own\_evaluation}(E,\text{any\_new},\text{failed}) \wedge$
$\neg\text{own\_evaluation}(E,\text{any\_new},\text{succeeded}));$

– $(\mathbf{PC}(\text{own\_extent}(\text{any\_new}) \wedge \text{finished} \wedge \text{own\_task\_control\_focus}(E)) \wedge$
$\bigwedge_{i=1}^{n}(\mathbf{C}(\text{target}(E,A_i,\text{reject})) \wedge$
$(\neg\mathbf{C}\text{false}(A_i) \vee (\text{false}(A_i) \wedge \mathbf{PC}\text{false}(A_i)))))$
$\rightarrow \mathbf{C}(\ \text{own\_evaluation}(E,\text{any\_new},\text{failed}) \wedge$
$\neg\text{own\_evaluation}(E,\text{any\_new},\text{succeeded}));$

– $(\mathbf{PC}(\text{own\_extent}(\text{any\_new}) \wedge \text{finished} \wedge \text{own\_task\_control\_focus}(E)) \wedge$
$\bigwedge_{i=1}^{n}(\mathbf{C}(\text{target}(E,A_i,\text{determine}) \wedge$
$(\neg\mathbf{C}\text{known}(A_i) \vee (\text{known}(A_i) \wedge \mathbf{PC}\text{known}(A_i)))))$
$\rightarrow \mathbf{C}(\text{own\_evaluation}(E,\text{any\_new},\text{failed}) \wedge$
$\neg\text{own\_evaluation}(E,\text{any\_new},\text{succeeded}));$

– $(\mathbf{PC}(\text{own\_extent}(\text{every}) \wedge \text{own\_task\_control\_focus}(E)) \wedge$
$\bigwedge_{i=1}^{n}(\mathbf{C}(\text{target}(E,A_i,\text{confirm}) \wedge$
$\text{true}(A_i))))$
$\rightarrow \mathbf{C}(\text{own\_evaluation}(E,\text{every},\text{succeeded}) \wedge$
$\neg\text{own\_evaluation}(E,\text{every},\text{failed}));$

– $(\mathbf{PC}(\text{own\_extent}(\text{every}) \wedge \text{own\_task\_control\_focus}(E)) \wedge$
$\bigwedge_{i=1}^{n}(\mathbf{C}(\text{target}(E,A_i,\text{reject}) \wedge$
$\text{false}(A_i))))$
$\rightarrow \mathbf{C}(\text{own\_evaluation}(E,\text{every},\text{succeeded}) \wedge$
$\neg\text{own\_evaluation}(E,\text{every},\text{failed}));$

– $(\mathbf{PC}(\text{own\_extent}(\text{every}) \wedge \text{own\_task\_control\_focus}(E)) \wedge$
$\bigwedge_{i=1}^{n}(\mathbf{C}(\text{target}(E,A_i,\text{determine}) \wedge$
$\text{known}(A_i))))$
$\rightarrow \mathbf{C}(\text{own\_evaluation}(E,\text{every},\text{succeeded}) \wedge$
$\neg\text{own\_evaluation}(E,\text{every},\text{failed}));$

– $(\mathbf{PC}(\text{own\_extent}(\text{every}) \wedge \text{finished} \wedge \text{own\_task\_control\_focus}(E)) \wedge$
$\bigvee_{i=1}^{n}(\mathbf{C}(\text{target}(E,A_i,\text{confirm}) \wedge$
$\neg\text{true}(A_i))))$
$\rightarrow \mathbf{C}(\text{own\_evaluation}(E,\text{every},\text{failed}) \wedge$
$\neg\text{own\_evaluation}(E,\text{every},\text{succeeded}));$

– $(\mathbf{PC}(\text{own\_extent}(\text{every}) \wedge \text{finished} \wedge \text{own\_task\_control\_focus}(E)) \wedge$
$\bigvee_{i=1}^{n}(\mathbf{C}(\text{target}(E,A_i,\text{reject}) \wedge$
$\neg\text{false}(A_i))))$
$\rightarrow \mathbf{C}(\text{own\_evaluation}(E,\text{every},\text{failed}) \wedge$
$\neg\text{own\_evaluation}(E,\text{every},\text{succeeded}));$

– $(\mathbf{PC}(\text{own\_extent}(\text{every}) \wedge \text{finished} \wedge \text{own\_task\_control\_focus}(E)) \wedge$
$\bigvee_{i=1}^{n}(\mathbf{C}(\text{target}(E,A_i,\text{determine}) \wedge$
$\neg\text{known}(A_i))))$
$\rightarrow \mathbf{C}(\text{own\_evaluation}(E,\text{every},\text{failed}) \wedge$
$\neg\text{own\_evaluation}(E,\text{every},\text{succeeded}));$

– $(\mathbf{PC}(\text{own\_extent}(\text{all\_p}) \wedge \text{own\_task\_control\_focus}(E))$
$\rightarrow \mathbf{C}(\ \text{own\_evaluation}(E,\text{any},\text{succeeded}) \wedge$
$\neg\text{own\_evaluation}(E,\text{any},\text{failed}));$

The first formula in Definition 9.41 defines, for the case that C is a composed component, finished as the expression $\bigwedge_{i=1}^{n} \text{stopped}(C_i)$, for $C_i \in Subc(C)$. Truth values for $\text{stopped}(C_i)$ are transmitted from the subcomponents to C by upward control links. For primitive components and control components, finished is not defined in terms of another proposition symbol. Instead, the reasoning engines associated with these components are expected to set truth values for finished directly. (In other words, truth values for finished are transmitted via implicit upward control links.)

Except for the first formula, all formulae given in Definition 9.41 are implications with a condition consisting of subformulae $\mathbf{PC}\varphi$ or $\mathbf{C}\varphi$ (for φ a non-temporal formula) and a conclusion of the form $\mathbf{C}\varphi$ (for φ a non-temporal formula). By Definition 9.40, all local component traces for a component C that satisfy the formulae presented in Definition 9.41 are elements of $Beh_{loc}(C)$. Many local component traces satisfy the formulae presented in Definition 9.41: because of the form of these formulae, the only requirement for local component traces in $Beh_{loc}(C)$ is that if two consecutive states satisfy the condition, than the latest of these states must satisfy the conclusion. However, in a set of compatible multitraces that describe the behaviour of a compositional system, the set of local component traces for C that actually occur in these multitraces is, in general, a true subset of $Beh_{loc}(C)$: in compatible multitraces, the set of all local component traces that occur is constrained by information transmission. In other words, the formulae that constitute the DESIRE local behaviour specification for components as defined in Definition 9.41 only partially describe the behaviour of a component.

In Figure 9.8, the behaviour of a composed component C in relation with other components is illustrated. Figure 9.8 depicts a local component trace for a composed component C, its control component C_{ctr}, a subcomponent A, the reasoning engine RE_{ctr} associated with C_{ctr}, and an export mediating link EL from A to C. (As this link plays only a small role in the figure, most of it is depicted in light gray to avoid unnecessary cluttering of the figure.) Assume that the knowledge base associated with C_{ctr} contains the following rules:

if start
then next_component_state(A, active);

if evaluation(A, tcf_2, any, succeeded)
and not previous_evaluation(A, tcf_2, any, succeeded)
then next_link_state(EL, uptodate);

Furthermore, assume that the default extent of A is any and that for C, a task control focus named tcf_1 is defined which contains an atom a.

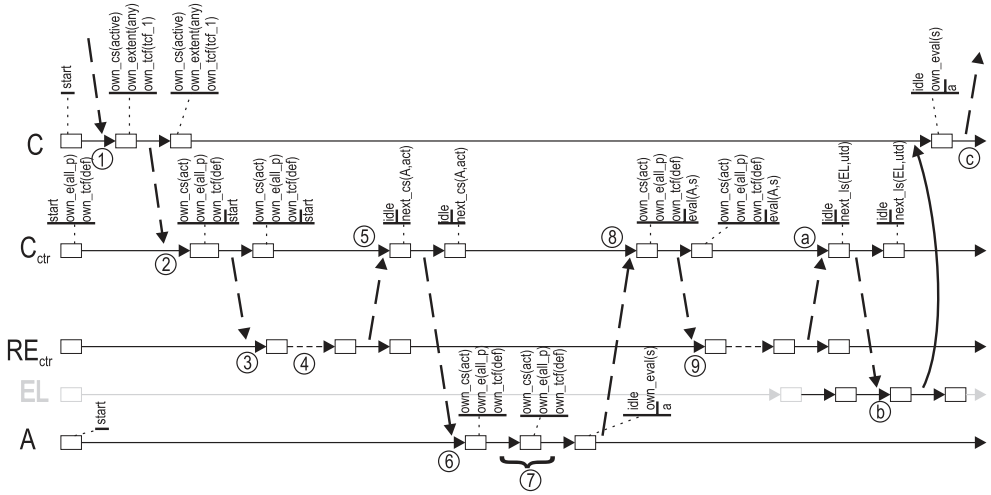

Figure 9.8: Example of component behaviour.

Figure 9.8 follows a number of conventions to illustrate the behaviour of a component. Local component states are depicted as rectangles, while transitions between states are depicted as solid, horizontal arrows. The rectangles, together with the solid, horizontal arrows that connect them, depict local component or link traces. The valuation of each state is indicated by listing the propositional symbols that are true. The names of some proposition symbols are abbreviated as follows: own_cs(act) for own_component_state(active), own_e for own_extent, and similarly for some other symbols. Furthermore, some arguments are omitted. Groups of proposition symbols that together determine the valuation of a specific state are indicated by a thick horizontal bar connected to that state by a dashed line. A short, thick vertical bar connected to the horizontal bar indicates which symbols describe the control part (left hand side of the vertical bar) and which symbols describe the kernel part (right hand side of the vertical bar). If no vertical bar is present, all symbols are used for the control part. To avoid cluttering the figure too much, input and output substates are not indicated. A broken arrow in Figure 9.8 depict a transmission octet consisting of the two states at the beginning and end of the horizontal arrow from which the broken arrow departs to the two states at the beginning and end of the horizontal arrow to which the broken arrow points. (The four link states that are also present in a transmission octet are not depicted in Figure 9.8.) The numbers in Figure 9.8 depict the following events:

- 1: Control information arrives via a downward control link from the parent of *C* to *C*. The arrival of this information triggers an import control link from *C* to its control component C_{ctr}, because the combination of the state immediately before and after the transmission labelled '1' form the first two states of a transmission octet for the import control link;

- 2: The control component receives control information. The arrival of this information triggers a downward control link from C_{ctr} to its reasoning engine;
- 3: Control information is transmitted to the reasoning engine of C_{ctr}. As a result, in the input information state of the reasoning engine, start gets truth value 1.
- 4: Based on the control knowledge rules associated with C_{ctr}, the reasoning engine derives new information (next_component_state(A,active)), which may take more than one step (depending on how the behaviour of the reasoning engine is modelled). Therefore, the arrow labelled '4' is dashed;
- 5: New information derived by the reasoning engine is transmitted to C_{ctr}. The arrival of this information triggers a downward control link from C_{ctr} to the controlled subcomponent *A*;
- 6: The newly derived control information (arrow labelled '5') is forwarded to subcomponent *A*;
- 7: Subcomponent *A* performs its tasks. As *A* can itself be either a composed component or a primitive component, no details on the internal behaviour of subcomponent A are depicted in Figure 9.8. Similar to component *C* itself, *A* completes its work in two steps (this is elaborated below). Note that in the state at the end of the transitions that constitute step '7', own_evaluation(tcf_2,any,succeeded) (abbreviated as own_eval(s)) is true, as specified in Definition 9.41;
- 8: The evaluation information on *A*'s work is transmitted to C_{ctr} via an upward control link;
- 9: The arrival of information on the evaluation of *A* triggers the transmission of this new information to the reasoning engine of C_{ctr};
- a: New information (next_link_state(EL,uptodate)) derived by the reasoning engine of C_{ctr} arrives in C_{ctr};
- b: Link *EL* receives control information stating that it should transmit information from its domain (component *A*) to its co-domain (the output interface of *C)*;
- c: The curved, solid arrow labelled 'c' does not depict a transmission octet. Instead, it indicates how the actual transmission of information by link *EL* results in a new state of *C*. In this state, the (linked) kernel output atom a becomes true. By one of the formulae of Definition 9.41, also own_evaluation(tcf_1,any,succeeded) (abbreviated as own_eval(s)) becomes true.

Figure 9.8 shows two key characteristics of the application of the semantic structure to DESIRE. First, the local component traces for *C* and its subcomponent *A* are similar: both consist of four states, and the transitions between these states are caused by similar events: the arrival and sending of control information and delegation to a reasoning engine or subcomponents. Such similarity is needed to support compositionality: as in each level of the subcomponent hierarchy, behaviour is modelled in the same way, it is possible to embed components in other components while retaining their behaviour. Second, Figure 9.8 shows that if a composed component has been made active, the evaluation of the component is available at the next moment, regardless of the number of subcomponents that have to be activated. This characteristic is a consequence of the fact that in the semantic structure, for each component, a set of local component traces is distinguished. The notion of next state in the local component traces of a component is independent of the notion of next state of any other component, including subcomponents. In other words, if global time were available, it is possible to observe that the clocks implied by the notion of next state for each component tick at different rates, and that in general, the clocks of subcomponents tick faster than the clock of their parent component. An advantage of this characteristic is that it is possible to define a relatively simple specification of the local behaviour of a composed component, which is not cluttered by timing considerations of subcomponents. (If such considerations were to be taken into account in a local trace, than it is in general no longer possible to state that an evaluation is available at the next moment. In this case, it is difficult to specify the correct moment in time at which evaluations are available.)

Figure 9.8 illustrates how the presence of a rule **if** start **then** next_component_state(A, active) via control links actually results in subcomponent *A* becoming active. An interesting question is: how does the intuitive reading of **if** start **then** next_component_state(A, active) (which is a knowledge base rule) as a the temporal formula start$\rightarrow$**X**component_state(*A*,active) relate to the multitrace depicted in Figure 9.8? In general, the intuitive reading of knowledge rules for control components as temporal formulae calls for a global language that has not been defined in this thesis. For example, consider the rule **if** start **then** next_component_state(A, active) **and** next_component_state(B, active). The temporal reading corresponds to the formula start$\rightarrow$**X**component_state(*A*,active)$\wedge$**X**component_state(*B*,active), which refers to two components, *A* and *B*. Although it is possible that this formula is a local formula (i.e., by choosing the right sets of proposition symbols, this formula conforms to the language defined in Definition 9.36), the formula is not interesting if it is interpreted with respect to a local component trace, as the intuitive reading calls for a global interpretation.

However, it is possible to support the temporal reading of control knowledge rules using the notion of common global states presented in Chapter 8. As stated in Chapter 8, common global states are linearly ordered. This enables the interpretation of the language defined in Definition 9.36 with respect to the subset

of all global states for a structure hierarchy that are common global states. (A number of details have to be solved, such as which proposition symbols are interpreted in case the local languages are not disjoint.) With such an interpretation, it is possible to evaluate whether the formula start→**X**component_state(*A*,active) is true for the multitrace depicted in Figure 9.8 (the multitrace that, as explained above, corresponds to the control knowledge rule **if** start **then** next_component_state(A, active)).

In Figure 9.9, four possible common global states for the combination of the local traces of C_{ctr}, *EL* and *A* are indicated, σ_1 to σ_4. The order of these states is as follows: σ_1 is the earliest common global state, followed by σ_2, followed by σ_3 and finally σ_4. As indicated by the valuations depicted in Figure 9.9, σ_1 satisfies start, while σ_2 satisfies component_state(*A*,active). Therefore, σ_1 and σ_2 together satisfy the formula start→**X**component_state(*A*,active), interpreted at state σ_1. Likewise, as σ_2 does not satisfy evaluation(*A*,tcf_2,any,succeeded) while σ_3 does and σ_4 satisfies link_state(*EL*,uptodate), at state σ_3 the formula (¬**PC**evaluation(*A*,tcf_2,any,succeeded) ∧ **C**evaluation(*A*,tcf_2,any,succeeded))→**X**link_state(*EL*,uptodate). This shows that the multitrace depicted in Figure 9.8 and Figure 9.9 satisfy the intuitive temporal reading of the control knowledge rules associated with C_{ctr}.

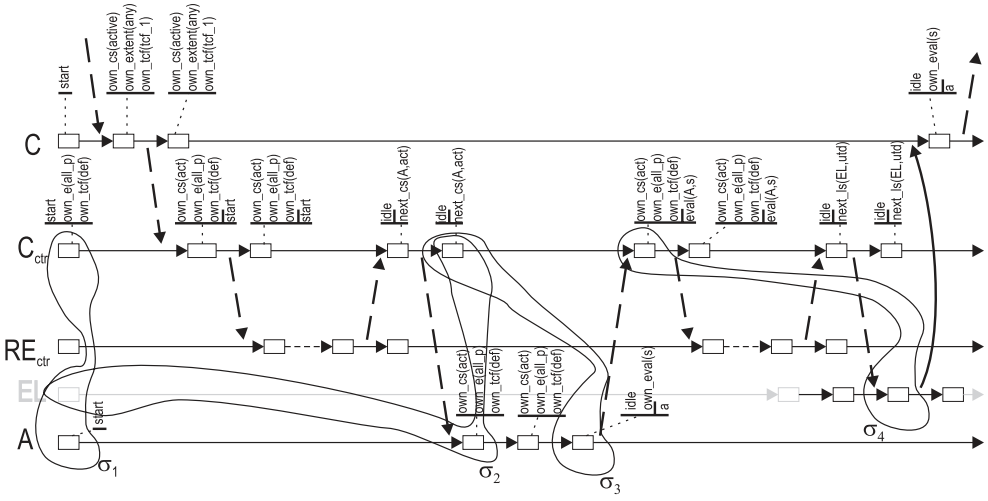

Figure 9.9: Common global states.

The questions is whether the common global states depicted in Figure 9.9 are indeed common global states, and whether there are no other common global states in between the common global states depicted in Figure 9.9. As explained in Chapter 8, a global state σ is a common global state if for each disjoint pair $\langle S1,S2 \rangle$ of components, where $S_1,S_2 \in \{C_{ctr},EL,A\}$, $prev_{\mu(S1)}(\sigma(S1)) \rightarrow_{sd} next_{\mu(S2)}(\sigma(S2))$. To ensure that the global states depicted in Figure 9.9 are common global states, and

that they are the only common global states, the implementation of DESIRE has to be carefully constructed in such a way that no unnecessary dependencies are introduced.

To conclude, the concepts developed in this thesis, particularly in Chapter 8, in principle enable the formal definition of a global interpretation of formulae such as $\mathsf{start} \rightarrow \boldsymbol{X} \mathsf{component\_state}(A, \mathsf{active})$. However, additional constraints on the implementation of DESIRE have to be imposed solely for this purpose that may restrict the efficiency of the implementation.

9.3.1.4 Local Link Traces

The three views on the behaviour of a composed component defined in Chapter 5 are defined relative to sets of local link traces $Beh_{loc}(I)$ for (a subset of) the links in a structure hierarchy. In this section, such sets are defined for links in DESIRE.

The elements of $Beh_{loc}(I)$ for a link I are local link traces, which consist of local link states. Local link states are defined in Section 9.2.2.3 as valuations for the DESIRE link state description signature (Definition 9.28). In other words, the state of a link in DESIRE is described by a local language defined over this signature.

With the temporal language defined in Section 9.3.1.1, a specification for the behaviour of links in DESIRE can be developed. Such a specification consists of temporal formulae that describe the behaviour of a link in terms of the DESIRE link state description signature. In DESIRE, from a local perspective, the behaviour of a link that is not a control link is almost not constrained. The only constraint is that a link cannot be idle and awake at the same time. So, for each link I in a DESIRE structure hierarchy with control that is not a control link, $Beh_{loc}(I)=\{lt \in \mathcal{L}_I \mid lt \models G(\mathsf{link\_state(idle)} \leftrightarrow \neg \mathsf{link\_state(awake)})\}$. Note that $\mathsf{link\_state(uptodate)}$ can be true at any time. The state of a control link is always awake, i.e., for a control link I, $Beh_{loc}(I)=\{lt \in \mathcal{L}_I \mid lt \models G\mathsf{link\_state(awake)}\}$.

From a more global perspective, the behaviour of links that are not control links is constrained by downward control links. The downward control links fully determine the state of links that are not control links. The three views on the behaviour of a composed component defined in Chapter 5 consist of compatible multitraces. Compatibility ensures that only local link traces that take information transmitted to them into account (in this case: information on their own state) are part of the overall behaviour of a composed component.

9.3.2 Compatibility Relations for DESIRE

As stated at the beginning of Section 9.3, each of the three views on the behaviour of a compositional system is relative to (i) a composition structure or a structure hierarcy, (ii) a family of sets of local component and link traces, and (iii) a collection of compatibility relations. Composition structures and structure hierarchies for DESIRE are defined in Section 9.2.2.1. Families of sets of local

component traces are defined in the previous section. In this section, collections of compatibility relations for DESIRE are presented.

As stated in Chapter 5 and Chapter 6, a compatibility relation for a link *I* is a ternary relation on the set of local component traces of the domain of *I*, the set of local link traces of *I*, and the set of local component traces of the co-domain of *I*. Chapter 6 defined properties of compatibility relations such as the lossless transmission property and the order-preserving transmission property. In DESIRE, information transmission is lossless and order-preserving, but not logically instantaneous. In (Brazier, Eck & Treur, 1996), an early inventory of possible choices with respect to properties of information transmission for DESIRE is reported. This inventory is summarised at the end of this chapter.

In Chapter 6, properties of compatibility relations are defined in terms of compatible state octets. Compatible state octets are defined in terms of information link mappings. To define compatibility relations for DESIRE, first information link mappings for control links are defined. In Section 9.2.2.3, information link mapping descriptions are defined as a means to specify information link mappings. The definition of compatibility relations for DESIRE therefore starts with the definition of information link mapping descriptions for DESIRE.

Information link mapping descriptions are octets of basic information elements. However, to describe information link mappings for control links in DESIRE, only the first and eighth basic information element are relevant, for the following reasons. First, In DESIRE, control links are always ready to transmit information. To describe the behaviour of multi-agent systems specified using DESIRE, it is not necessary to pay attention to the state of the control links themselves. Consequently, the third, fourth, fifth and sixth basic element, which are used to describe conditions on the state of the link itself, are not relevant. (The state of control links may be important to describe a specific implementation of DESIRE.) Second, components that transmit control information (the domain components of control links) do not evaluate the result of such information transmission. Therefore, the result of information transmission for the domain (e.g., whether control information has been succesfully sent) is not taken into account. Consequently, the second basic information element, which is used to describe results of information transmission at the domain of the link, is not relevant. Third, components that receive control information are always enabled for receipt of such control information. The seventh basic information element in an information link mapping description, which is used to specify enabling conditions, is therefore not relevant.

As only the first and eighth state of information link mappings are of importance, DESIRE control information link mapping descriptions are binary relations, defined as follows:

Definition 9.42. (DESIRE control information link mapping descriptions). *Let* $SH=\langle Comp \cup Ctr; Lnk \cup UCLnk \cup DCLnk \cup ECLnk \cup ICLnk; \prec; dom; cdom \rangle$ *be a DESIRE structure hierarchy with control, with Ctr a set of control components, UCLnk a set of*

upward control links, DCLnk a set of downward control links, ECLnk a set of export control links and ICLnk a set of import control links. Let C∈Comp be a component with control component C_{ctr}. Let TCF(S,SH) be the set of task control foci for S and let EC(S,SH) be the set of evaluation criteria for S (for S∈Comp).

- *Let UTCL∈UCLnk be an upward control link from a component or link S to C_{ctr}. The* DESIRE upward control information link mapping description Λ_{UTCL} *is defined as follows:*
  - *If S is a component, then:*
    Λ_{UTCL}= {⟨[own_evaluation(*ec,et,tt*):*s*];[evaluation(*S,ec,et,tt*):*s*]⟩ | *ec*∈*EC(S,SH)*,*et*∈{any,any_new,all_p,every},*tt*∈{succeeded,failed}, *s*∈{0,1,u}} ∪ {⟨[own_component_state(*at*):*s*];[component_state(*S,at*):*s*]⟩ | *at*∈{idle,active,awake},*s*∈{0,1,u}} ∪ {⟨[own_task_control_focus(*tcf*):*s*];[task_control_focus(*S,tcf*):*s*]⟩ | *tcf*∈*TCF(S,SH)*,*s*∈{0,1,u}} ∪ {⟨[own_extent(*et*):*s*];[extent(*S,et*):*s*]⟩ | *et*∈{any,any_new,all_p,every},*s*∈{0,1,u}}
  - *If S is a link, then* $\Lambda_{UTCL}=\varnothing$ *(currently, DESIRE does not use link monitoring links.)*
- *Let DTCL∈DCLnk be a downward control link from C_{ctr} to a component or link S. The* DESIRE downward control information link mapping description Λ_{DTCL} *is defined as follows:*
  - Λ_{DTCL}= {⟨[next_component_state(*S,at*):*s*];[own_component_state(*at*):*s*]⟩ | *at*∈{idle,active,awake},*s*∈{0,1,u}} ∪ {⟨[next_task_control_focus(*S,tcf*):*s*];[own_task_control_focus(*tcf*):*s*]⟩ | *tcf*∈*TCF(S,SH)*,*s*∈{0,1,u}} ∪ {⟨[next_extent(*S,et*):*s*];[own_extent(*et*):*s*]⟩ | *et*∈{any,any_new,all_p,every},*s*∈{0,1,u}}
  - *If S is a link, then:*
    Λ_{DTCL}= {⟨[next_link_state(*S,at*):*s*];[link_state(*at*):*s*]⟩ | *at*∈{idle,up_to_date,awake},*s*∈{0,1,u}} ∪ {⟨[next_link_sequence_state([…,*S*,…],*at*):*s*];[link_state(*at*):*s*]⟩ | *at*∈{idle,uptodate,awake},*s*∈{0,1,u}}
- *Let ITCL∈ICLnk be an import control link from C to C_{ctr}. The* DESIRE import control information link mapping description Λ_{ITCL} *is defined as follows:*
  - Λ_{ITCL} = {⟨[start:*s*];[start:*s*]⟩ | *s*∈{0,1,u}} ∪ {⟨[own_component_state(*at*):*s*];[own_component_state(*at*):*s*]⟩ | *at*∈{idle,active,awake},*s*∈{0,1,u}} ∪ {⟨[own_task_control_focus(*tcf*):*s*];[own_task_control_focus(*tcf*):*s*]⟩ | *tcf*∈*TCF(C,SH)*,*s*∈{0,1,u}} ∪

$\{\langle[\text{own\_extent}(et):s];[\text{own\_extent}(et):s]\rangle \mid$
$et \in \{\text{any,any\_new,all\_p,every}\}, s \in \{0,1,u\}\}$

- *Let ETCL∈ECLnk be an export control link from C_{ctr} to C. The* DESIRE export control information link mapping description Λ_{ETCL} *is defined as follows:*
  - $\Lambda_{ETCL} = \{\langle[\text{stop}:s];[\text{stop}:s]\rangle \mid s \in \{0,1,u\}\} \cup$
    $\{\langle[\text{own\_evaluation}(ec,et,tt):s];[\text{own\_evaluation}(ec,et,tt):s]\rangle \mid$
    $ec \in EC(C,SH), et \in \{\text{any,any\_new,all\_p,every}\}, tt \in \{\text{succeeded,failed}\},$
    $s \in \{0,1,u\}\} \cup$
    $\{\langle[\text{own\_component\_state}(at):s];[\text{own\_component\_state}(at):s]\rangle \mid$
    $at \in \{\text{idle,active,awake}\}, s \in \{0,1,u\}\} \cup$
    $\{\langle[\text{own\_task\_control\_focus}(tcf):s];[\text{own\_task\_control\_focus}(tcf):s]\rangle \mid$
    $tcf \in TCF(C), s \in \{0,1,u\}\} \cup$
    $\{\langle[\text{own\_extent}(et):s];[\text{own\_extent}(et):s]\rangle \mid$
    $et \in \{\text{any,any\_new,all\_p,every}\}, s \in \{0,1,u\}\}.$

Compatibility relations for DESIRE can now be defined in terms of DESIRE control information link mapping descriptions. To simplify the definition of compatibility relations for DESIRE, the function $\Lambda\Lambda$ (introduced in Section 9.2.2.1) that associates user-defined information link mapping descriptions with a DESIRE structure hierarchy (without control) is extended to DESIRE structure hierarchies with control. Given a DESIRE structure hierarchy with control $SH'=\langle Comp;Lnk \cup CLnk;\prec;dom;cdom\rangle$ generated from a DESIRE structure hierarchy SH, $\Lambda\Lambda(I,SH')=\Lambda\Lambda(I,SH)$ for I in Lnk and $\Lambda\Lambda(I,SH')$ is as defined in the previous definition for I in $CLnk$.

For a DESIRE structure hierarchy with control $SH'=\langle Comp;Lnk \cup CLnk;\prec;dom;cdom\rangle$, all information link mapping descriptions denoted by $\Lambda\Lambda(I,SH')$ are sets of *pairs* of basic information elements. However, in the general semantic structure, information link mapping descriptions consist of octets of eight basic information elements. For links $I \in Lnk$, the extension to octets of eight states is presented at the end of Section 9.2.2.3. For $I \in CLnk$, a pair of basic information elements $\langle[p:s];[q,s']\rangle$ as defined above for control links in DESIRE denotes the following complete information link mapping description:

$\{\langle\langle[p:s];[\mathbf{T}:1]\rangle;\langle[\text{link\_state(awake)}:1];[\mathbf{T}:1];[\mathbf{T}:1];[\mathbf{T}:1]\rangle;\langle[\mathbf{T}:1];[q,s']\rangle\rangle\}.$

Compatibility relations for DESIRE are defined as follows, where (similar to Chapter 6) a state $V_C(i)$ in a local trace is abbreviated to $v_{C,i}$:

Definition 9.43. (DESIRE compatibility relations collection). *Let $SH=\langle Comp;Lnk;\prec;dom;cdom\rangle$ be a DESIRE structure hierarchy with control. The* DESIRE compatibility relations collection *is the smallest collection $\gamma=(\gamma_I)_{I \in Lnk}$ of compatibility relations such that for each $\gamma_I=\langle LT_{dom(I)};LT_I;LT_{cdom(I)}\rangle$ with $LT_{dom(I)}=\langle\langle T_{dom(I)};<_{dom(I)}\rangle;V_{dom(I)}\rangle$, $LT_I=\langle\langle T_I;<_I\rangle;V_I\rangle$, and $LT_{cdom(I)}=\langle\langle T_{cdom(I)};<_{cdom(I)}\rangle;V_{cdom(I)}\rangle$:*

- *For all* $i \in T_{dom(I)}$: $\langle\langle v_{dom(I),i}; v_{dom(I),j}\rangle;\langle v_{I,i''}; v_{I,j''}; v_{I,k}; v_{I,l}\rangle;\langle v_{cdom(I),i'}; v_{cdom(I),j'}\rangle\rangle \models_{Lnk} \Lambda\Lambda(I,SH)$, *for some* $j \in T_{dom(I)}$, $i'',j'',k,l \in T_I$, $i',j' \in T_{cdom(I)}$ *with* $v_{dom(I),j} = next_{LT_I}(v_{dom(I),i})$, $v_{cdom(I),j'} = next_{LT_I}(v_{cdom(I),i'})$ *and* $i'' <_I j'' <_I k <_I l$;

- *For all* $j' \in T_{cdom(I)}$: $\langle\langle v_{dom(I),i}; v_{dom(I),j}\rangle;\langle v_{I,i''}; v_{I,j''}; v_{I,k}; v_{I,l}\rangle;\langle v_{cdom(I),i'}; v_{cdom(I),j'}\rangle\rangle \models_{Lnk} \Lambda\Lambda(I,SH)$, *for some* $j \in T_{dom(I)}$, $i'',j'',k,l \in T_I$, $i',j' \in T_{cdom(I)}$ *with* $v_{dom(I),j} = next_{LT_I}(v_{dom(I),i})$, $v_{cdom(I),j'} = next_{LT_I}(v_{cdom(I),i'})$ *and* $i'' <_I j'' <_I k <_I l$;

- *For all* $i,m \in T_{dom(I)}$: *if*

  $\langle\langle v_{dom(I),i}; v_{dom(I),j}\rangle;\langle v_{I,i''}; v_{I,j''}; v_{I,k}; v_{I,l}\rangle;\langle v_{cdom(I),i'}; v_{cdom(I),j'}\rangle\rangle \models_{Lnk} \Lambda\Lambda(I,SH)$

  and $\langle\langle v_{dom(I),m}; v_{dom(I),n}\rangle;\langle v_{I,m''}; v_{I,n''}; v_{I,o}; v_{I,p}\rangle;\langle v_{cdom(I),m'}; v_{cdom(I),n'}\rangle\rangle \models_{Lnk} \Lambda\Lambda(I,SH)$

  then $j' <_{cdom(I)} m'$.

The first and second clause of this definition state that for a triple of traces $\langle LT_{dom(I)}; LT_I; LT_{cdom(I)}\rangle$ in a DESIRE compatibility relation, each state in $LT_{dom(I)}$ and $LT_{cdom(I)}$, respectively, must be part of a compatible state octet that satisfies $\Lambda\Lambda(I,SH)$, the information link mapping description of I. In other words, according to the definition of satisfaction of information link mapping descriptions given in Section 9.2.2.3, the first clause states that for any state in $LT_{dom(I)}$, one of the following conditions hold:

- According to the information link mapping of I, the state is not involved in information transmission via link I, or
- The state is involved in information transmission via link I, and in the trace related by compatibility for the co-domain of I, a pair of states occurs in which the information is received.

The second clause states that for any state in $LT_{cdom(I)}$, one of the following conditions hold:

- According to the information link mapping of I, the state is not involved in information transmission via link I, or
- The state is involved in information transmission via link I, and in the trace related by compatibility for the domain of I, a pair of states occurs in which the information is sent.

The third clause of the definition is exactly the order-preserving transmission property defined in Chapter 6.

9.3.3 Sequential link activations

In DESIRE, groups of link activations can be performed in a user-specified order, which is called *sequential link activation*. In task control rules, sequential link

activation is specified using the predicate next_link_sequence_state(*LinkList*, *ActivationType*), as in the following example rule:

```
if      evaluation(OPC,default_targets,any,succeeded)
        and not previous_evaluation(OPC,default_targets,any,succeeded)
then    next_link_sequence_state([L1,L2],uptodate);
```

In this rule, a typical condition (which denotes a new evaluation result becoming available for a component called OPC) is followed by conclusion next_link_sequence_state([L1, L2],uptodate). The square brackets indicate that the two link activations are grouped and will be performed in the order specified: first link L1, then link L2. This means that information transmission by link L2 only starts after information transmission by link L1 has finished.

The DESIRE compatibility relations defined in the previous subsection do not ensure that sequential link activations are executed in the right order. As there are no links between the links that are to be activated sequentially (i.e., L1 and L2 in the example), there are no compatibility relations that relate the occurrence of states for L1 and L2 or their co-domains. To ensure that sequential link activations are executed in the right order, several adaptions to the definitions presented in this chapter can be considered.

First, the global perspective developed in Chapter 7 and Chapter 8 can be used. Sequential link activation is a global-level phenomenon: it relates the order of occurrences of specific states at different locations in a multi-agent system. Sequential link activation is comparable to the intuitive global reading of task control rules discussed at the end of Section 9.3.1.3. The formal definition of the requirement below expresses the correct activation order for sequential link activations in terms of common strict global states, similar to the intuitive reading of task control rules.

Definition 9.44. (Correct link sequence activation property). *Let* $SH=\langle Comp \cup Contr;Lnk;\prec;dom;cdom\rangle$ *be a DESIRE structure hierarchy with control, let* γ *be a collection of DESIRE compatibility relations and let* μ *be a multitrace compatible for* γ. *Then* $\langle CSGS(SH,\mu,\gamma);next_{CSGS(SH,\mu,\gamma)}\rangle$ *has the* correct link sequence activation property *if for each* σ,σ',σ'' *in* $CSGS(SH,\mu,\gamma)$ *and for each* C_{ctr} *in Contr:*
if $\sigma(C_{ctr}) \models_3^+$ next_link_sequence_state([...,*L1*,*L2*,...],*at*) *and* $\sigma'(L1) \models_3^+$ link_state(*at*) *and* $\sigma'(L2) \models_3^+$ link_state(*at*), *then* $\langle\sigma;\sigma'\rangle \in next_{CSGS(SH,\mu,\gamma)}$ *and* $\langle\sigma';\sigma''\rangle \in next_{CSGS(SH,\mu,\gamma)}$.

Note that the partial order relation $next_{CSGS(SH,\mu,\gamma)}$ is linear for common strict global states.

Second, a seventh kind of information links can be introduced. This kind of links would have other links as domain *and* as co-domain. With carefully defined compatibility relations for these links, it is probably possible to ensure sequential link activation in the correct order. However, the seventh kind of information link has to be introduced at the level of the general semantic structure, only to address an issue arising for this specific application;

The third alternative is to establish an indirect relation via link monitoring links and the control component that controls L1 and L2 instead of a direct relation. In this case, additional rules are needed for the behaviour of the control component. This is beneficial, as a rule in which next_link_sequence_state occurs can be viewed as an abbreviation for these additional rules. In fact, the normal, non-formal meaning of next_link_sequence_state is an abbreviation for rules that do not use next_link_sequence_state. However, for a precise, formal definition of the meaning of next_link_sequence_state in the form of an abbreviation of other rules, additional details have to be taken care of. For instance, sequential link activations are atomic in the sense that during the entire transmission via L1 and L2, no other task control rules can fire. Consequently, all rules in Definition 9.41 need an additional condition to ensure that the behaviour specified by them does not interfere with sequential link activations.

As the third alternative closely matches the normal, non-formal meaning of next_link_sequence_state, it is the best way to incorporate link sequence activations in the semantic structure.

9.3.4 Summary: The Dynamics of DESIRE

The previous subsections defined a family of sets of local component and link traces and a collection of compatibility relations for DESIRE. These definitions suffice to describe the dynamics of multi-agent systems specified using DESIRE. Given a DESIRE structure hierarchy *SH*, together with task control foci *TCF(C,SH)*, evaluation criteria *EC(C,SH)*, default extent *EXT(C,SH)* and other additional information associated with *SH* as described in Section 9.2.2.1, a DESIRE structure hierarchy *SH'* with control can be constructed from *SH*. The dynamics of the multi-agent system described by *SH* is defined by the three views on behaviour defined in Chapter 5: the black box view, the white box view and the glass box view, such that the input persistence (defined in Chapter 6) property holds. The three views on behaviour are defined relative to *SH'* and to a family of sets of local component and link traces and a collection of compatibility relations which are given in the previous sections.

9.3.5 Local Behaviour of Primitive Components with a Knowledge Base

As stated in Section 9.3.1.3, reasoning engines are responsible for carrying out computations in primitive and control components. It is interesting to describe more precisely the behaviour of reasoning engines for primitive and control components with which a knowledge base is associated, as such components are most commonly used in DESIRE. In this section, the relation between task control and the execution of knowledge rules is discussed.

Knowledge rules in primitive components as well as in control components are (non-temporal) logical implications of the form defined in Definition 9.6. In general, in DESIRE the chaining process outlined Section 9.3.1.2 does not compute

all logical consequences of the input information and knowledge rules. Instead, a subset determined by task control foci and extents is computed. As a consequence, knowledge rules cannot be directly incorporated in the specification of behaviour of a component: the (classical) interpretation of these rules would specify that *all* consequences of a specific input substate (its deductive closure) are *immediately available* regardless of task control foci and extents.

The solution to this problem is taken from (Engelfriet & Treur, 1994; Engelfriet, 1999). The general form of the solution is as follows. For each knowledge rule if φ then ψ in a knowledge base of a component, an atomic proposition symbol $at_{\varphi \rightarrow \psi}$ can be used in the specification of the behaviour of the component. Thus, the knowledge rule if φ then ψ is lifted to a meta level, where it is represented by the symbol $at_{\varphi \rightarrow \psi}$. For each rule, a formula of the following form is added to the specification: $\boldsymbol{C}(\varphi \wedge at_{\varphi \rightarrow \psi}) \rightarrow \boldsymbol{X}\psi$, which is read as: if currently φ holds and a rule if φ then ψ is present, then in the next state, ψ holds. Thus, this rule describes how conclusions are drawn over time. (In practise, detailed control over the reasoning process is often added, in which case not every ψ is derived at the next moment in time.)

In the semantic structure, the behaviour of a DESIRE component *C* is described by propositional temporal formula, which determine the set $Beh_{loc}(C)$. The set of atomic formulae is defined as the set of ground atoms of a specific signature. For knowledge rules, an important part of this signature consists of sorts, objects and functions used to lift knowledge rules to a meta level. These sorts, objects and functions form the *primitive component meta signature*, which is defined as follows:

Definition 9.45. (Primitive component meta signature). *The* primitive component meta signature Σ_{prim} *is a signature* $\langle \langle S; < \rangle; Func; Pred \rangle$ *with:*

- $S = \{$PREM,CONC,FOR,LIT,IOA,ATOM$\}$;
- $< = \{\langle$PREM;FOR$\rangle, \langle$CONC;FOR$\rangle, \langle$IOA;ATOM$\rangle, \langle$ATOM;LIT$\rangle, \langle$LIT;FOR$\rangle\}$;
- *Func is an* S^+*-indexed family of sets with the following members:*
  - $Func_{\langle \text{ATOM;LITERAL} \rangle} = \{\text{not}\}$;
  - $Func_{\langle \text{LITERAL;FOR;FOR} \rangle} = \{\text{conj}\}$;
  - $Func_{\langle \text{FOR;PREM} \rangle} = \{\text{prem}\}$;
  - $Func_{\langle \text{FOR;CONC} \rangle} = \{\text{conc}\}$;
- *Pred is an* S^**-indexed family of sets with the following members:*
  - $Pred_{\langle \text{LITERAL} \rangle} = \{\text{fact}\}$;
  - $Pred_{\langle \text{FOR} \rangle} = \{\text{true,false}\}$;
  - $Pred_{\langle \text{PREM;CONC} \rangle} = \{\text{rule}\}$;

The reasoning process is described in a way similar to the description of DESIRE local component behaviour presented in Section 9.3.1.3. Thus, a temporal language is used which is defined in terms of the local component specification language. The local component specification language, in turn, assumes that three sets of

propositional symbols are given. To describe the reasoning process executed by chaining in a primitive DESIRE component or control component *C*, the following sets are used:

Definition 9.46. (DESIRE meta level description). *Let SH=⟨Comp;Lnk;≺;dom;cdom⟩ be a DESIRE structure hierarchy with control that represents a DESIRE model, let C∈Comp be a primitive or control component with which a knowledge base is associated and let speclevel be a level identifier not in LEV(C,SH) such that speclevel is the top element in the partial order associated with LEV(C,SH). To describe the behaviour of a component in SH, the following sets of proposition symbols are used:*

- *If C is a primitive component that is not a control component:*
  - $Prop_{spec,C,in} = Gratom(\Sigma_{prim} \oplus meta_{\langle IOA,speclevel\rangle}(\Sigma\Sigma_{in}(C,SH) \oplus \Sigma\Sigma_{int}(C,SH)) \oplus \Sigma_{C,in}{}^{tc})$;
  - $Prop_{spec,C,int} = \varnothing$;
  - $Prop_{spec,C,out} = Gratoms(\Sigma_{prim} \oplus meta_{\langle IOA,speclevel\rangle}(\Sigma\Sigma_{out}(C,SH) \oplus \Sigma\Sigma_{int}(C,SH)) \oplus \Sigma_{C,out}{}^{tc})$;
- *If C is a control component:*
  - $Prop_{spec,C,in} = Gratom(\Sigma_{prim} \oplus meta_{\langle IOA,speclevel\rangle}(\Sigma_{C,in}{}^{tc}))$;
  - $Prop_{spec,C,int} = \varnothing$;
  - $Prop_{spec,C,out} = Gratoms(\Sigma_{prim} \oplus meta_{\langle IOA,speclevel\rangle}(\Sigma_{C,out}{}^{tc}))$.

As this definition show, user-supplied signatures $\Sigma\Sigma_{in}(C,SH)$, $\Sigma\Sigma_{int}(C,SH)$, and $\Sigma\Sigma_{out}(C,SH)$ are lifted to the sort IOA, which is a subsort of ATOM in Σ_{prim}. Thus, atoms used to describe the kernel part of DESIRE components and used in knowledge bases of DESIRE primitive components, are incorporated in the meta-level signature Σ_{prim}.

All rules in the knowledge base *KB(C,SH)* of a primitive component *C* are lifted to the specification meta-level to form atoms of the form rule*(…,…)*. (These atoms play the same role as atoms $at_{\varphi\to\psi}$ in (Engelfriet & Treur, 1994; Engelfriet, 1999).) The exact translation of knowledge base rules to atoms of the form rule*(…,…)* is straightforward and is not presented in this thesis. Instead, an example shows the result of the meta-lifting.

Example 9.47. A knowledge base rule

if P_1 **and** … **and** P_n **and not** P'_1 **and** … **and not** $P'_{n'}$
then C_1 **and** … **and** C_n **and not** C'_1 **and** … **and not** $C'_{n'}$;

is represented at the specification level as:

rule(prem(conj(P_1,conj(…,conj(P_n,conj(not(P'_1),conj(…,not($P'_{n'}$)…)))…))),
conc(conj(C_1,conj(…,conj(C_n,conj(not(C'_1),conj(…,not($C'_{n'}$)…)))…))))) ■

The following serves as a description of the relation between task control and the reasoning process by chaining for a primitive or control component. The description consists of a number of *propositional* temporal formulae. The variables and quantification over these variables used in the definition below is *not* part of the temporal language. Instead, these variables and quantification are part of the notation used in this thesis to define mathematical notions. The definition is followed by an example, which shows the propositional structure of the specification.

Definition 9.48. (DESIRE behaviour of knowledge base components specification). *Let SH=⟨Comp;Lnk;≺;dom;cdom⟩ be a DESIRE structure hierarchy with control and let C∈Comp be a primitive component or control component with which a knowledge base is associated. Let:*

- *T range over the set TCF(C,SH) of task control foci of C;*
- *E range over the set EC(C,SH) of evaluation criteria of C;*
- *P range over the set of premises of all rules in KB(C,SH);*
- *Each L, Lp_j, and each Lc_i range over sets of terms of Σ_{prim}, where each set only contains terms of the appropriate function symbol*[1]*;*

The DESIRE behaviour of knowledge base components *is described by a specification Spec(C) with the following formulae:*

- *For each fact L in the knowledge base of C, there is a formula* **C**fact(*L*)*;*
- *For each rule* if Lp_1 and … and Lp_n then Lc_1 and … and $Lc_{n'}$ *in KB(C,SH), the specification of C contains the following formulae:*
  - rule(prem(conj(Lp_1,conj(…,Lp_n)…))), conc(conj(Lc_1,conj(…,$Lc_{n'}$)…)));
- *For each premise P:* satisfied(*P*) ↔ true(A_1) ∧ true(A_n) ∧ false(A'_1) ∧ false($A'_{n'}$) *if P is the term conj(A_1,conj(…,conj(A_n,conj(not(A'_1),conj(…,not($A'_{n'}$)…)))…)));*
- *For each rule* if Lp_1 and … and Lp_n then Lc_1 and … and Lc_i and … and $Lc_{n'}$ *in KB(C,SH), for each target T in TCF(C,SH), the specification of C contains the following formulae (with Lc_i an atom):*
  - $\bigvee_{i=1}^{n'}$((**C**(rule(prem(*P*), conc(conj(Lc_1,conj(…,conj(Lc_i,conj(…,$Lc_{n'}$)…)…)))) ∧ satisfied(*P*) ∧ own_task_control_focus(*T*) ∧

[1] Consider a rule **if** a(X:INT) **and** b(Y:INT) **then** … This rule is represented as rule(prem(conj(a(*x*),b(*y*))),conc(…)). In an expression 'rule(prem(conj(Lp_1,Lp_2)),conc(…))', Lp_1 ranges over the set {a(*0*),a(*1*),…} and Lp_2 ranges over the set {b(*0*),b(*1*),…}.

(target(T,Lc_i,confirm) $\vee$ target(T,Lc_i,determine)) $\wedge$
(target(E,Lc_i,confirm) $\vee$ target(E,Lc_i,determine)) $\wedge$
own_extent(any)))
$\rightarrow$ **XC**(true(Lc_i) $\wedge$
own_evaluation(E,any,succeeded) $\wedge$
$\neg$own_evaluation(E,any,failed)));

- $\bigvee_{i=1}^{n'}$((**C**(rule(prem(P),
conc(conj(Lc_1,conj(…,conj(A_i,conj(…,$Lc_{n'}$)…)…)))) $\wedge$
satisfied(P) $\wedge$
own_task_control_focus(T) $\wedge$
(target(T,A_i,confirm) $\vee$ target(T,A_i,determine)) $\wedge$
(target(E,A_i,confirm) $\vee$ target(E,A_i,determine)) $\wedge$
own_extent(any_new)) $\wedge$
$\neg$**YC**known(A_i))
$\rightarrow$ **XC**(true(A_i) $\wedge$
own_evaluation(E,any_new,succeeded) $\wedge$
$\neg$own_evaluation(E,any_new,failed)));

- $\bigwedge_{i=1}^{n'}$((**C**(rule(prem(P),
conc(conj(Lc_1,conj(…,conj(A_i,conj(…,$Lc_{n'}$)…)…)))) $\wedge$
satisfied(P) $\wedge$
own_task_control_focus(T) $\wedge$
(target(T,A_i,confirm) $\vee$ target(T,A_i,determine)) $\wedge$
(target(E,A_i,confirm) $\vee$ target(E,A_i,determine)) $\wedge$
(own_extent(every) $\vee$ own_extent(all_p))))
$\rightarrow$ **XC**(true(A_i) $\wedge$
own_evaluation(E,every,succeeded) $\wedge$
$\neg$own_evaluation(E,every,failed)));

- *For each rule* if Lp_1 and … and Lp_n then Lc_1 and … and Lc_i and … and $Lc_{n'}$ *in KB(C,SH), for each target T in TCF(C,SH), the specification of C contains the following formulae (with* Lc_i *a literal* not(A) *for some atom A):*

- $\bigvee_{i=1}^{n'}$((**C**(rule(prem(P),
conc(conj(Lc_1,conj(…,conj(Lc_i,conj(…,$Lc_{n'}$)…)…)))) $\wedge$
satisfied(P) $\wedge$
own_task_control_focus(T) $\wedge$
(target(T,Lc_i,reject) $\vee$ target(T,Lc_i,determine)) $\wedge$
(target(E,Lc_i,reject) $\vee$ target(E,Lc_i,determine)) $\wedge$
own_extent(any)))
$\rightarrow$ **XC**(false(Lc_i) $\wedge$

own_evaluation(*E*,any,succeeded) ∧
¬own_evaluation(*E*,any,failed)));

– $\bigvee_{i=1}^{n'}$((**C**(rule(prem(*P*),
conc(conj(*Lc*$_1$,conj(…,conj(*Lc*$_i$,conj(…,*Lc*$_{n'}$)…)…)))) ∧
satisfied(*P*) ∧
own_task_control_focus(*T*) ∧
(target(*T*,*Lc*$_i$,reject) ∨ target(*T*,*Lc*$_i$,determine)) ∧
(target(*E*,*Lc*$_i$,reject) ∨ target(*E*,*Lc*$_i$,determine)) ∧
own_extent(any_new)) ∧
¬**YC**known(*Lc*$_i$))
→ **XC**(false(*Lc*$_i$) ∧
own_evaluation(*E*,any_new,succeeded) ∧
¬own_evaluation(*E*,any_new,failed)));

– $\bigwedge_{i=1}^{n'}$((**C**(rule(prem(*P*),
conc(conj(*Lc*$_1$,conj(…,conj(*Lc*$_i$,conj(…,*Lc*$_{n'}$)…)…)))) ∧
satisfied(*P*) ∧
own_task_control_focus(*T*) ∧
(target(*T*,*Lc*$_i$,reject) ∨ target(*T*,*Lc*$_i$,determine)) ∧
(target(*E*,*Lc*$_i$,reject) ∨ target(*E*,*Lc*$_i$,determine)) ∧
(own_extent(every) ∨ own_extent(all_p)))
→ **XC**(false(*Lc*$_i$) ∧
own_evaluation(*E*,every,succeeded) ∧
¬own_evaluation(*E*,any,failed)));

- *For each rule* if *Lp*$_1$ and … and *Lp*$_j$ and … and *Lp*$_n$ then *Lc*$_1$ and … and *Lc*$_i$ and … and *Lc*$_{n'}$ *in KB(C,SH), for each target T in TCF(C,SH), the specification of C contains the following formulae (with Lp*$_j$ *and Lc*$_i$ *atoms):*

– $\bigwedge_{j=1}^{n}\bigwedge_{i=1}^{n'}$((**C**(rule(prem(conj(*Lp*$_1$,conj(…,conj(*Lp*$_j$,conj(…,*Lp*$_n$)…)…))),
conc(conj(*Lc*$_1$,conj(…,conj(*Lc*$_i$,conj(…,*Lc*$_{n'}$)…)…)))) ∧
own_task_control_focus(*T*) ∧
(target(*T*,*Lc*$_i$,confirm) ∨ target(*T*,*Lc*$_i$,determine)) ∧
¬known(*Lc*$_i$) ∧
¬known(*Lp*$_j$)))
→ **XC**(required(*Lp*$_j$,pos)));

- *For each rule* if *Lp*$_1$ and … and *Lp*$_j$ and … and *Lp*$_n$ then *Lc*$_1$ and … and *Lc*$_i$ and … and *Lc*$_{n'}$ *in KB(C,SH), for each target T in TCF(C,SH), the specification of C contains the following formulae (with Lp*$_j$ *a literal* not(*A*) *for some atom A and Lc*$_i$ *an atom):*

– $\bigwedge_{j=1}^{n}\bigwedge_{i=1}^{n'}$((**C**(rule(prem(conj(Lp_1,conj(…,conj(Lp_j,conj(…,Lp_n)…)…))),
conc(conj(Lc_1,conj(…,conj(Lc_i,conj(…,$Lc_{n'}$)…)…)))) ∧
own_task_control_focus(T) ∧
(target(T,Lc_i,confirm) ∨ target(T,Lc_i,determine)) ∧
¬known(Lc_i) ∧
¬known(Lp_j)))
→ **XC**(required(A,neg)));

- *For each rule* if Lp_1 and … and Lp_j and … and Lp_n then Lc_1 and … and Lc_i and … and $Lc_{n'}$ *in KB(C,SH), for each target T in TCF(C,SH), the specification of C contains the following formulae (with* Lp_j *an atom and* Lc_i *a literal* not(A) *for some atom A):*

– $\bigwedge_{j=1}^{n}\bigwedge_{i=1}^{n'}$((**C**(rule(prem(conj(Lp_1,conj(…,conj(Lp_j,conj(…,Lp_n)…)…))),
conc(conj(Lc_1,conj(…,conj(Lc_i,conj(…,$Lc_{n'}$)…)…)))) ∧
own_task_control_focus(T) ∧
(target(T,A,reject) ∨ target(T,A,determine)) ∧
¬known(A) ∧
¬known(Lp_j)))
→ **XC**(required(Lp_j,pos)));

- *For each rule* if Lp_1 and … and not AP_j and … and Lp_n then Lc_1 and … and not A_i and … and $Lc_{n'}$ *in KB(C,SH), for each target T in TCF(C,SH), the specification of C contains the following formulae (with* Lp_j *and* Lc_i *literals* not(A), not(A') *for some atoms A and A'):*

– $\bigwedge_{j=1}^{n}\bigwedge_{i=1}^{n'}$((**C**(rule(prem(conj(Lp_1,conj(…,conj(Lp_j,conj(…,Lp_n)…)…))),
conc(conj(Lc_1,conj(…,conj(Lc_i,conj(…,$Lc_{n'}$)…)…)))) ∧
own_task_control_focus(T) ∧
(target(T,A',reject) ∨ target(T,A',determine)) ∧
¬known(A') ∧
¬known(A)))
→ **XC**(required(A,neg)));

- *Furthermore, for each T and E, the specification of C contains the following formulae:*
  – (**Y**¬**C**busy ∧ **C**busy)
    → (own_evaluation(T,any,failed) ∧ ¬own_evaluation(T,any,succeeded) ∧
    own_evaluation(T,any_new,failed) ∧ ¬own_evaluation(T,any_new,succeeded) ∧
    own_evaluation(E,every,failed) ∧ ¬own_evaluation(E,every,succeeded) ∧
    own_evaluation(E,all_p,succeeded) ∧ ¬own_evaluation(E,all_p,failed))
  – ***start*** → **C**own_task_control_focus(TCF_{init}(C,SH))
  – ***start*** → **C**own_extent(EXT_{init}(C,SH))

- ***start*** $\rightarrow$ **C**target(*T,A,Ext*) *for each element* target(*T,A,Ext*) $\in MF_{init}(C,SH)$*, with A an atom and Ext an extent type;*
- ***start*** $\rightarrow$ **C**assumption(*A,S*) *for each element* assumption(*A,S*) $\in MF_{init}(C,SH)$*, with A an atom and S* $\in$ {pos,neg}*;*

If *C* is a control component, then $TCF_{init}(C,SH)$={default_focus}, $EXT_{init}(C,SH)$= {all_p} and $MF_{init}(C,SH)$={target(default_focus,*A*,determine) | *A* $\in$ *TCoutput*}.

Example 9.49. Example 9.7 presented the following knowledge rule as an example rule in a knowledge base:

if communicated_by(M: MATCH, provider_1)
then belief(M: MATCH);

A the specification level, this rule is represented by a *set* of ground atoms as follows:

{ rule(communicated_by(match(*t,q*),provider_1),belief(match(*t,q*))) | *t* $\in OT_2$ *and q* $\in Q$ };

This set is formed by taking all ground atoms for the predicate rule. Suppose *TCH(C,SH)*={default_tcf} and *EC(C,SH)*={default_ec}. In this example, the first rule scheme presented in Definition 9.48 is instantiated for the knowledge rule and the sets of task control foci and evaluation criteria given above. Instantiation of the first rule scheme presented in Definition 9.48 results in the following rule:

$$\bigvee_{t \in OT_2} \bigvee_{q \in Q} (\mathbf{C}(\text{rule(communicated\_by(match}(t,q)\text{,provider\_1),belief(match}(t,q))) \wedge$$
$$\text{true(communicated\_by(match}(t,q)\text{,provider\_1))} \wedge$$
$$\text{own\_task\_control\_focus(default\_tcf)} \wedge$$
$$(\ \text{target(default\_tcf,belief(match}(t,q))\text{,confirm)} \vee$$
$$\text{target(default\_tcf,belief(match}(t,q))\text{,determine))} \wedge$$
$$(\ \text{target(default\_ec,belief(match}(t,q))\text{,confirm)} \vee$$
$$\text{target(default\_ec,belief(match}(t,q))\text{,determine))} \wedge$$
$$\text{own\_extent(any)))}$$
$$\rightarrow \mathbf{XC}(\text{true(belief(match}(t,q))) \wedge$$
$$\text{own\_evaluation(default\_ec,any,succeeded)} \wedge$$
$$\neg\text{own\_evaluation(default\_ec,any,failed));}$$

Quantification over *t* and *q* is at the level of the mathematical presentation employed in this thesis. The formula given above is strictly a (temporal) propositional formula. The rule can be rewritten as follows:

$$(\bigwedge_{t \in OT_2} \bigwedge_{q \in Q} \mathbf{C}(\text{rule(communicated\_by(match}(t,q)\text{,provider\_1),belief(match}(t,q))) \wedge$$
$$\text{true(communicated\_by(match}(t,q)\text{,provider\_1))} \wedge$$
$$\text{own\_task\_control\_focus(default\_tcf)} \wedge$$
$$(\ \text{target(default\_tcf,belief(match}(t,q))\text{,confirm)} \vee$$

$$
\begin{aligned}
& \text{target(default\_tcf,belief(match}(t,q)\text{),determine)} \wedge \\
& (\ \text{target(default\_ec,belief(match}(t,q)\text{),confirm)} \vee \\
& \ \ \text{target(default\_ec,belief(match}(t,q)\text{),determine)} \wedge \\
& \text{own\_extent(any)}) \\
) \quad \rightarrow \mathbf{XC}(& \bigvee_{t \in OT_2} \bigvee_{q \in Q} \text{true(belief(match}(t,q))) \wedge \\
& \text{own\_evaluation(default\_ec,any,succeeded)} \wedge \\
& \neg\text{own\_evaluation(default\_ec,any,failed)});
\end{aligned}
$$

This formula captures the meaning of own_extent(any) as follows. Suppose that for a specific state in a local component trace, communicated_by(match(*t*,*q*),provider_1) is true for all *t* and *q*. To satisfy the rule presented above, in the next state belief(match(*t*,*q*)) only has to be true for one *t* and *q*. ■

The previous definition and example describe the relation between task control information and the chaining process executed for DESIRE components with which a knowledge base is associated. For a precise description of the behaviour of such components in DESIRE, more detail has to be added to Definition 9.48, for instance for truth maintainance and persistency. (See Gavrila and Treur, 1994, for a formal description of truth maintainance in DESIRE. See Engelfriet, 1999, for a formal description of persistency using temporal completion.)

9.4 Discussion

In this chapter, the dynamics of DESIRE are described in terms of the semantic structure developed in this thesis: a DESIRE model is represented by a compositional system that is described using the semantic structure. As a consequence, the dynamics of DESIRE shares all properties committed to for the semantic structure as described in Chapter 2 and Chapter 8 (Section 8.2). A number of these properties are discussed again in this section to describe their ramifications in terms of DESIRE. At the end of this section, some general conclusions with respect to DESIRE and the application of the semantic structure are drawn.

9.4.1 Control and Autonomy

The most important feature of an agent is its autonomy. Section 8.3.1 indicated that, at first sight, control over agents is impossible as such control is in conflict with an agent's autonomy. However, as stated in Section 8.3.1, control over an agent should be viewed as an attempt to influence the agent such that it adopts goals wanted by the controlling agent, without any guarantee that the agent will comply.

As stated before in this chapter, in a DESIRE model, all agents are represented by (often composed) components that are subcomponents of the toplevel component. The toplevel component is a normal composed component, which

implies that, as for all composed components, task control knowledge is associated with the toplevel component. As a consequence, the behaviour of agents can be directly controlled by the task control of the toplevel component.

In Section 8.3.1, two options for control over subagents were discussed. The option chosen in DESIRE (at some levels, control is possible, at others, it is not: see Section 3.3.1) implies that Task control knowledge for the toplevel component is, similar to other composed components in DESIRE, expressed in the form of knowledge rules for *TCinput* and *TCouput*. With these signatures, control can be specified at great detail. It is possible to directly state which tasks should be performed (using task control foci) and to what extent. Moreover, the semantics of DESIRE ensures that the controlled components (the agents) will indeed behave as specified by the task control knowledge of the toplevel component. This level of control is clearly in conflict with the autonomy of the agents. In fact, task control of the toplevel component introduces a new agent in the multi-agent system. This agent is not present in the multi-agent system that is modelled, but is an artefact of the DESIRE modelling framework. The presence of the toplevel task control (which is ubiquitous in the framework DESIRE), if used with all its possibilities, makes all 'real' agents subordinate to a 'supervising agent' that itself cannot be directly influenced and has almost divine power over the agents it controls. Thus, it becomes impossible to model situations in which the existence of such a deity is denied.

To avoid the unintended introduction of a supervising agent, users of the DESIRE framework constrain the toplevel task control structure: they only use rules of the form:

if **start**
then next_component_state(A_1,awake) **and** … **and** next_component_state(A_n,awake)
and next_link_state(I_1,awake) **and** … **and** next_link_state(I_m,awake);

Thus, all agents A_1 to A_n and all information links I_1 to I_m are made awake, and no more control is specified. With this solution, the toplevel's only role is to activate the agents, which it cannot control any further. In a sense, the toplevel task control becomes superfluous: if it is assumed that agents exist at all times, or if another means of creating them is introduced, there is no longer any function for the toplevel task control.

9.4.2 Link Granularity

The semantic structure describes information transmission as a relation between states, as is indicated by the definition of information link mappings in Chapter 5. More precisely, the semantic structure describes information transmission as a relation between the output substate of the domain of an information link and the input substate of the co-domain of the link. In [[[Chapter 8]]], input and output substates are further divided in control and domain parts. These parts do not have

any internal structure. As a consequence, it is not possible to describe information transmission of a 'subpart' of the domain or control part of a substate.

In this chapter, states are described by propositions that are true in these states. Moreover, information transmission is described by information link mapping descriptions, which are sets of octets of propositions that describe the input and output substates related by information transmission. These sets of octets may suggest that substates have internal structure beyond the division in a control part and a domain part, i.e., that a state *consists* of propositional symbols that are transmitted one by one. Indeed, experience with DESIRE has shown that users tend to view information transmission in this way. As an aside, in prototypes generated by the current implementation generator, component states are indeed represented by sets of propositional symbols.

If states are viewed as sets of propositional symbols and information transmission as the sequential transmission of these symbols, then such transmission is atomic, or, in other words, indivisible in the semantics of DESIRE. This is automatically enforced by the semantic structure: these sets of symbols describe one specific (domain or control part of a) substate, which is linked to one specific other (domain or control part of a) substate. Indeed, it is not even possible, using the semantic structure, to speficy partial transmission of such sets of symbols. In other words, no temporal logic specification formulae or properties of compatibility relations have to be given to constrain information transmission such that sets of symbols are transmitted atomically.

Although information transmission is atomic, in implementations it generally takes time to transmit information over information links, which, moreover, takes place simultaneously with other activities in a multi-agent system. Consequently, implementations have to ensure that link activations are guaranteed to run without interference of other agent activities.

Alternative options for the granularity of indivisible information transmission can be distinguished:

- While the input interface of a specific component is being modified by an information transmission, subcomponents of that component have access to the interface;
- Information transmission for one link activation is atomic. However, link sequence activations are not atomic.

With the first option, no indivisible part in a link activation is distinguished. This option has an enormous impact on the specification, e.g., of an agent. If it is possible to use truth values that are being updated at that moment, chances are that sets of values which describe both old facts and new are encountered. The agent may not know this is the case, so it may interpret this set as valid information. However, this information does not correspond to any real world state. So, the agent has to be able to reason about states which in fact may not occur

in the real world. This is clearly undesirable. The second option defines the size of the indivisible part of information transmission to be a single link activation.

9.4.3 Concluding Remarks with respect to DESIRE

As stated in Chapter 1, the aim of this thesis is to develop a formal, compositional, semantic structure for multi-agent systems dynamics. In this chapter, the semantic structure is applied to describe the dynamics of models of multi-agent systems modelled using the DESIRE modelling framework. The design of the DESIRE modelling framework itself is not part of this thesis. However, some concluding remarks with respect to DESIRE are presented to support the general assumption adopted in this thesis: multi-agent systems are represented as compositional systems.

- Extensive experience with DESIRE shows that it is possible to design multi-agent systems as compositional systems. Examples are presented in various papers, including (Brazier, Dunin-Keplicz, Jennings & Treur, 1997; Brazier, Jonker, Jungen & Treur, 1999; Jonker, Lam & Treur, 1999). An extensive overview of papers presenting applications of DESIRE can be found at the World Wide Web[2];
- In DESIRE, the functionality of agents is specified (as knowledge) in a declarative, implementation-independent specification language (that resembles natural language). As a result, extension and/or modification of parts of specifications is straightforward. The example DESIRE models of multi-agent systems presented in Chapter 10 and Chapter 11 further illustrate this point;
- Not only numerical formalisms (such as, e.g., in the mathematical toolkit described in (Gaylford & D'Andria, 1998)), but also logical and knowledge based formalisms can be used to (declaratively) specify the often knowledge-intensive functionality of agents. This characteristic distinguishes a the DESIRE modelling framework from the more conventional system development methods and modelling frameworks, such as the UML (Booch, Rumbaugh & Jacobson, 1998). In addition to the language within which knowledge is represented, the compositional structures within DESIRE provide a means to group functionality;
- A graphical design support environment is available and executable prototype systems can be generated automatically from a detailed design.

The main contribution of the application of the semantic structure presented in this chapter is the use of temporal logic to specify sets of local component and link traces, which are assumed to be given in previous chapters. Control within

[2] http://www.cs.vu.nl/~treur/

DESIRE models is also specified using temporal logic (in the form of knowledge base rules for the signatures *TCinput* and *TCoutput*). The use of temporal logic for the specification of control is not new. Important examples are Temporal Logic of Actions (TLA; Lamport, 1994), Troll (Jungclaus, Saake, Hartmann & Sernadas, 1996, the semantics of which is defined as a translation to OSL (Sernadas, Sernadas & Costa, 1995), discussed in Chapter 12) and Concurrent MetateM (Fisher & Wooldridge, 1997, also discussed in Chapter 12). However, the combination of temporal logic as a specification language for control together with the principles of compositionality and locality as presented in the previous chapters is to the best of the author's knowledge, new.

Locality and compositionality are the key features of the semantic structure employed in this chapter to describe the dynamics of DESIRE models. Locality enables the description of the behaviour of single components in isolation, which is necessary to cope with the complexity of a detailed description of the behaviour of a flexible, real-world framework such as DESIRE. Compositionality is needed to put such local descriptions in their overall context.

Chapter 10

Example: Exclusive Access

In this chapter, the use of DESIRE is illustrated for competitive interaction, such as the interaction required to access limited resources. A reusable generic model for a competitive agent is presented together with a model of interaction between such agents. The generic model for competitive interaction introduced in this chapter, is based on analysis of competitive agent in knowledge-intensive situations. An example of a knowledge intensive situation is the situation in which a number of information agents can access a given information agent and explicit knowledge is available (to either the accessing agents, or the agent to be accessed, or all agents involved) on appropriate orderings or priorities between the transactions from the different agents. In this chapter, an example of a knowledge intensive situation is described and used to illustrate the types of knowledge used to instantiate a generic model of a competitive agent.

This chapter is outlined as follows. In Section 10.1, competitive co-ordination is introduced and sketched for the example domain of human resource allocation. Section 10.3 shows how a formal model for limited resource acquisition has been modelled and specified in DESIRE, illustrated for the example presented in Section 10.1. In Section 10.3, an algorithmic approach to limited resource access in the domain of operating systems is presented based on algorithms for mutual exclusion (see for example (Ricart & Agrawala, 1981). Such algorithms describe a solution to a specific type of interaction for situations in which a number of additional assumptions can be made. These assumptions are included in the discussion in Section 10.4. Discussion and further research, as well as the relation with the semantic structure, are presented in Section 10.4. A preliminary version of this chapter has been published as (Brazier, Eck & Treur, 1997b).

10.1 Competitive Agents

A common phenomenon in many real life situations is the phenomenon of limited availability of resources. Resources can be tangible such as books in libraries, seats in aircraft, frequencies for transmission, articles produced by manufacturers, but also time and space are often resources upon which restrictions are placed. To model situations in which limited access to resources plays a role, not only the

resources, but also the participants are of importance. Participants, in general, have different goals, needs and ambitions, but also vary with respect to the degree with which they are willing to acknowledge other participants' needs, and to act upon them. The degree to which participants are willing to co-operate often depends on the situation within which they are placed.

In everyday practice many situations are solved by 'one' of the participants (or class of participants): in a library the librarian decides in which sequence requests are granted, airlines decide which categories of clients are served first, etc. Central co-ordinating participants often strive for collective user satisfaction: they try to achieve as much satisfaction as possible (according to some measure) for as many users as possible (see also (Brazier & Ruttkay, 1993)).

In other situations, however, participants must come to mutual agreement on the designation of resources. For example, allocation of space is one of the important aspects in complex engineering design, for instance of industrial products. Robot co-ordination is another example. The example used in this chapter to illustrate how agent co-operation of this type can be formally modelled and specified in DESIRE, is related to human resource allocation. In many companies, open applications are considered once every n months. Applicants are informed about the organisation and about the positions the company offers. Applicants are given the opportunity to acquire more information about relevant positions and to indicate in which positions they would be most interested. The opposite holds for the heads of departments with vacancies: the managers. They acquire more information about the candidates and decide whom they would be most interested in offering a position. In an 'ideal' situation the candidates' choices would be mutually exclusive and would match the preferences of the managers. This is, however, not frequently the case. Most often the managers negotiate how the limited resources (the applicants) are to be allocated. The process is often not predictable: the managers must come to some agreement, on the basis of the information available. Not only the individual manager's interests play a role, but also the interests of the organisation. Career perspectives of very well qualified and promising candidates may be considered of more importance in allocating a position for these candidates than a strong 'local' need of a specific department. Factors such as how long a position has been open, how qualified an applicant is, how well an applicant is perceived to fit into a particular group, etc., all play a role. Within an organisation not all managers (c.q. departments) are assigned equal 'rights'. In practice this means that some managers will have more right to a specific applicant than other managers. Managers differ in their interpretation of the factors involved, but they also vary in the degree to which they are willing to co-operate with other managers.

In situations in which one manager clearly has more right to an applicant than another, this, in general, will be acknowledged. The manager will most often wait for approval from all other potentially interested managers before offering the applicant a position, to minimise the chance that an applicant receives conflicting

offers. In situations in which managers have equal rights, once all objective criteria have been examined, the managers' characters may be the decisive factor. A more assertive manager may be more convincing than a less assertive manager, in which case the assertive manager may be more effective. The more assertive manager will also need approval from all potentially interested managers before offering an applicant a position. If all interested managers are equally convincing, a deadlock situation occurs. In such situations the personnel manager often decides which department is granted the right to offer an applicant a position.

10.2 A Model of a Competitive Agent

In this section, the generic agent model introduced in Chapter 3 is refined to support the design of a multi-agent system that models competitive agents. The three types of knowledge described in Section 9.1 are illustrated for the domain described in Section 10.1. For the purpose of explanation, process composition and knowledge structures are described together, followed by information exchange, control knowledge and the relation between agents and processes.

10.2.1 Process Composition and Knowledge Structures

In Figure 10.1, the process composition for this example is depicted. The six top level components are described below in Section 10.2.1.1 to Section 10.2.1.6 together with the knowledge structures involved (input atoms, output atoms and knowledge bases).

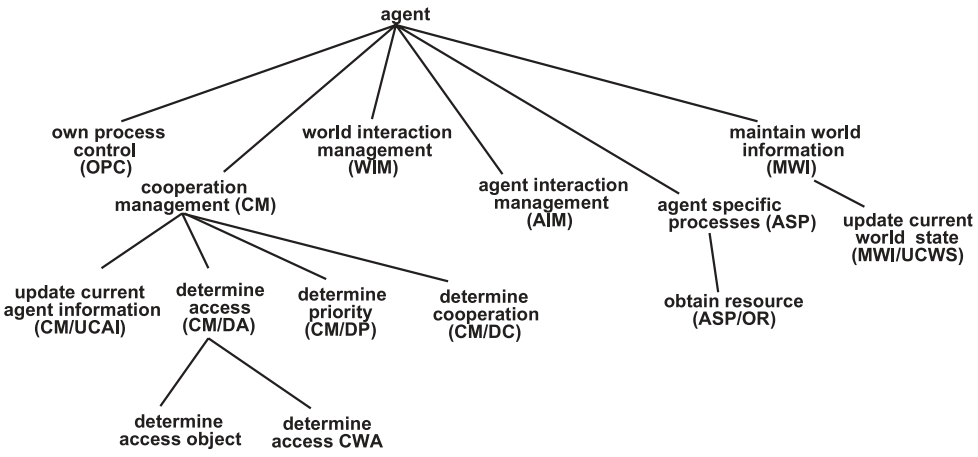

Figure 10.1: Composition structure of the specialised task model.

10.2.1.1 Maintain World Information (MWI)

The only component modelled for the process Maintain World Information is the component update_current_world_state:

Component: update_current_world_state (MWI/UCWS)

The component update_current_world_state stores the information the agent has about the world state. This information can be acquired by observation of the world, communication with other agents, or (defeasible) reasoning. For resource access competition, only information on the presence of agents and resources is maintained. This component only maintains information. Therefore, its knowledge base is empty.

Input atoms	from comp.	Output atoms	to comp.
agent_present(A: Agents)	(from world)	agent_present(A: Agents)	CM/DA
resource_present	(from world)	resource_present	OPC

10.2.1.2 Agent Specific Processes (ASP)

The most relevant process specific for a particular agent for which limited access of resources is modelled in this example is represented by the component obtain_resource. This process is characterised for a specific agent ('self') in the specifications of the related components below. Other subprocesses of agent_specific_tasks are responsible for determining whether access to a specific resource is required for the agent. These specific tasks are not specified in this chapter.

Component: obtain_resource (ASP/OR)

The component obtain_resource has the following input and output atoms, which are transmitted as indicated by the table below:

Input atoms	from comp.	Output atoms	to comp.
action_performed(communicate_need)	WIM	proposed_action(communicate_need)	WIM
action_performed(observe_presence_agents)	WIM	proposed_action(observe_presence_agents)	WIM
action_performed(take_resource)	WIM	proposed_action(take_resource)	WIM
access_to_be_forced	CM	query(access_allowed)	CM
is_known(access_allowed)	CM	resource_obtained	OPC

Internal atoms: to_be_obtained(access_allowed)

The knowledge base of the component obtain_resource specifies the knowledge required to propose new actions on the basis of the input:

if **not** is_known(access_allowed)

then to_be_obtained(access_allowed);

if to_be_obtained(access_allowed)
and not action_performed(communicate_need)
then proposed_action(communicate_need);

if to_be_obtained(access_allowed)
and not action_performed(observe_presence_agents)
then proposed_action(observe_presence_agents);

if access_allowed
and not action_performed(take_resource)
then proposed_action(take_resource)
and resource_obtained;

if access_to_be_forced
then proposed_action(take_resource)
and resource_obtained;

10.2.1.3 Cooperation Management (CM)

Within the process of Cooperation Management, four subprocesses are distinguished, namely Update Current Agent Information, Determine Access, Determine Priority and Determine Cooperation. These subprocesses appear as subcomponents of the component cooperation_management in the specification.

Component: update_current_agent_information (CM/UCAI)

The component update_current_agent_information stores the information the agent has about other agents. This information is usually obtained by communication or (defeasible) reasoning. This component has the following input and output atoms. (The knowledge base of this component is empty, as the component is only used as an information store.)

Input atoms	from comp.	Output atoms	to comp.
irrelevant(A: Agents)	(defeasible)	irrelevant(A: Agents)	CM/DA
wants_resource(A: Agents)	(communication)	wants_resource(A: Agents)	CM/DA
to_leave_resource_for(A: Agents, self)	(communication)	to_leave_resource_for(A: Agents, self)	CM/DA

Component: determine_access (CM/DA)

The component determine_access receives input facts about the world; e.g., obtained by observations, communications, or (default or closed world) assumptions. Moreover, it uses input about priorities between agents from the component determine_priority and information about the co-operativeness of other agents from the component determine_cooperation. The component determine_access analyses a world state (no matter how it was reached) and draws conclusions about access to the resource: which agent(s) should be allowed access to a resource and which should not. The component has the following input and output atoms:

Input atoms	from comp.	Output atoms	to comp.
agent_present(A: Agents)	MWI/UCWS	access_allowed	ASP/OR
irrelevant(A: Agents)	CM/UCAI	to_leave_resource_for(self, A: Agents)	CM/DC
wants_resource(A: Agents)	CM/UCAI		
to_leave_resource_for(A: Agents, self)	CM/UCAI		
has_priority_over(A: Agents, B: Agents)	CM/DP		
cooperative(A: Agents)	CM/DC		

Most of the knowledge in the knowledge base of determine_access specifies under which conditions access to the resource is not granted. The first two rules below determine whether there is a conflict in the sense that another agent and the agent itself are both interested in accessing the resource. Note that the knowledge specified in this knowledge base does not refer directly to the application domain described in Section 10.1. It is generic in the sense that it can in principle be used for all domains in which limited access to resources plays a role.

if agent_present(A: Agents)
and not irrelevant(A: Agents)
then relevant(A: Agents);

if relevant(A: Agents)
and wants_resource(A: Agents)
and wants_resource(self)
then conflicting_needs(A: Agents, self);

In some situations, the conditions in which the other agent (*A*) blocks access to the resource for the agent itself are known:

if conflicting_needs(A: Agents, self)
and has_priority_over(A: Agents, self)
then access_blocked_by(A: Agents);

The consequence of the other agent blocking the resource is that the agent itself cannot access the resource and leaves the resource for *A*. This is expressed by the following rule:

if access_blocked_by(A: Agents)
then **not** access_allowed
and to_leave_resource_for(self, A: Agents);

If agent self knows that the other agent *A* is co-operative, but that agent *A* has not let agent self know that agent self may access the resource, agent self is assumed to not be allowed access to the resource. (This rule is discussed in more detail at the end of the description of this component).

if conflicting_needs(A: Agents, self)
and cooperative(A: Agents)
and not to_leave_resource_for(A: Agents, self)

then **not** access_allowed;

Priority for agent 'self' with respect to the other agent can not always be determined. In this case, the determine_access component concludes that it cannot derive whether access is allowed and defers this decision to the component determine_cooperation.

if conflicting_needs(A: Agents, self)
and not has_priority_over(A: Agents, self)
and not has_priority_over(self, A: Agents)
then no_access_decision;

In situations in which nothing is known about allowed access to the resources, the conclusion (by assumption) can be drawn that access is allowed by employing a closed world assumption (CWA). The CWA is modelled using explicit meta-knowledge in the separate meta-component determine_access_CWA depicted in Figure 10.1:

if **not** false(access_allowed)
then to_assume(access_allowed, positive);

This closed world assumption expresses the assumption that all relevant agents have been observed (if one agent is overlooked, the CWA leads to an incorrect conclusion). Another closed world assumption is made on the basis of communication. This closed world assumption expresses that if no information is received from a co-operative colleague with respect to resource allocation to the agent itself then it is safe to assume that the resource will not be made available to itself.

if **not** true(to_leave_resource_for(A: Agents, self))
then to_assume(to_leave_resource_for(A: Agents, self), negative);

The conclusion drawn by this CWA may be incorrect if communication problems occur.

Component: determine_priority (CM/DP)

The task of the component determine_priority is to determine which of two agents may access the resource first, if a conflict is detected by evaluating the world state. As stated before, the knowledge in this component is domain-specific. The instantiation of determine_priority presented contains knowledge as outlined in Section 10.1. It is important to note that this instantiation of the component is not always able to derive a conclusion: there may be circumstances in which no priority can be assigned. The component determine_priority has the following input and output atoms:

Input atoms	from comp.	Output atoms	to comp.
		has_priority_over(A1: Agents, A2: Agents)	

The knowledge base of component determine_priority contains the following rules:

```
if      manager_rights(M1: Agents, low)
        and manager_rights(M2: Agents, high)
then    has_priority_due_to_manager_rights(M2: Agents, M1: Agents);

if      vacancy_duration(P1: Positions, short)
        and manager_of(M1: Agents, P1: Positions)
        and vacancy_duration(P2: Positions, long)
        and manager_of(M2: Agents, P2: Positions)
then    has_priority_due_to_vacancy_duration(M2: Agents, M1: Agents);

if      suitable(A: Agents, P1: Positions, low)
        and manager_of(M1: Agents, P1: Positions)
        and suitable(A: Agents, P2: Positions, high)
        and manager_of(M2: Agents, P2: Positions)
then    has_priority_due_to_suitability(M2: Agents, M1: Agents);

if      has_priority_due_to_suitability(M1: Agents, M2: Agents)
then    has_priority_over(M1: Agents, M2: Agents);

if      not has_priority_due_to_suitability(M2: Agents, M1: Agents)
        and has_priority_due_to_vacancy_duration(M1: Agents, M2: Agents)
then    has_priority_over(M1: Agents, M2: Agents);

if      not has_priority_due_to_suitability(M2: Agents, M1: Agents)
        and not has_priority_due_to_vacancy_duration(M2: Agents, M1: Agents)
        and has_priority_due_to_manager_rights(M1: Agents, M2: Agents)
then    has_priority_over(M1: Agents, M2: Agents);

if      has_priority_over(M1: Agents, M2: Agents)
then    not has_priority_over(M2: Agents, M1: Agents);
```

Component: determine_cooperation (CM/DC)

The component determine_cooperation contains facts describing the agent's knowledge of which agents are willing to co-operate with which other agents. In general, these are instances (or negated instances) of the atom is_cooperative_for(A: Agents, B: Agents). Note that the truth values for this atom are the result of observations and related conclusions. Below, in the knowledge base, three options for *static* facts about the agent self are specified: one for a shy agent, one for a bold agent and one for a moderate agent. The component determine_cooperation has the following input and output atoms and knowledge base:

Input atoms	from comp	Output atoms	to comp
is_cooperative_for(A, B: Agents)	WIM	is_cooperative(A: Agents)	CM/DA
concluded(to_leave_resource_for(self, A: Agents)	CM/DA	to_communicate_to(A: Agents, ok)	AIM
no_access_decision	CM/DA	force_access	AST/OR

Knowledge base:

```
bold(self);                 /* for the bold agent */
```

```
modest(self);              /* for the modest agent */
moderate(self);            /* for the moderate agent */

if      concluded(to_leave_resource_for(self, A: Agents))
        and is_cooperative_for(self, A: Agents)
then    to_communicate_to(A: Agents, ok);

if      concluded(no_access_decision)
        and bold(self)
then    access_to_be_forced;

if      concluded(no_access_decision)
        and modest(self)
then    to_communicate_to(A: Agents, ok);

if      not is_cooperative_for(A: Agents, B: Agents)
then    not is_cooperative(A: Agents);
```

By the closed world assumption an agent that is not known to be uncooperative is assumed to be co-operative (the positive view on agenthood); the following meta-knowledge is used:

```
if      not false(cooperative(A: Agents))
then    to_assume(cooperative(A: Agents),positive);
```

10.2.1.4 World Interaction Management (WIM)

The subprocesses of world_interaction_management are (1) to perform observations (including observation of the presence and relevance of other agents) and (2) to perform the action proposed by the task of obtaining a resource. The precise specification of these subprocesses is not relevant for mutually exclusive resource access. Moreover, the precise specification of these processes depends on details of the external world component in the multi-agent systems. Specifications of a component designed for a similar subprocesses in another domain can be found in (Brazier, Dunin-Keplicz, Jennings & Treur, 1997).

10.2.1.5 Agent Interaction Management (AIM)

The component agent_interaction_management manages communication between an agent and other agents. Below, in addition to the input and output atoms, a relevant part of the knowledge base is specified.

Input atoms	from	Output atoms	to
Communicated(wants_resource(A: Agents))	(communication)	query(next_communication(A: Agents, ok))	OPC
next_communication(A: Agents, ok)	OPC	communicate(A: Agents, ok)	(communication)

Knowledge base:

if communicated(wants_resource(A: Agents))
and next_communication(A: Agents, ok)
then communicate(A: Agents, ok);

Further details of how communication takes place is not relevant for mutually exclusive resource access, and depends on details of the agents not provided in Section 10.1. Therefore, a complete specification of the knowledge base of this component has been omitted. For examples of specifications of similar tasks in another domain: see e.g. (Brazier, Dunin-Keplicz, Jennings & Treur, 1997).

10.2.1.6 Own Process Control (OPC)

The role of own_process_control is to determine which information is needed to decide whether access to a resource is allowed, and where this information is to be found. The input, output and knowledge base for the component own_process_control is as follows:

Input atoms	from	Output atoms	to
to_leave_resource_for(A: Agents, B: Agents)	CM	decided(resource_assigned)	ASP/OR
only_agent_present(self)	MWI	decided(next_communication(A: Agents,ok))	AIM
resource_present	MWI	query(resource_obtained)	ASP
determined(holding_resource, S: Signs)	ASP/OR		
info_needed(to_communicate_to(A: Agents, ok))	AIM		
info_needed(resource_assigned)	ASP/OR		

Knowledge base:

if info_needed(to_communicate_to(A: Agents, ok))
and determined(holding_resource, pos)
then decided(next_communication(A: Agents, ok), neg);

if info_needed(to_communicate_to(A: Agents, ok))
and determined(holding_resource, neg)
and determined(to_leave_resource_for(self, A: Agents))
then decided(next_communication(A: Agents, ok), pos);

if info_needed(to_communicate_to(A: Agents, ok))
and determined(holding_resource, neg)
and not determined(to_leave_resource_for(self, A: Agents))
and determined(to_communicate_to(A: Agents, ok)
then decided(next_communication(A: Agents, ok), pos);

if info_needed(resource_assigned)
and only_agent_present(self)
and resource_present
then decided(resource_assigned, pos);

```
if      info_needed(resource_assigned)
        and not only_agent_present(self)
        and resource_present
        and determined(to_leave_resource_for(A: Agents, self)
        and resource_obtained
then    decided(resource_assigned, pos);
```

10.2.2 Control Knowledge

Knowledge of the sequencing of processes is represented as control knowledge. The component own_process_control determines which information is needed to decide whether access to a resource is allowed, and where this information is to be found. Control knowledge is used to evaluate the conclusions drawn by the component own_process_control. The component own_process_control is activated in one of the following ways:

- The component agent_interaction_management notices that one of the other agents requests attention, for example to access a resource. Component agent_interaction_management notifies the component own_process_control of the fact that new information has been received. On the basis of this new information, the component own_process_control decides to activate the component cooperation_management to determine whether access is granted (the truth value of the atom to_leave_resource_for(self, A)).
- To perform one of its own specific tasks, an agent determines a need for information. The component agent_specific_tasks expresses this need to the component own_process_control, which decides which specific information is needed and where it is to be found. If the component own_process_control recognises the need to access a limited resource, it first activates the component cooperation_management to determine whether access is allowed or not (the truth value of the atom access_allowed). If access is allowed, the component own_process_control activates the component obtain_resource.

10.2.3 Relation between Processes and Agents

In the preceding sections, resource access is modelled on the basis of a top-level composition (generic agent model) as given in Chapter 3 for one specific agent. In this section, the process of mutually exclusive access acquisition described in Section 10.1 is modelled as a whole with respect to the delegation of subprocesses to agents. Three managers, called 'manager 1', 'manager 2', and 'manager 3' are modelled as agents with different characteristics (bold, modest, moderate). These agents, depicted in Figure 10.2 below as agent_1, agent_2 and agent_3, each correspond to specialisations (further decomposition) and instantiations (definition of (domain) specific instances of knowledge structures) of the agent model described by the composition depicted in Figure 10.1, with slightly different co-operation knowledge reflecting their characteristics.

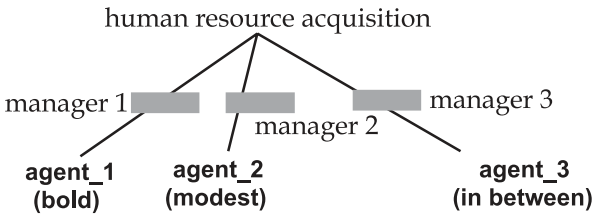

Figure 10.2: Relation between processes and agents.

The specification of the relation between processes and agents concludes the description of the example DESIRE model. In the rest of this chapter, the example DESIRE model is first compared with an algorithmic approach for mutually exclusive access acquisition. The relation between the example presented in this chapter and the semantic structure developed in this thesis is discussed in the final section of this chapter.

10.3 Comparison with an Algorithmic Approach

The specification of the mutually exclusive access process presented in this chapter can be compared to approaches to mutual exclusion in more conventional environments, such as the algorithm described by Ricart and Agrawala (1981). Ricart and Agrawala's algorithm assumes the following conditions hold: (1) each agent notices each other agent's presence; (2) communication never fails; (3) each agent has the same *complete* knowledge of priorities between agents and (4) if agent *A* has higher priority than agent *B*, agent *B* is assumed to communicate that it grants access to the resource to agent *A*.

If these conditions hold, the model described in this chapter specifies the same process as the algorithm. The conditions describe a rather strictly defined domain of application, as found in, for example, the domain of operating systems. For less strictly defined real world domains, however, incompleteness of observations, defeasible communications, incomplete knowledge of priorities, inconsistencies between conclusions drawn by different agents, uncooperative agents, etc. are most common. Modelling practise in these domains reveals the strengths of the compositional approach presented in this chapter. A compositional approach is particularly suitable in these domains for the following reasons: (1) by virtue of the reflective structure of the task model, different strategies can be modelled with minimal effort. (2) assumptions with respect to for instance communication, priorities and co-operation appear explicitly in the task model and (3) the different types of knowledge are explicitly distinguished. The distinction between different types of knowledge and different types of behaviour results in flexibility, adaptability and transparency: essential characteristics of a knowledge-based approach.

10.4 Discussion

This chapter presented, for the purpose of illustrating the semantic structure developed in this thesis, a generic DESIRE model for mutually exclusive resource access and instantiated this model for the domain of human resource management. Chapter 9 explains how the behaviour of such a DESIRE model can be represented using the semantic structure. Before the behaviour of the specific model presented in this chapter is discussed, first some remarks with respect to the purpose of using DESIRE and the semantic structure are made:

- A modelling framework such as DESIRE can be used for purposes of knowledge management, understanding, and design in the early stages of a software engineering process. The use of a formal modelling framework helps in achieving a clear picture of the software system that is to be created. The advantages are well known. Compared to conventional application areas, in the area of multi-agent systems it is often not possible to make very strict assumptions about the environment of agents. (Such as the assumptions made by Ricart and Agrawala (1981), as presented in the previous section.) Instead, knowledge is used more intensively, first because the environment is more complex, and second to react to unexpected events in the environment. (As an example of the first more intensive use of knowledge, in this chapter resource access is granted first on the basis of (given) priorities, and (if no decision can be taken) on the basis of dynamically observed co-operativeness of other agents.) However, the modularity of the model, in which both static and dynamic behaviour are explicitly specified, allows for flexible adaptability to other strategies. The DESIRE modelling framework has extensive support for such knowledge intensive applications. The semantic structure facilitates this use of a modelling framework by supplying a precisely defined domain for the interpretation for specifications of the system that is designed. Such an interpretation may help to further understand complex and important details of the system;

- A modelling framework can also be used in a rapid prototyping approach to software design. In this case, in the early stages of a software engineering process, experiments are carried out with a prototype implementation of the system that is to be designed to evaluate the results of the design process so far. The DESIRE modelling framework supports this approach by its automatic prototype generator. The semantic structure presented in this thesis is not directly used. However, it is used in the design and evaluation of the prototype generator;

- Finally, a use specific for *formal* modelling frameworks is evaluation of properties of a multi-agent system. A formal specification of a multi-agent system enables proving that specific safety properties hold. If (the

specification language in) the modelling framework is equipped with a logic, this can be done formally and probably even computer-assisted or automatically. However, it is also possible to evaluate (safety) properties semantically, i.e., at the level of the interpretation of the model in a specific semantic structure. As the semantic structure defined in this thesis is mathematically defined, a rigorous, mathematical proof can be developed. If the semantic structure is *itself* equipped with a logic that enables proving properties of constructs in the interpretation of a model, (safety) properties of the multi-agent system can be formally proved, probably even computer-assisted or automatically.

For the model presented in this chapter, all three purposes apply. In the rest of this chapter, the model presented in this chapter is evaluated and the use of the semantic structure in such an evaluation is discussed. Emphasis is put on the third purpose, the evaluation of properties of the model.

To start, the context of the model is further elaborated. As stated at the beginning of this chapter, mutually exclusive access is a common phenomenon in many multi-agent systems. In general, there are a number of agents that are willing to access a specific resource. (An applicant in the example domain used in this chapter). The resource to which mutually exclusive access is wanted may be an agent, or it may be a part of the external world. In the first case, actually accessing the resource consists of information exchange with this agent. In the second case, actually accessing the resource consists of executing specific actions in the external world. In both cases, the agents that seek mutually exclusive access as well as the (agent that is) the resource represented as subcomponents of a component called toplevel, which is itself not a subcomponent of any other component. In Figure 10.3, the context is illustrated in DESIRE for two agents, *A* and *B*, which try to access a resource represented by an agent or the external world. Both agents are assumed to be instances of the generic model presented in this chapter.

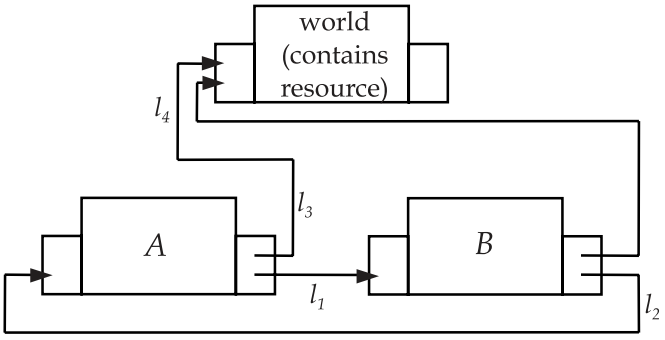

Figure 10.3: Simple representation of resource access.

In the detailed model presented in Section 10.2.1, the subprocess that actually accesses the resource is not specified. However, it is assumed that access of the resource starts with executing an action called obtain_resource in the external world. Execution of this action is represented in the model by the occurrence of an atom action_to_be_performed(obtain_resource) in the output interface of the agent that executes the action. (It is placed there via an export mediating link from the output interface of the World Interaction Management component. After the resource is obtained, additional actions may be executed. If an agent is done with the resource, an action called give_back_resource is executed, which is represented by the occurrence of an atom action_to_be_performed(give_back_resource) in the output interface of the agent that executes the action.

As explained in Chapter 9, the DESIRE model presented in Figure 10.3 is represented in the semantic structure by a DESIRE structure hierarchy with control. This control structure contains the components presented in this chapter as well as additional control components that correspond to the composed DESIRE components. In this chapter, for all primitive components, knowledge bases are given. Chapter 9 explains how, using these knowledge bases, sets $Beh_{loc}(C)$ for the primitive components can be found. Together with the standard DESIRE compatibility relations and the control structure that represents the DESIRE model in the semantic structure, the sets $Beh_{loc}(C)$ are used to determine the three views on the behaviour of the model. For the evaluation of the model for mutually exclusive access presented in this chapter, the white box view on the toplevel component is the most appropriate view, as it contains the behaviour of both agents and the resource component in their context, without unnecessary details on the behaviour of the subcomponents of the agents. (The white box view on the behaviour of the toplevel component is a set of compatible multitraces. Each multitrace consists of local traces for the toplevel component, both agents, the component in which the resource resides, and the links between the agents and this component.)

The stage is now set to discuss the evaluation of the model presented in this chapter and the role of the semantic structure in this evaluation. The most evident question to ask in the evaluation of the model is whether exclusive access can be guaranteed. First of all, in a multi-agent systems, it may not be possible to provide an unqualified 'yes' or 'no' for this question (as is assumed in conventional approaches, where (explicitly or implicitly), strict assumptions are made, cf. the approach presented by Ricart and Agrawala (1981)). For instance, the domain knowledge presented for the domain of human resource management presented in this chapter is not complete, as is often the case in real-world environment: priority relations between managers are not always established. Nevertheless, evaluation of the model is centred on the question whether exclusive access is guaranteed (under specific assumptions).

As explained above, it is possible to obtain the white box view on the behaviour of the toplevel component in Figure 10.3. Elements of the white box view are

compatible multitraces that consist of local component and link traces for components *A*, *B*, the world, the toplevel component that represents the complete multi-agent system, and links between these components. An element of the white box view on the behaviour of the toplevel component is depicted in Figure 10.4. Local link traces for the links between these components are not depicted in Figure 10.4. In Figure 10.4, boxes denote local component states. Horizontal, solid arrows between boxes denote state transitions. Diagonal, dashed arrows denote information transmission. A number of the input and output atoms for most states are also depicted in Figure 10.4. Atoms at the left side of the small vertical bars are input atoms, atoms at the right side are output atoms. Atoms without a bar are internal atoms (e.g., priority(self,B)). The names of input and output atoms are abbreviated as follows: to_be_c_to(…,…) for to_be_communicated_to(…,…), comm'ed_by for communicated_by, a_to_be_perf for action_to_be_performed, and res for resource. In the example element, both *A* and *B* try to obtain access to a resource. Upon receipt of information from the other agent that access is sought, both agent *A* and *B* determine that *A* has priority over *B*. Therefore, *B* grants access to *A*. Consequently, *A* takes the resource (an action in the world), and releases the resource some time later. Finally, agent *A* again tries to obtain the resource.

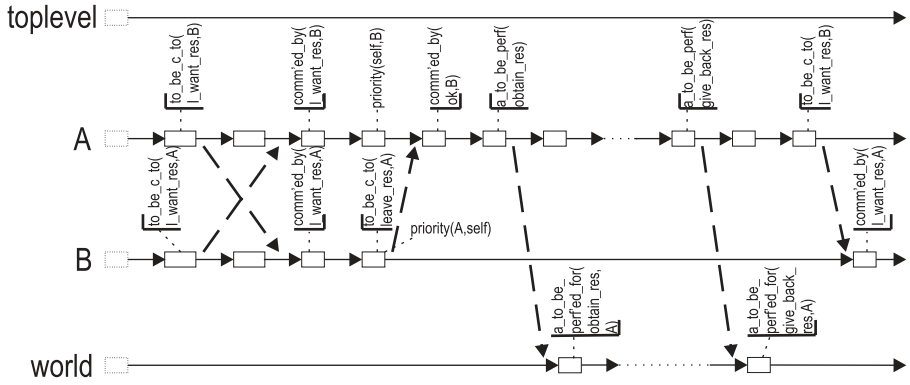

Figure 10.4: element of the white box view on the behaviour of the example system.

The local component traces for the agents *A* and *B* in such multitraces do not directly provide an answer to the question whether the agents access the resource in a mutually exclusive way. Consider the local component trace for component *A*, as depicted in Figure 10.4:

tr_A = …|…|to_be_communicated_to(I_want_resource,B)→…→
communicated_by(I_want_resource,B)|…|…→…|priority(self,B)|…→
communicated_by(ok,B)|…|…→
…|…|action_to_be_performed(obtain_resource)→…→

…|…|action_to_be_performed(give_back_resource)→…→
…|…|to_be_communicated_to(I_want_resource,B)→…→…

In this trace, A transmits to B a request for access to the resource. After some time, B grants access. Agent A then obtains the resource, probably executes other actions, and returns the resource. Some time later, A again wants access to the resource and transmits a request to B. The process may be repeated several times. A trace tr_B for B is of similar structure. If tr_A and tr_B are compatible, they may appear as elements of one multitrace in the white box view on the behaviour of the toplevel component. However, in the multitrace, the time points of tr_A are not directly related to the time points of tr_B. It is therefore not possible to determine if, for instance, between the occurrence of action_to_be_performed(obtain_resource) and the first occurrence of action_to_be_performed(give_back_resource) after the occurrence of action_to_be_performed(obtain_resource) for agent A, there is an occurrence of action_to_be_performed(obtain_resource) for agent B. (This situation clearly violates exclusive access.)

However, the trace of the external world can be used to distinguish violations of exclusive access. The following trace for the external world clearly is a violation of exclusive access:

tr_{world} = …→action_to_be_performed_for(obtain_resource,A)|…|…→
action_to_be_performed_for(obtain_resource,B)→…

Note that receipt of the atom action_to_be_performed_for(obtain_resource,A) is immediately followed by the receipt of the atom action_to_be_performed_for(obtain_resource,B).

Thus, the answer to the question whether access is indeed mutually exclusive can be obtained by proving (or disproving) that in no multitrace in the white box view on the behaviour of the toplevel component, there is a trace such as tr_{world}. There are, however, some additional issues:

- The safety of the system can be evaluated, as indicated in this section, because there is a trace available that records access to the resource (the local component trace of the external world). Both in DESIRE as well as in the semantic structure, transitions from one state to the next (i.e., the activity of components and links) cannot have side effects. (To be more precise, side effects are not represented in any way by DESIRE and the semantic structure. In prototypes of DESIRE models generated by the DESIRE software environment, side effects are not possible, except for one case, which is discussed at the end of this section.) As there are no side effects, access to the resource has to be explicitly modelled as action performance. The benefit is that the dynamics of resource access is represented by a trace for the resource and that it is possible to prove properties of resource access. Metaphorically speaking, the property is checked locally, for the resource in applications of the semantic structure. On the one hand, this aspect of

support for locality provided by the semantic structure probably reduces complexity of proofs. On the other hand, the model itself is more complex as it has to represent the resource itself;

- The central question of the evaluation of the model presented in this chapter can be approached by means of the global perspective presented in Chapter 7. As explained in Chapter 7, starting from e.g. the white box view on the behaviour of a component, a partial order of global states can be constructed. Different observers of the component observe different behaviour, where each observation corresponds with a path in the partial order of states. To ensure that access to the resource is mutually exclusive, there should be no path in the partial order in which there is a global state such that for (the output substate of) the local states of both *A* and *B*, action_to_be_performed(obtain_resource) is true[3]. In other words, not a single possible observer should observe a violation of mutually exclusive access. (See (Schwarz & Mattern, 1994) for a discussion of the relevance of different observers. See (Katz & Peled, 1990) for a logic that enables distinguishing properties observed by all observers from properties observed by a proper subset of all observers.) This approach resembles the usual approach to verification in concurrent systems. For instance, in Concurrent MetateM (Fisher & Wooldridge, 1997), resource access can be represented by a transition from a state in which the resource is not accessed to a state in which it is, *for the agent that accesses the resource.* The resource itself need not be explicitly represented as a Concurrent MetateM object. In other words, actually accessing the resource is a side effect of the state transition. The question whether such access is mutually exclusive is answered by proving that there is no global trace in which more than one agent performs this state transition at the same time. (As Concurrent MetateM assumes that global time is available, there is no need to distinguish different observations.); Metaphorically speaking, the property is checked globally.
- Similar evaluations as the evaluation proposed in this section have been performed for DESIRE models (based on an alternative semantic structure in which global states are assumed, see (Jonker & Treur, 1998a)). These evaluations consist of rigorous, mathematical proofs of the existence or non-existence of specific global traces. The complexity of these proofs calls for a method for *compositional verification*, which is presented in (Jonker & Treur, 1998a). The reader is referred to this publication for details. Providing similar proofs for the model provided in this chapter requires a refinement of the compositional verification method, as well as additional detail with respect to how a resource is accessed and how agents communicate;

[3] Actually, the proper requirement is more complicated.

- The question whether access is indeed exclusive is not the only question that needs to be answered. Manna and Pnueli (1992) present a hierarchy of temporal formulae that represent properties for concurrent systems. The highest level divides the class of all formulae in safety formulae ('something bad will never happen') and liveliness formulaes ('something good will eventually happen'). Manna and Pnueli provide a formal proof that each formulae is either a safety property or a liveliness formulae. Probably the most evident liveliness formulae to require for the model presented in this chapter is a fairness formulae, e.g., each agent that requests the resource is eventually granted the resource;
- In the discussion presented in this section, it is assumed that the resource is represented by (a subcomponent of) the external world. If the resource is an agent, similar remarks apply: the trace of this resource agent reveals whether agents *A* and *B* mutually exclusively transmit information to it.

To conclude this chapter, as an aside the following remark is made. In DESIRE, it is possible that primitive components of a specific type have side effects. This is the case for primitive components that do not have a knowledge base. Instead, the functionality of such a component is specified by some other means chosen by the user of the DESIRE framework. If the functionality is specified by a computer program, the DESIRE software environment executes this program whenever the associated component is made active. Such a program may alter the output interface of its component, but it may also execute actions that create side effects. For the evaluation of a model that relies on such side effects, the approach utilising the global perspective presented in Chapter 7 has to be followed.

Chapter 11

Example: A Society of Small Agents

Much research concerning the design of multi-agent systems (at a conceptual level) addresses complex agents that exhibit complex interaction patterns. Due to this complexity, it is difficult to perform rigorous experimentation. On the other hand, systematic experimental work regarding behaviour of societies of more simple agents, while reporting valuable results, often lacks conceptual specification of the system under consideration. In this chapter, the DESIRE modelling framework is not only used to develop a conceptual specification of the simple agents discussed in (Cesta, Micelli & Rizo, 1996a), but also to simulate the behaviour in a dynamical environment. The prototype automatically generated implementation of the conceptual specification of the simple agents has been used to replicate, and extend, one of the experiments reported in (Cesta *et al.*, 1996a).

This chapter is outlined as follows. Section 11.1 introduces the problem approached in this chapter. Section 11.2 provides a description of the multi-agent system examined in (Cesta et al., 1996a), for which a conceptual specification using DESIRE is introduced in Section 11.3. In Section 11.4, the conceptual specification is further developed, showing a level of detail suitable for automatic prototype generation. Section 11.5 presents results obtained by experimenting with the generated prototype. Section 11.6 discusses the results obtained as well as the relation with the semantic structure developed in this thesis. Section 11.7 provides a complete listing of all knowledge bases used in the primitive components. Preliminary versions of this chapter appeared at the International Conference on Computer Simulations and Social Sciences, ICCS&SS '97, (Brazier, Eck and Treur, 1997a) and in *Applied Intelligence* (Brazier, Eck and Treur, 2001a).

11.1 Introduction

Although much research within the multi-agent community has focused on the design of individual agents and their interaction, other research has addressed emergent behaviour within societies of agents (see for instance the three papers in

the chapter on emergence in (Velde *et al.*, 1996)). The behaviour of an individual agent can often be conceptually specified, as can the interaction between individual agents. The result of interaction between larger numbers of agents in a dynamic environment is often not easy to predict (Axelrod, 1997). Experimental research, in which interaction between agents is studied in a simulated dynamic environment, provides a means to actually test and compare results of interacting agents (Hanks, Pollack & Cohen, 1993). As stated in (Axelrod, 1997), the KISS principle is of utmost importance: the elements of a model need to be fully understood to be able to interpret results of experimentation.

In this chapter, the DESIRE modelling framework is used to conceptually specify individual agents and to examine the behaviour of relatively simple agents within a large group of agents. This method is supported by tools with which detailed specifications (with which the behaviour of individual agents and their interactions is defined) are automatically translated into prototype implementations. To examine such behaviour, experiments reported by (Cesta *et al.*, 1996a) that test social theories by simulating interaction between different types of simple agents (i.e., agents with limited knowledge and capabilities), have been repeated and extended.

Due to the nature of the environment in which the experimentation of (Cesta *et al.*, 1996a) was originally performed (using the MICE testbed (Montgomery & Durfee, 1990)), most information about agent characteristics and behaviour is implicitly defined by the implementation and simulation environment. On the basis of the informal, textual descriptions provided by (Cesta *et al.*, 1996a), a generic model of a simple agent is defined and refined for each of the four types of agents (Cesta et al., 1996a) distinguished: social, solitary, selfish and parasite. One of the aims of this chapter is to show how this approach leads to a flexible, conceptual-level specification, from which prototypes can be generated automatically for experimentation.

The conceptual models of the different agents are fully specified within DESIRE, including knowledge about how each individual agent interacts with its environment, and with other agents. One of the advantages of a conceptual description of an agent and its behaviour is that not only does a conceptual specification define the behaviour of an individual agent explicitly, it can also be easily adapted at a conceptual level (without having to rewrite low-level code for each agent). In the experiments described in (Cesta *et al.*, 1996a), agents could move in four different directions. In this chapter, not only are these experiments repeated with agents automatically implemented from the conceptual DESIRE specifications, additional experimentation is performed to examine the influence of an increase in the number of directions (8 instead of 4) in which agents can move.

11.2 The Original Experiment

(Cesta *et al.*, 1996a) examined the behaviour of different types of agents in interaction. Four types of agents are distinguished on the basis of their social

characteristics: social agents, parasite agents, solitary agents and selfish agents. The effect of an agent's social characteristic on interaction with other agents is measured by simulating agent behaviour in a situation in which 30 agents try to survive on a 15 * 15 grid in which 60 pieces of food are continually available in random positions. An agent's welfare is measured on the basis of its energy level. The end result of a simulation is the number of agents that survive in a given society of agents, given the energetic value of the food available. Agents do not communicate explicitly but implicitly: a hungry agent changes colour, and this can be seen by other agents. Agents' social characteristics are assumed to be static. An agent does not change from being, for example, selfish to social. The implications of agents' social characteristics for its behaviour are shown below in Table 11.1, taken from (Cesta *et al.*, 1996a), p. 131.

TYPE OF AGENT	INTERNAL STATE	GOAL
Solitary	*any*	Find Food
Parasite	*any*	Look for Help
Selfish	Danger	Look for Help
	Hunger, Normal	Find Food
Social	Danger	Look for Help
	Hunger	Find Food
	Normal	(if help-seekers are seen:) Give Help (if no help-seekers are seen:) Find Food

Table 11.1: Relationships among Types of Agent, Internal States, and Goals (from (Cesta et al., 1996a), p. 131).

The effects of interaction between societies in which 30 agents with varying configurations of social characteristics (for example, 15 social agents and 15 parasite agents in one world), and varying energetic food values have been examined in a number of experiments described in (Cesta *et al.*, 1996a). The MICE testbed, a discrete event simulator, was used to perform the experiments

mentioned above. The MICE testbed does not support a specific agent architecture: agents are LISP functions, activated pseudo-concurrently.

11.3 Conceptual Model of Simple Agents

At the highest level of abstraction, the conceptual model of the society of simple agents consists of 31 components: 30 agents and the external world. The agent components are named agent00 to agent29. The external world is connected to each agent by two links, e.g. agent00_to_world, world_to_agent00, etc. The highest abstraction level of the society of simple agents is depicted in Figure 11.1.

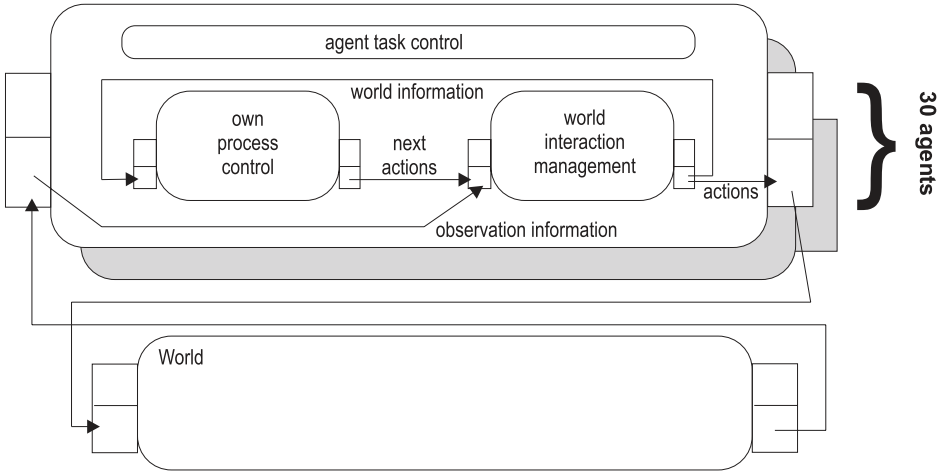

Figure 11.1: Model of the society.

The generic agent model presented in Chapter 3 includes more functionality than required for the small agents described in (Cesta *et al.*, 1996a). These small agents are not capable of communication with other agents and reasoning about other agents' knowledge, nor are they capable of reasoning about communication. Their only task is to stay alive in a dynamic environment. In fact, the only components within the generic agent model applicable to these small agents are the components own_process_control and world_interaction_management. Figure 11.2 depicts not only the remaining composition of a small agent's tasks at the highest level of abstraction, it also shows the information links between the components.

The only information a small agent receives is the information it observes in the external world. This information is forwarded directly to the component world_interaction_management. The component world_interaction_management interprets this information. The result, information about the agent's position, about available food, and, if applicable, information about other needy agents is transferred to the component own_process_control. The component own_process_control determines which

actions should be taken next, depending on the small agent's social characteristics and the agent's direct environment. This information is transferred to the component world_interaction_managment, which derives the information required to actually perform the action in the external world. This information is the only output a small agent provides to the external world. The external world maintains a representation of the grid in which the agents live and is responsible for actually performing the agent's actions by changing the state of the grid with respect to agent's positions and appearance. This updated state may be observed by other agents. (As in (Cesta *et al.*, 1996a), agents have a visibility range of three cells, that is, they observe a rectangular piece of the grid of size 7*7 cells, with the observing agent in the middle of this rectangle.) Other tasks of the external world are to place new food at random locations if a piece of food is eaten and maintaining statistics with respect to the number of alive agents. The internal structure of the component own_process_control is described below in Section 11.3.1. The internal structure of the world_interaction_managment is described in Section 11.3.2. The external world, fully specified in a C program, is not further discussed.

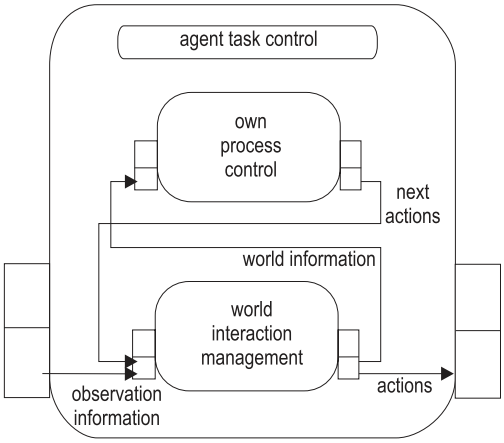

Figure 11.2: Generic structure of a small agent.

11.3.1 The Internal Structure of Component Own Process Control

The component own_process_control is composed of four components: own_resource_management, own_characteristics, goal_determination and plan_determination. The component own_resource_management receives information about its current energy level and the resources it has consumed. This component uses this information to determine its new energy level. On the basis of information the component goal_determination receives about its own social characteristics and its own energy level, it determines the goals the agent is to pursue: for example to find food, or to look for help. The component own_characteristics receives information on the agent's

energy level from the component own_resource_management. This information is used to determine the agent's next state (e.g., hungry, normal or in danger). The component plan_determination receives information (1) from the component own_characteristics, namely the agent's current state, (2) from the component goal_determination, namely which goals are to be pursued and (3) from outside the component, namely the current state of the world. With this information the component plan_determination determines which actions to take in the external world. Figure 11.3 depicts the composition structure of own_process_control.

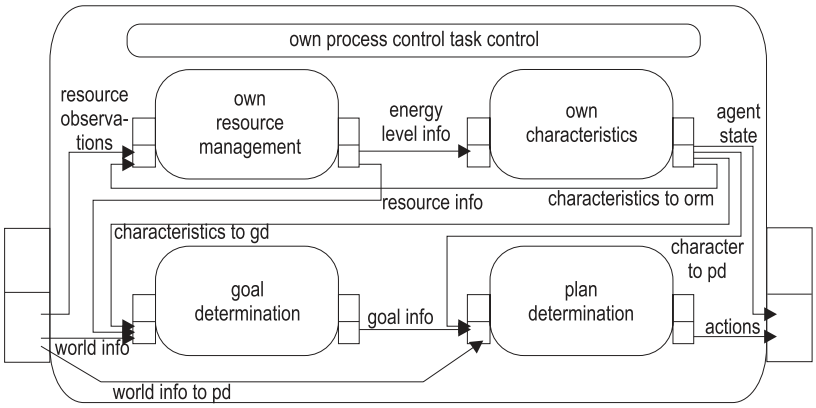

Figure 11.3: Component own_process_control.

11.3.2 Internal Structure of the Component World Interaction Management

The component world_interaction_management interprets information received from the external world, and transforms information about actions to be taken in the external world into specifications for actions to be executed in the external world. Two components are defined to perform these tasks: the component observation_information_interpretation and the component action_execution_preparation.

The component observation_information_interpretation receives information from the external world, for example, information on which pieces of food and which agents have been observed within a given range. This information, termed sensory information in (Cesta *et al.*, 1996a), is translated into information which can be used by the component own_process_control to reason about new goals and plans.

As stated above, the component action_execution_preparation receives information about actions to be taken in the external world from the component own_process_control and translates these actions into specifications to be executed in the external world. These specifications are also the output of effectors in the terminology used in (Cesta *et al.*, 1996a): elementary actions to be performed in the external world. Figure 11.4 depicts the composition structure of world_interaction_management.

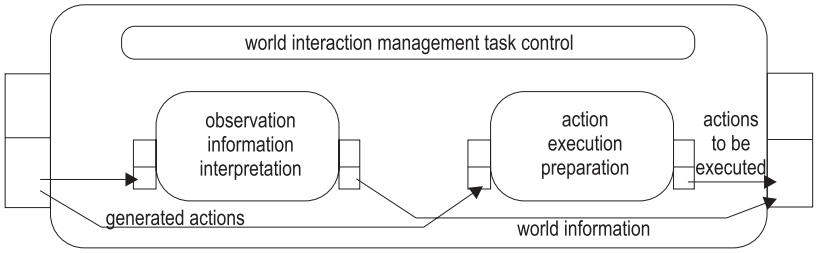

Figure 11.4: Component world_interaction_management.

11.4 Detailed Design

The conceptual generic model of a small agent presented above, has been specified in detail. In this section, relevant examples of the specification are presented to illustrate the level of abstraction with which knowledge is represented. (See Section 11.7 for a complete overview of the knowledge bases used in the primitive components of the agents.) The specification as a whole contains sufficient detail to allow for automatic prototype implementation and simulation. The knowledge structures used by the two main components of a generic small agent, the component world_interaction_management and the component own_process_control are discussed below in more detail, together with the information exchange.

11.4.1 World Interaction Management

The composed component world_interaction_management receives, as input, observation information (obtained from the external world), expressed by the (unary) relation observed, and information from the component own_process_control, expressed by the relation next_action. As output, world information is provided (to be used by own_process_control), and the actions that are to be executed, expressed by the relation to_perform (to be placed in the output interface of the agent).

11.4.1.1 Observation Information Interpretation

The primitive component observation_information_interpretation receives observation information as input and draws conclusions from this information. As an example, the following rules interpret the last action performed by the agent:

```
if      observed( prev_performance( pick_up_food ))
then    food_in_possession;

if      observed( prev_performance( feed_help_seeker ))
then    not food_in_possession;

if      observed( prev_performance( feed_self ))
```

```
then    not food_in_possession;
```

These rules state that food is in possession if an agent has previously picked up a piece of food. Food is no longer in possession if an agent has either eaten the food itself, or has given it to a help seeking agent. See Section 11.7.1.1 for the complete knowledge base.

11.4.1.2 Action Execution Preparation

In the primitive component action_execution_preparation the executability of actions that have been selected within the component own_process_control is verified. If an action is dependent on a number of preconditions, these preconditions are tested. If the preconditions are satisfied, the conclusion is drawn that the action should be performed. For example, to be able to take food, that food must be available at the same position as the agent. This is formalised using the following rule:

```
if      next_action( pick_up_food )
        and observed( food_at( cell( 0, 0 )))
then    to_perform( pick_up_food );
```

Cell locations in observations are relative to the position of the agent. Therefore, to be able to pick up food, it must be observed at location *(0,0)*. The component action_execution_preparation also prepares the execution of plans to move to a specific cell. The following rules, marked with a bar in the margin for reference purposes later in this chapter, show the preparation of movement to the north or the south:

```
if      next_action( move_to( cell( X: ints, Y: ints )))
        and  Y: ints > 0
then    to_perform( go_north );
```

```
if      next_action( move_to( cell( X: ints, Y: ints )))
        and Y: ints < 0
then    to_perform( go_south );
```

See Section 11.7.1.2 for the complete knowledge base.

11.4.2 Own Process Control

The component own_process_control receives world information from the component world_interaction_management and determines the next actions to be performed, expressed by the relation next_action.

11.4.2.1 Own Characteristics

Within the primitive component own_characteristics an agent's own characteristics are specified. Because the example society contains agents of different types the knowledge differs between agents; it is expressed using the following information types. The first information type expresses the characteristics that in the current chapter are assumed to be static (they are pre-specified), such as the (meta-)fact

that an agent is selfish; the second information type expresses dynamic characteristics, such as being in danger.

```
information type agent_character_it
        sorts    Agent_character
        objects solitary, parasite, selfish, social: Agent_character;
        relations agent_character: Agent_character;
end information type

information type agent_state_it
        sorts Agent_state
        objects dead, in_danger, hungry, normal: Agent_state;
        relations current_state: Agent_state;
end information type
```

In this component, the state of an agent is determined based on its energy level, as shown by the following rule:

```
if      not energy_level < 20
        and energy_level < 60
then    current_state( hungry );
```

See Section 11.7.2.1 for the complete knowledge base.

11.4.2.2 Own resource management

In the primitive component own_resource_management the energy level of an agent is determined. An agent's energy level changes as a result of actions an agent has performed: for example after eating food with food value 40 the result is to_increment_energy_level_with(40). An information link links these atoms to atoms of the form delta_energy_level = 40, after which the new energy level can be determined. The following rule, marked with a bar in the margin for reference purposes later in this chapter, shows that in the case that an agent does nothing, its energy level decreases by one:

```
if      not observed( prev_performance( eat_food ))
        and not observed( prev_performance( go_north ))
        and not observed( prev_performance( go_south ))
        and not observed( prev_performance( go_east ))
        and not observed( prev_performance( go_west ))
        and not observed( prev_performance( change_appearance ))
        and not observed( prev_performance( pick_up_food ))
        and not observed( prev_performance( feed_help_seeker ))
        and not observed( prev_performance( feed_self ))
then    to_increment_energy_level_with( -1 );
```

See Section 11.7.2.2 for the complete knowledge base.

11.4.2.3 Goal Determination

In the primitive component goal_determination an agent determines its goals on the basis of information about its own characteristics, its current state and partial

world information. Selected goals are defined by the following information type:

```
information type agent_goal_it
        sorts Agent_goal;
        objects find_food, look_for_help, give_help: Agent_goal;
        relations selected_goal: Agent_goal;
end information type
```

See Section 11.7.2.3 for the complete knowledge base, which is a direct formalisation of Table 11.1.

11.4.2.4 Plan Determination

Based on an agent's goals the component plan_determination determines which plans are to be executed. The information type to express selected actions is as follows:

```
information type agent_action_it
        information types cell_it;
        sorts Agent_action
        objects
                pick_up_food, eat_food, change_appearance,
                feed_self, feed_help_seeker: Agent_action;
        functions
                move_to: Cell -> Agent_action;
        relations
                next_action: Agent_action;
end information type
```

As an example of action selection, consider the following rule:

```
if      selected_goal( give_help )
        and food_in_possession
        and closest( help_seeker_at( cell( X: ints, Y: ints )))
        and not X: ints = 0
then    next_action( move_to( cell( X: ints, Y: ints )));
```

This rule states that if the selected goal is to give help, and food is already in possession of an agent (so the agent does not have to look for food first), and the closest help seeking agent is observed at location (X,Y), relative to the agent, which is not our own position (and therefore, $X \neq 0$), then the agent decides to go to that cell. See Section 11.7.2.4 for the complete knowledge base.

11.4.3 Control Knowledge

Control knowledge is specified at all levels of abstraction. This subsection briefly describes control knowledge at the highest level of abstraction, namely the society level and one level lower, namely control knowledge within individual agents.

11.4.3.1 Control Knowledge at the Society Level

Control knowledge at the society level specifies how individual agents are activated. One option is to run all agents in parallel, the other is to predefine the

sequence of activation. In the first case only one control rule is needed:

```
if      start
then    next_component_state( external_world, awake )
        and next_component_state( agent00, awake )
        …
        and next_component_state( agent29, awake )
        and next_link_state( world_to_agent00, uptodate )
        …
        and next_link_state( world_to_agent29, uptodate )
        and next_link_state( world_to_agent00, uptodate )
        …
        and next_link_state( world_to_agent29, awake );
```

This control rule specifies that immediately after the system has started, the external world, each individual agent, and the links between the agents and the world, are all made awake. In this state, each agent, a link and the external world are continually ready to receive information and becoming active upon receipt of new information. Moreover, all components immediately start processing as specified by their default task control focus (as explained in Chapter 9). For the external world, this processing determines observations for all agents. For the agents, this processing consists of interpreting observations received from the external world. Thus, immediately after the system has started, the external world is the first component to actually run. After that, observations are transmitted to the agents, which become active concurrently at the moment the observations are received.

A second option is to exercise more control over the execution sequence between agents and the activation of links. In this case, the following control rules may be specified at the society level:

```
if      start
then    next_component_state( external_world, active )
        and next_task_control_focus( external_world, determine_observations );

if      evaluation( external_world, determine_observations, all_p, succeeded )
        and not previous_evaluation( external_world, determine_observations, all_p, succeeded )
then    next_component_state( agent00, active )
        and next_task_control_focus( agent00, default_focus )
        and next_link_state( world_to_agent00, uptodate );

if      evaluation( agent00, default_focus, all_p, succeeded )
        and not previous_evaluation( agent00, default_focus, all_p, succeeded )
then    next_link_state( agent00_to_world, uptodate )
        and next_component_state( agent01, active )
        and next_task_control_focus( agent00, default_focus )
        and next_link_state( world_to_agent01, uptodate );

…

if      evaluation( agent28, default_focus, all_p, succeeded )
        and not previous_evaluation( agent28, default_focus, all_p, succeeded )
then    next_link_state( agent28_to_world, uptodate )
```

```
        and next_component_state( agent29, active )
        and next_task_control_focus( agent29, default_focus )
        and next_link_state( world_to_agent29, uptodate );

if      evaluation( agent29, default_focus, all_p, succeeded )
        and not previous_evaluation( agent29, default_focus, all_p, succeeded )
then    next_link_state( agent29_to_world, uptodate )
        and next_component_state( external_world, active )
        and next_task_control_focus( external_world, determine_observations );
```

The first rule specifies that, immediately after the system is started, the external world becomes active, processes the available information in view of its task control focus, and becomes idle. In this case, a task control focus is set such that the external world makes observations available to each of the agents. The second rule specifies that if the external world has successfully met the evaluation criterion determine_observations, a specific agent, agent00, is made active, and observations from the world are transmitted to this agent by up-dating the link from the external world to agent00. The following rules specify that the agents are activated sequentially, and the relevant links between an agent and the external world, and the external world and the next agent to be activated, are up-dated. The agents transmit actions to be performed to the external world. The external world is activated once all agents have been given the opportunity to convey their own actions to the external world. After that the cycle is repeated. As the experimental system was designed to replicate (Cesta *et al.*, 1996a), option 2 has been implemented.

11.4.3.2 Control Knowledge at the Agent Level

Within an agent, the execution order of the components that correspond to the four tasks identified in (Cesta *et al.*, 1996a) should mirror the execution sequence described in (Cesta *et al.*, 1996a). This means that observation_information_interpretation, goal_determination, plan_determination, and action_execution_preparation should run in this order. However, these components are not at the agent level. Instead, they are subcomponents of the agent-level components own_process_control and world_interaction_management. The following agent-level control rules specify that these components are executed in the correct sequence:

```
if      start
then    next_component_state( world_interaction_management, active )
        and next_task_control_focus( world_interaction_management, interpret_observations );

if      evaluation( world_interaction_management, interpret_observations, all_p, succeeded )
        and not previous_evaluation( world_interaction_management, interpret_observations,
                                     all_p, succeeded )
then    next_component_state( own_process_control, active )
        and next_task_control_focus( own_process_control, determine_plan );

if      evaluation( own_process_control, determine_plan, any, succeeded )
        and not previous_evaluation( own_process_control, determine_plan, any, succeeded )
```

```
        and previous_task_control_focus( own_process_control, determine_plan )
then    next_component_state( world_interaction_management, active )
        and next_task_control_focus( world_interaction_management, prepare_action_execution );
```

For each of the components own_process_control and world_interaction_management, more specific control rules at the component level are specified. These rules activate specific subcomponents depending on the current task control focus for the component. More specifically, control rules of own_process_control first activate goal_determination before plan_determination as a result of an activation with task control focus determine_plan.

Control rules at the agent level also determine the activation order of links between own_process_control and world_interaction_management. Moreover, control knowledge for the component own_process_control also specifies the activation order for the subcomponents own_resource_management and own_characteristics. For reasons of brevity, these control rules are not presented here.

11.5 Experimentation

The first goal of this exercise is to replicate the results presented in (Cesta *et al.*, 1996a), on the basis of a conceptual specification of agent behaviour, simulated in the DESIRE software environment. The second goal is to examine the effects of increasing the number of directions in which an agent can move from 4 to 8.

11.5.1 Method

One of the experiments discussed in (Cesta *et al.*, 1996a) has 15 social agents and 15 parasite agents with varying food energetic values in one world, with 500 steps per agent per run. The first experiment based on the DESIRE model used the same number and types of agents and the same number of steps. The second experiment consisted of re-running the first experiment in an environment in which agents could move in more than 4 directions: they could move in 8 directions. The experiments were carried out using an early version of the DESIRE prototype generator, which did not support a hierarchical component structure and which executed prototypes pseudo-concurrently on one processor. The prototype generator was therefore augmented with a custom tool that transformed the hierarchical model presented in this chapter to a non-hierarchical model.

For thes second experiment, extra rules had to be added to the different knowledge bases:

The following rules have been added to the knowledge base of component action_execution_preparation (Section 11.4.1.2):

```
if      next_action( move_to( cell( X: ints, Y: ints )))
        and X: ints > 0
        and Y: ints > 0
then    to_perform( go_northeast );
```

```
if      next_action( move_to( cell( X: ints, Y: ints )))
        and  X: ints < 0
        and  Y: ints > 0
then    to_perform( go_northwest );

if      next_action( move_to( cell( X: ints, Y: ints )))
        and  X: ints < 0
        and  Y: ints < 0
then    to_perform( go_southwest );

if      next_action( move_to( cell( X: ints, Y: ints )))
        and  X: ints > 0
        and  Y: ints < 0
then    to_perform( go_southeast );
```

The rules marked with a bar in the knowledge base of component action_execution_preparation (Section 11.4.1.2) have to be changed:

```
if      next_action( move_to( cell( 0, Y: ints )))
        and  Y: ints > 0
then    to_perform( go_north );

if      next_action( move_to( cell( 0, Y: ints )))
        and Y: ints < 0
then    to_perform( go_south );
```

The following rules have been added to the knowledge base of component own_resource_management (Section 11.4.2.2):

```
if      observed( prev_performance( go_northwest ))
then    to_increment_energy_level_with( -2 );

if      observed( prev_performance( go_northeast ))
then    to_increment_energy_level_with( -2 );
```

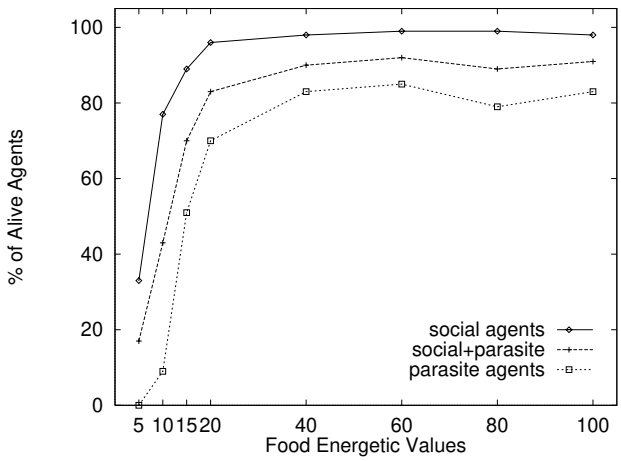

Figure 11.5: Comparison between Parasite and Social (from (Cesta et al., 1996a), p. 133).

```
if      observed( prev_performance( go_southwest ))
then    to_increment_energy_level_with( -2 );

if      observed( prev_performance( go_southeast ))
then    to_increment_energy_level_with( -2 );
```

The rule marked with a bar in the knowledge base of component own_resource_management (Section 11.4.2.2) has to be changed by adding the following conjuncts to the rule condition:

```
and not observed( prev_performance( go_northwest ))
and not observed( prev_performance( go_southwest ))
and not observed( prev_performance( go_northeast ))
and not observed( prev_performance( go_southeast ))
```

11.5.2 Results

Figure 11.5 depicts results averaged over 10 runs as presented in (Cesta *et al.*, 1996a), p. 133. Figure 11.6 depicts the DESIRE results on the basis of 2 runs. Figure 11.7 shows the results for the same experiment in the more flexible environment (thus, with 8 directions of movement), averaged over 5 runs.

11.5.3 Evaluation

The results acquired in the DESIRE simulation are comparable to the results in (Cesta *et al.*, 1996a): social agents survive more often than parasite agents in situations with low food energetic values. The same holds for the experiment in which agents have more degrees of freedom. In our experiments, the chance of survival increases with an increase in the degree of freedom for both types of agents. Additional experimentation with different situations have been reported by Cesta, Micelli and Rizo in (Cesta *et al.*, 1996b).

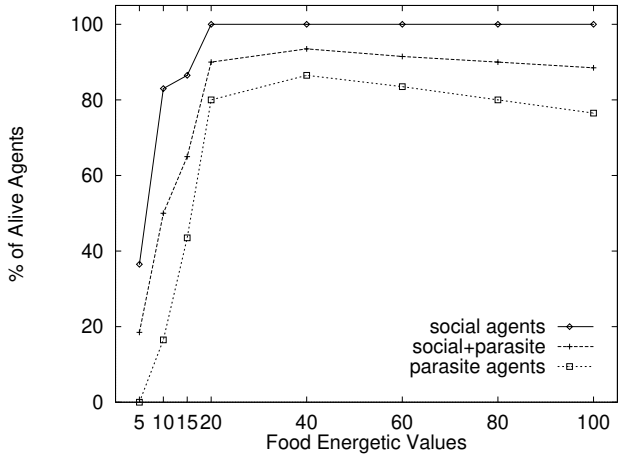

Figure 11.6: Comparison between Parasite and Social. Results from DESIRE prototype.

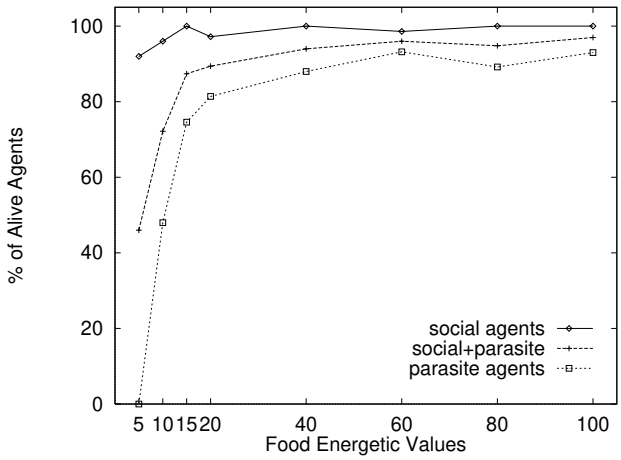

Figure 11.7: Comparison between Parasite and Social. 8 instead of 4 directions for movement.

11.6 Discussion

Much research concerning the design of multi-agent systems (at a conceptual level) addresses complex agents which exhibit complex interaction patterns. Due to this complexity, it is difficult to perform rigorous experimentation. On the other hand, systematic experimental work regarding behaviour of societies of more simple agents, while reporting valuable results, often lacks conceptual specification of the system under consideration.

In this chapter, the DESIRE modelling framework is not only successfully used to develop a conceptual specification of the simple agents discussed in (Cesta *et al.*, 1996a), but also to simulate the behaviour in a dynamical environment. In DESIRE, a conceptual specification, which provides a high-level view of an agent, has enough detail for automatic prototype generation. As stated in Section 9.5, support for knowledge-intensive domains is an advantage of DESIRE over conventional modelling frameworks such as UML. In the application presented in this chapter knowledge-intensity and complexity were not distinguishing factors; in applications with more complexity (for example more complex knowledge within cognitive agents) this advantage is more clear.

A social simulation environment that comes close to DESIRE is SDML (see (Moss, Gaylard, Wallis & Edmonds, 1998)). As in DESIRE, in SDML declarative specification is a central aim. Some of the differences are:

- In DESIRE time can be left implicit, whereas in SDML it is automatically attached to the information structures;
- In DESIRE a more strict form of modularity is used;

- DESIRE provides explicit constructs for control knowledge;
- DESIRE is supported by a distributed execution platform.

The prototype implementation of the conceptual specification of the simple agents has been used to replicate and extend one of the experiments reported in (Cesta *et al.*, 1996a). One of the advantages of conceptual specification has been explored, namely the ease with which existing specifications can be modified. The conceptual specification of a simple agent was modified as follows: the number of directions in which the simple agents can move was increased from 4 to 8, by two minor, local modifications to knowledge included in the specification. A new experiment was performed to compare the behaviour of these agents to the simple agents with only 4 directions of movement.

The ultimate goal of research in the Social Sciences is to develop a theory that explains how the behaviour of the society as a whole (in this chapter, the survival rate of the agents) emerges from the behaviour of the individual agents (which, in this chapter, is described by Table 11.1). Castelfranchi and Conte (1996) call such a theory a middle ground theory. As stated in Chapter 1, such a theory is also helpful in the area of multi-agent systems to predict the behaviour of a multi-agent system under development.

A possible use of the semantic structure developed in this thesis for the development of a middle ground theory can be illustrated using the multi-agent system described in this chapter. As the behaviour of the individual agents in the system is modelled using the DESIRE modelling framework, associated with each agent is a DESIRE structure hierarchy with control that represents the agent in the semantic structure. This DESIRE structure hierarchy with control contains the components presented in this chapter, but also control components associated with the composed components, as described in Chapter 9. For a primitive component or link in this control structure, a set $Beh_{loc}(S)$ is given by the standard dynamics of DESIRE knowledge bases. Together with the standard DESIRE compatibility relations, the behaviour of a single agent can be described by, among others, the white box view on the behaviour of the component that represents the agent. In this way, white box views can be obtained for each of the 30 agents in the society.

An element of the white box view on the behaviour of a simple agent, agent00, is depicted in Figure 11.8. (The name of this agent is abbreviated to a00.) Elements of the white box view are compatible multitraces that consist of local component and link traces for the components a00, $a00_{ctr}$, WIM and OPC, and for the links between these components. (Local link traces for the links are not depicted in Figure 11.8.) In Figure 11.8, boxes denote local component states. Horizontal, solid arrows between boxes denote state transitions. Diagonal, dashed arrows denote information transmission. A number of the input and output atoms for most states are also depicted in Figure 11.8. Atoms at the left side of the small vertical bars are input atoms, atoms at the right side are output atoms. The names of input and output atoms are abbreviated as follows: comp for component, tcf for task_control_focus,

obs_interpr for observation_interpretation, eval for evaluation, WIM for world_interaction_management, OPC for own_process_control, determ_plan for determine_plan, and prep_a_exe for prepare_action_execution.

In the upper left corner of Figure 11.8, agent00 receives new observations. Upon receipt of these new observations, task control at the agent level determines that world_interaction_management is the first subcomponent to activate. Task control focus for world_interaction_management is observation_interpretation, which directs world_interaction_management to activate its subcomponent observation_interpretation. (This is not visible in the white box view of the agent level. Task control rules for world_interaction_management specify that subcomponent observation_interpretation is activated whenever the task control focus of world_interaction_management is observation_interpretation.) In the example trace in Figure 11.8 observation_interpretation concludes that a help seeker is present. The next component activated by task control at the agent level is own_process_control. This component first activates its subcomponent goal_determination, and after that plan_determination. (In the white box view of the agent level, this subcomponent activation is not visible.) In the example trace, plan_determination decides to move to a specific cell. This result is transmitted to world_interaction_management. Task control at the agent level activates world_interaction_management with task control focus prepare_action_execution. As a result, subcomponent action_execution_preparation of observation_interpretation is activated, which determines that the next action to execute is go_south.

To support the development of a middle ground theory, the essential step consists of composing a description of the overall, emerging behaviour of the society from the separate descriptions of the behaviour of the agents. Proposition 5.26, presented in Chapter 5, enables such a composition: roughly speaking, this proposition states that a multitrace for a structure hierarchy that represents the society is an element of the glass box view on the behaviour of the society if, for each agent, the restriction of the multitrace to this agent is an element of the white box view on the behaviour of the agent. The three views on the behaviour of a compositional system presented in Chapter 5, together with the propositions that relate these three views, provide facilities for the development of theories on emergent behaviour. These facilities are flexible with respect to the amount of detail represented in the behaviour of individual agents or the society as a whole. Note that Proposition 5.26 is not itself a middle ground theory. However, this proposition facilitates the development of such a theory. Alternatively, as the complete society of 30 simple agents, together with the external world, is modelled using DESIRE, it is also possible to construct a DESIRE structure hierarchy with control that represents the society directly from the DESIRE model.

Figure 11.9 depicts an example element of the white box view on the behaviour of the entire society. All agents, agent00 to agent29, (abbreviated a00 to a29), together with a component that represents the world, are made subcomponent of a component called toplevel. The white box view on the behaviour of toplevel consists of local component and link traces for toplevel, agent00 to agent29, the world, and all

links between agents and the world. (The traces for links are not depicted in Figure 11.9.) In Figure 11.9, the world transmits observations to each agent, and receives actions to execute from each agent in turn. The element of the white box view depicted in Figure 11.9 is composed of elements of the white box views of the agents as described above. For example, in Figure 11.9, the local component trace for a00 consists of two copies of the trace for component a00 depicted in Figure 11.8.

A possible direction for future research is the design of more complex agents: agents capable of adapting their own characteristics to increase their chances of survival. These agents possess the capability to learn from the observed effects of their own behaviour and that of others. For some first steps in compositional modelling of adaptive animal behaviour, see (Jonker & Treur, 1998b). The explicit conceptual specification makes it possible to make such adaptations at a conceptual level. For example, the explicitly specified agent characteristic (the own_character(selfish) fact in the knowledge base of Section 11.4.2.2) can be replaced by knowledge that can be used to derive the characteristic in a dynamic manner. Another extension of this work is current research on agents that can design new agents on the basis of given requirements. Some results in this direction can be found in (Brazier, Jonker, Treur & Wijngaards, 2000).

11.7 Knowledge Bases

In this section, complete knowledge bases for each primitive component are presented. The structure of this section follows the structure of Section 11.4.

11.7.1 World Interaction Management

The knowledge bases for the (primitive) subcomponents of world_interaction_management are presented in Section 11.7.1.1 and Section 11.7.1.2.

11.7.1.1 Observation Information Interpretation

Knowledge base:

if observed(closest_food_at(C: Cell))
then closest(food_at(C: Cell));

if observed(closest_help_seeker_at(C: Cell))
then closest(help_seeker_at(C: Cell));

if observed(help_seeker_at(C: Cell))
then help_seeker_present;

if observed(self_looking_like_a_help_seeker)
then looking_like_a_help_seeker;

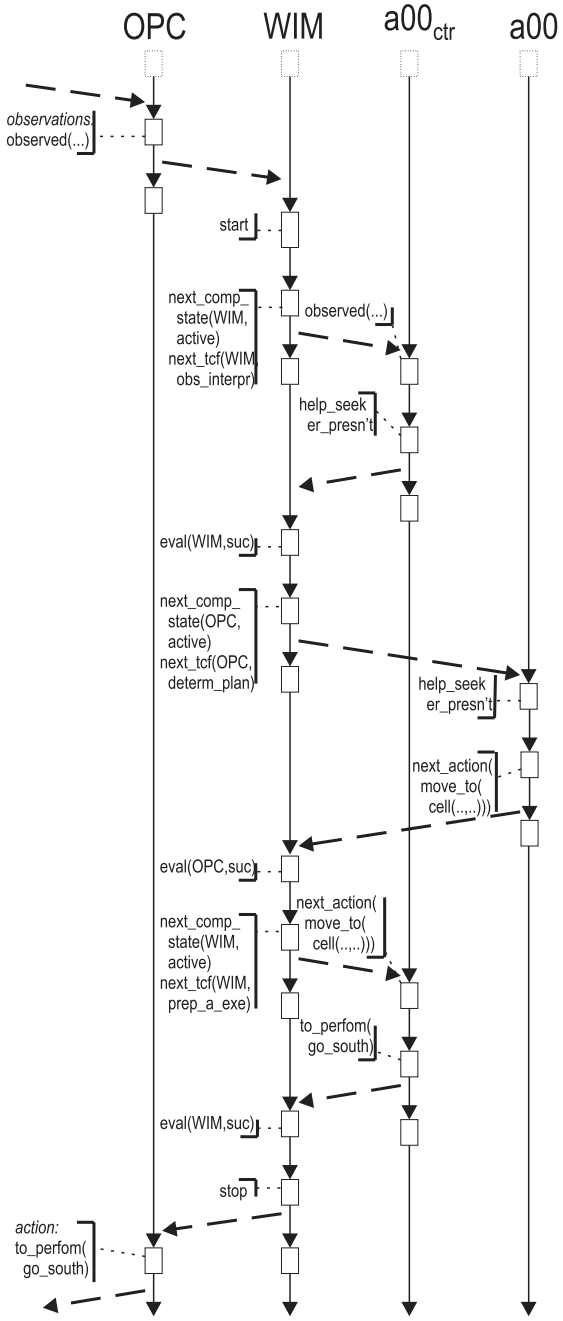

Figure 11.8: Element of the white box view on the behaviour of a simple agent.

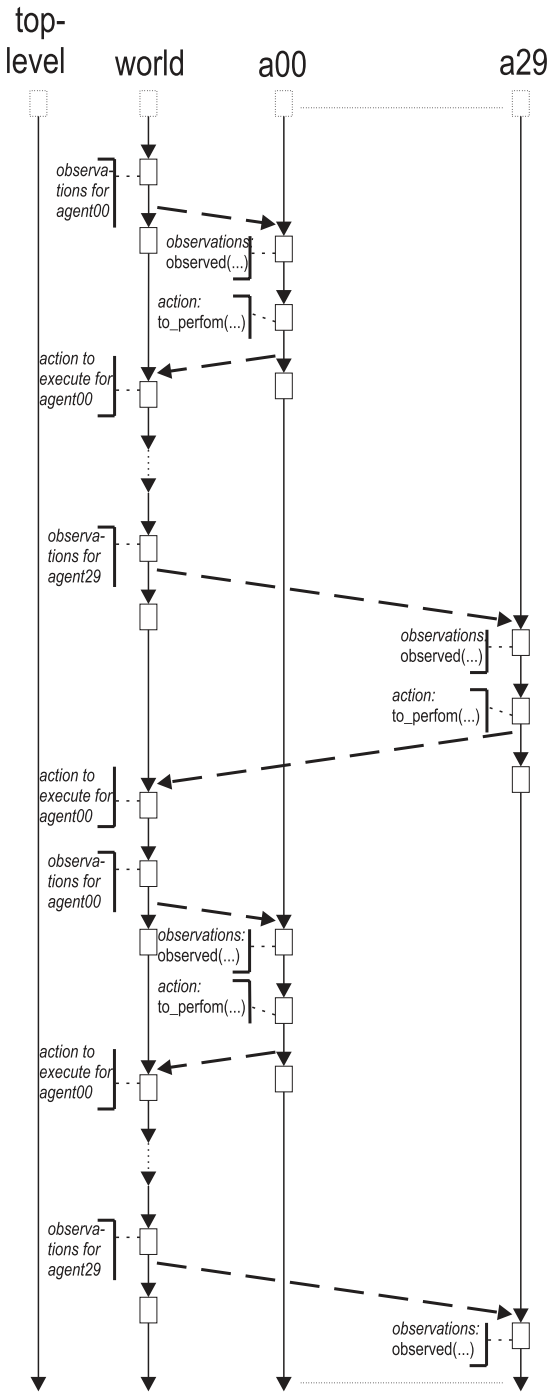

Figure 11.9: Element of the white box view on the behaviour of a society of simple agents.

```
if      not observed( self_looking_like_a_help_seeker )
then    not looking_like_a_help_seeker;

if      observed( prev_performance( pick_up_food ))
then    food_in_possession;

if      observed( prev_performance( feed_help_seeker ))
then    not food_in_possession;

if      observed( prev_performance( feed_self ))
then    not food_in_possession;
```

11.7.1.2 Action Execution Preparation

Knowledge base (some rules are marked with a bar in the margin for reference purposes):

```
if      next_action( change_appearance )
then    to_perform( change_appearance );

if      next_action( feed_self )
then    to_perform( feed_self );

if      next_action( pick_up_food )
        and observed( food_at( cell( 0, 0 )))
then    to_perform( pick_up_food );

if      next_action( feed_help_seeker )
        and observed( help_seeker_at( cell( 0, 0 )))
then    to_perform( feed_help_seeker );

if      next_action( eat_food )
        and observed( food_at( cell( 0, 0 )))
then    to_perform( eat_food );

if      next_action( move_to( cell( X: ints, Y: ints )))
        and  Y: ints > 0
then    to_perform( go_north );

if      next_action( move_to( cell( X: ints, Y: ints )))
        and Y: ints < 0
then    to_perform( go_south );

if      next_action( move_to( cell( X: ints, 0 )))
        and  X: ints > 0
then    to_perform( go_east );

if      next_action( move_to( cell( X: ints, 0 )))
        and  X: ints < 0
then    to_perform( go_west );
```

11.7.2 Own Process Control

The knowledge bases for the (primitive) subcomponents of own_process_control are presented in Section 11.7.2.1 to Section 11.7.2.4.

11.7.2.1 Own Characteristics

Knowledge base (for a selfish agent):

```
agent_character( selfish );

if      energy_level < 0
then    current_state( dead );

if      current_state( dead )
then    agent_died;

if      not energy_level < 0
        and energy_level < 20
then    current_state( in_danger );

if      not energy_level < 20
        and energy_level < 60
then    current_state( hungry );

if      not energy_level < 60
then    current_state( normal );
```

11.7.2.2 Own Resource Management

Knowledge base (for food value 40):

```
if      observed( prev_performance( eat_food ))
then    to_increment_energy_level_with( 40 );

if      observed( prev_performance( feed_self ))
then    to_increment_energy_level_with( 40 );

if      observed( prev_performance( go_north ))
then    to_increment_energy_level_with( -2 );

if      observed( prev_performance( go_south ))
then    to_increment_energy_level_with( -2 );

if      observed( prev_performance( go_west ))
then    to_increment_energy_level_with( -2 );

if      observed( prev_performance( go_east ))
then    to_increment_energy_level_with( -2 );

if      observed( prev_performance(
                        change_appearance ))
then    to_increment_energy_level_with( -1 );
```

```
if      observed( prev_performance( pick_up_food ))
then    to_increment_energy_level_with( -1 );

if      observed( prev_performance( feed_help_seeker ))
then    to_increment_energy_level_with( -1 );

if      not observed( prev_performance( eat_food ))
        and not observed( prev_performance( go_north ))
        and not observed( prev_performance( go_south ))
        and not observed( prev_performance( go_east ))
        and not observed( prev_performance( go_west ))
        and not observed( prev_performance( change_appearance ))
        and not observed( prev_performance( pick_up_food ))
        and not observed( prev_performance( feed_help_seeker ))
        and not observed( prev_performance( feed_self ))
then    to_increment_energy_level_with( -1 );

if      old_energy_level = V1: ints
        and delta_energy_level = V2: ints
        and not V1: ints + V2: ints > 100
then    energy_level = V1: ints + V2: ints;

if      old_energy_level = V1: ints
        and delta_energy_level = V2: ints
        and V1: ints + V2: ints > 100
then    energy_level = 100;
```

11.7.2.3 Goal Determination

Knowledge base:

```
if      agent_character( solitary )
then    selected_goal( find_food );

if      agent_character( parasite )
then    selected_goal( look_for_help );

if      agent_character( selfish )
        and current_state( in_danger )
then    selected_goal( look_for_help );

if      agent_character( selfish )
        and current_state( hungry )
then    selected_goal( find_food );

if      agent_character( selfish )
        and current_state( normal )
then    selected_goal( find_food );

if      agent_character( social )
        and current_state( in_danger )
then    selected_goal( look_for_help );

if      agent_character( social )
        and current_state( hungry )
```

```
then    selected_goal( find_food );

if      agent_character( social )
        and current_state( normal )
        and help_seeker_present
then    selected_goal( give_help );

if      agent_character( social )
        and current_state( normal )
        and not help_seeker_present   /* Obtained by CWA */
then    selected_goal( find_food );
```

11.7.2.4 Plan Determination

Knowledge base:

```
if      selected_goal( look_for_help )
        and not looking_like_a_help_seeker
then    next_action( change_appearance );

if      selected_goal( find_food )
        and looking_like_a_help_seeker
then    next_action( change_appearance );

if      selected_goal( give_help )
        and looking_like_a_help_seeker
then    next_action( change_appearance );

if      selected_goal( find_food )
        and food_in_possession
then    next_action( feed_self )

if      selected_goal( find_food )
        and not food_in_possession
        and closest( food_at( cell( 0, 0 )))
then    next_action( eat_food );

if      selected_goal( find_food )
        and not food_in_possession
        and closest( food_at( cell( X: ints, Y: ints )))
        and not X: ints = 0
then    next_action( move_to( cell( X: ints, Y: ints )));

if      selected_goal( find_food )
        and not food_in_possession
        and closest( food_at( cell( X: ints, Y: ints )))
        and not Y: ints = 0
then    next_action( move_to( cell( X: ints, Y: ints )));

if      selected_goal( give_help )
        and not food_in_possession
        and closest( food_at( cell( X: ints, Y: ints )))
        and not X: ints = 0
then    next_action( move_to( cell( X: ints, Y: ints )));
```

```
if      selected_goal( give_help )
        and not food_in_possession
        and closest( food_at( cell( X: ints, Y: ints )))
        and not Y: ints = 0
then    next_action( move_to( cell( X: ints, Y: ints )));

if      selected_goal( give_help )
        and not food_in_possession
        and closest( food_at( cell( 0, 0 )))
then    next_action( pick_up_food )

if      selected_goal( give_help )
        and food_in_possession
        and closest( help_seeker_at( cell( X: ints, Y: ints )))
        and not X: ints = 0
then    next_action( move_to( cell( X: ints, Y: ints )));

if      selected_goal( give_help )
        and food_in_possession
        and closest( help_seeker_at( cell( X: ints, Y: ints )))
        and not Y: ints = 0
then    next_action( move_to( cell( X: ints, Y: ints )));

if      selected_goal( give_help )
        and food_in_possession
        and closest( help_seeker_at( cell( 0, 0 )))
then    next_action( feed_help_seeker );
```

Chapter 12

Comparison and Conclusions

In this final chapter, the semantic structure developed in this thesis is first compared with a number of semantic structures for other approaches. Section 12.1 presents a comparison of semantic structures for the Concurrent MetateM framework and the semantic structure developed in this thesis. To illustrate the intended use of these semantic structures, the Concurrent MetateM framework itself is described and briefly compared with the DESIRE modelling framework. Section 12.2 presents a comparison of the semantics of Object Specification Logic and the semantic structure developed in this thesis. Section 12.3 discusses the relationship between so-called local model semantics for the representation of contextual reasoning and the semantic structure developed in this thesis. Section 12.4 briefly discusses relationships with a number of other frameworks. Directions for further research are sketched in Section 12.5. Final conclusions are drawn in Section 12.6. Please note that the description of OSL in Section 12.2 is taken from (Eck, Engelfriet, Fensel, Harmelen, Venema & Willems, in press).

12.1 Concurrent MetateM

Concurrent MetateM (Fisher, 1995a) stems from research in the field of executable temporal logic (Fisher & Owens, 1995) and the MetateM programming language (Barringer, Fisher, Gabbay, Owens & Reynolds, 1996). Concurrent MetateM extends the MetateM language with constructs for concurrent programming. One of the primary applications of Concurrent MetateM is analysis and high-level executable specification of multi-agent systems. The aim and scope of Concurrent MetateM is in a number of respects comparable to the aim and scope of the DESIRE modelling framework, as is the use of temporal logic. For this reason, a relatively extensive comparison between the two frameworks and the semantic structures used to described their semantics are presented in this section. A more detailed comparison between Concurrent MetateM and DESIRE can be found in (Mulder, Treur & Fisher, 1998).

In this section, first the *two frameworks* themselves are compared. This comparison focuses on the structure of agents, inter-agent communication and meta-level reasoning. A simple example multi-agent system is then described

using Concurrent MetateM and the semantic structure developed in this thesis. Finally, *semantic structures* for Concurrent MetateM are compared to the semantic structure developed in this thesis.

12.1.1 Structure of Agents, Communication and Meta-Level Reasoning

In a DESIRE specification of a multi-agent system, the agents of the multi-agent system are (usually) subcomponents of a top-level component that represents the whole (multi-agent) system, together with one or more components that represent the rest of the environment. A component that represents an agent is most often a composed component. The compositional structure of the component represents a hierarchy of agent processes. In a Concurrent MetateM model, agents are modelled as objects that have no further structure at all: all subprocesses of an agent are represented as one process, described by one set of temporal rules. In Concurrent MetateM, the environment is usually not explicitly modelled, although it is possible to introduce separate objects that represent the environment.

In DESIRE, the knowledge structures used in the knowledge bases and for the input and output interfaces of components, are defined in terms of information types, in which sort hierarchies can be defined. Information types define sets of ground atoms. Each component has an internal state, and all input and output interfaces have states. The states of components and links evolve over time. Components have the input persistence property (see Chapter 6): if, for a specific state of a component or link, a ground atom is e.g. true, than this is also the case for the next state, unless the state has changed because of updating an information link has been updated.

Concurrent MetateM does not have information types, there is no predefined set of atoms, and there are no sorts. The input and output interfaces of an object consist only of the names of predicates, called environment predicates and component predicates, respectively. Two-valued logic is used with a closed world assumption, thus a state is defined by the set of atoms that are true. Moreover, no persistency assumption is used: if an atom that is true in a specific state is not explicitly declared true in a next state, then it will be automatically false, irrespective of its previous truth value.

Communication between agents in DESIRE is defined by the information links between them. Communication between agents in Concurrent MetateM is done by broadcast message passing as follows. An agent continuously computes new truth values for its component predicate atoms (output atoms). These new values are implicitly broadcast to all other agents where they become the new truth values for the environment predicate atoms (input atoms) with the same name. Thus, when an object sends a message, it can be received by all other objects. On top of this, both multi-cast and point-to-point message passing can be defined. A newer version of DESIRE which is not discussed in Chapter 9 also provides support for broadcast communication. As an aside, let it be noted that in Concurrent MetateM,

the specification of an agent may use atoms that are not part of the agent's interface. These atoms are internal atoms and are not visible for other agents.

DESIRE automatically lifts input and output atoms to a meta-level representation and provides standard information types to express statements about lifted atoms. In DESIRE, meta-reasoning is modelled by using separate components for the object and the meta-level. For example, one component can reason about the reasoning process and state of another component. Two types of interaction between object- and meta-level are distinguished: upward reflection (from object- to meta-level) and downward reflection (from meta- to object-level).

For meta-reasoning in Concurrent MetateM, the logic MML has been developed. In MML, the domain over which terms range has been extended to incorporate the names of object-level formulae. Execution of temporal formulae can be controlled by executing them by a meta-interpreter. These meta-facilities have, to the best of the author's knowledge, not been implemented in the Concurrent MetateM execution environment.

12.1.2 Example

The following example shows the specification of an example multi-agent system in Concurrent MetateM. The example also shows how the multi-agent system can be described using the semantic structure developed in this thesis.

Example 12.1. A small multi-agent system used as an example in (Fisher & Wooldridge, 1997) serves as a running example for this section. The system is described in the quote below taken from (Fisher & Wooldridge, 1997). In Figure 12.1, the system is specified in Concurrent MetateM notation (although a different notation for the temporal operators is used). The first line of each box is an agent declaration, consisting of the name of the agent, its input propositions (between parentheses), and its output propositions (between square brackets). In each box, the agent declaration is followed by Concurrent MetateM rules in temporal logic describing the behaviour of the agent. The temporal operators **F** and **Y** are to be read as 'sometime in the future' and 'at the previous moment', respectively. The following quotation from (Fisher & Wooldridge, 1997) introduces the example multi-agent system:

> "A common form of multi-agent system is based upon the idea of distributed problem solving. Here, we consider a simple abstract distributed problem solving system, in which a single agent, called *top*, broadcasts a problem to a group of problem solvers. Some of these problem solvers can solve the problem completely, and some will reply with a solution. We define such a Concurrent MetateM system in [Figure 12.1]. Here, *solvera* can solve a different problem from the one *top* poses while *solverb* can solve the desired problem, but does not annouce the fact (as solution1 is not a component predicate for *solverb*); *solverc* can solve the problem posed by *top*, and will *eventually* reply with a solution."

"(…), we will verify some properties of the above system in [Section 6.2 of (Fisher & Wooldridge, 1997)]. We will also consider the refinement of individual agents, (e.g., a single problem-solver) into groups of agents with the same properties."

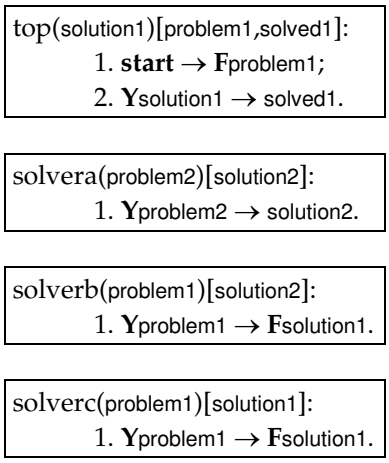

Figure 12.1: A distributed problem solving system (from Fisher & Wooldridge, 1997).

To formally define the multi-agent system presented in Figure 12.1 using the semantic structure developed in this thesis, first an additional component toplevel (not included in (Fisher & Wooldridge, 1997)) is introduced. This additional component represents the multi-agent system as a whole. Communication channels in a multi-agent are explicitly represented in the semantic structure, but not in Concurrent MetateM. In Concurrent MetateM, changes to environment predicates of the agents are broadcast, so all agents are able to receive them. To represent all possible information exchange, for the four agents in the example, twelve information links are needed. However, in the example, the agents only react to environment predicates of agent top, and top is the only agent that reacts to changes to environment predicates of the other agents. Therefore, six information links suffice to represent all information transmission that takes place in the example system: one from top to each other agent, and one from solvera, solverb and solverc to top. These links are named top_to_solvera, top_to_solverb, top_to_solverc, solvera_to_top, solverb_to_top, solverc_to_top. The example multi-agent system is represented by the structure hierarchy $sh_1=\langle Comp_1;Lnk_1;\prec_1;dom_1;cdom_1\rangle$ with:

- $Comp_1=\{$toplevel,top,solvera,solverb,solverc$\}$;
- $Lnk_1=\{$top_to_solvera,top_to_solverb,top_to_solverc,solvera_to_top,solverb_to_top, solverc_to_top$\}$;
- $\prec_1=\{\langle$top;toplevel$\rangle,\langle$solvera;toplevel$\rangle,\langle$solverb;toplevel$\rangle,\langle$solverc;toplevel$\rangle\} \cup \{\langle$solvera_to_top;toplevel$\rangle,\langle$solverb_to_top;toplevel$\rangle,\langle$solverc_to_top;toplevel$\rangle,\langle$top_to_solvera;toplevel$\rangle,\langle$top_to_solverb;toplevel$\rangle,\langle$top_to_solverc;toplevel$\rangle\}$;

- dom_1={⟨solvera_to_top;solvera⟩,⟨solverb_to_top;solverb⟩,⟨solverc_to_top;solverc⟩, ⟨top_to_solvera;top⟩,⟨top_to_solverb;top⟩,⟨top_to_solverc;top⟩};
- $cdom_1$={⟨top_to_solvera;solvera⟩,⟨top_to_solverb;solverb⟩,⟨top_to_solverc;solverc⟩, ⟨solvera_to_top;top⟩,⟨solverb_to_top;top⟩,⟨solverc_to_top;top⟩}.

The following component information state description signatures can be used to describe the states of the components that represent the agents. (See Definition 10.12 for the definition of component information state description signatures. As this section shows how the example presented in (Fisher & Wooldridge, 1997) can be represented with the semantic structure developed in this thesis (and not with DESIRE), the DESIRE constructs presented in Chapter 9 are not used.)

CONCURRENT METATEM AGENT DECLARATION	SIGNATURE
-	$\Sigma_{toplevel}$=⟨∅;∅;∅⟩;
top(solution1)[problem1,solved1]:	Σ_{top}=⟨{solution1};∅;{problem1,solved1}⟩;
solvera(problem2)[solution2]:	$\Sigma_{solvera}$=⟨{problem2};∅;{solution2}⟩;
solverb(problem1)[solution2]:	$\Sigma_{solverb}$=⟨{problem1};{solution1};{solution2}⟩;
solverc(problem1)[solution1]:	$\Sigma_{solverc}$=⟨{problem1};∅;{solution1}⟩.

In the example, the multi-agent system represented by toplevel does not interact with anything outside this component. Therefore, the input and output parts of $\Sigma_{toplevel}$ do not contain any proposition symbols. Moreover, in this example there is no need to define an internal part of $\Sigma_{toplevel}$,. ■

Contrary to DESIRE and the semantic structure developed in this thesis, Concurrent MetateM does not provide support for hierarchical composition. As a consequence, refinement of an agent in the model of a multi-agent system is represented by replacing the original model by its refinement. This is illustrated by a refinement of the example, again based on (Fisher & Wooldridge, 1997) to illustrate the structural aspects of components and information links. In their paper, Fisher and Wooldridge present a new system in which:

> "*solverc* is replaced by two agents who together can solve problem1, but cannot manage this individually. These agents, called *solverd* and *solvere* are defined in [Figure 12.2]. (…)Thus, when *solverd* receives the problem it cannot do anything until it has heard from *solvere*. When *solvere* receives the problem, it broadcasts the fact that it can solve part of the problem (i.e., it broadcasts solution1.2). When *solverd* sees

this, it knows it can solve the other part of the problem and broadcasts the whole solution." (Underlining by the author of this thesis.)

solverd(problem1,solution1.2)[solution1]:
1. (**Y**solution1.2 ∧ **P**problem1) → **F**solution1.

solvere(problem1)[solution1.2]:
1. **Y**problem1 → **F**solution1.2.

Figure 12.2: Refined Problem Solving Agents (from Fisher & Wooldridge, 1997)

Fisher and Wooldridge replace *solverc* by the two agents, *solverd* and *solvere*, which together perform the task of *solverc*. Using the semantic structure developed in this thesis, *solverc* is not replaced but refined; that is: *solverc* is decomposed into two subcomponents *solverd* and *solvere*. The refined agent *solverc* is depicted in Figure 12.3.

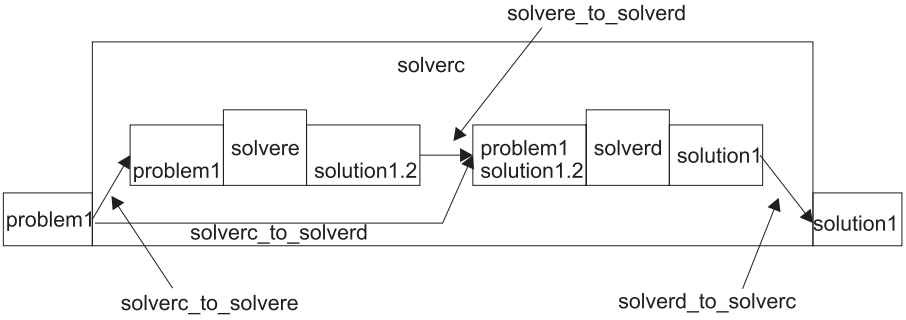

Figure 12.3: Refined Problem solving agents in graphical notation.

In Example 12.1, the compositional structure of the example multi-agent system was described by a structure hierarchy $sh_1=\langle Comp_1;Lnk_1;\prec_1;dom_1;cdom_1\rangle$. The refined system can be described by a structure hierarchy $sh_2=\langle Comp_2;Lnk_2;\prec_2;dom_2;cdom_2\rangle$ with:

- $Comp_2=Comp_1\cup\{$solverd,solvere$\}$;
- $Lnk_2=Lnk_1\cup\{$solverc_to_solvere,solverc_to_solverd,solvere_to_solverd,solverd_to_solverc$\}$;
- $\prec_2=\prec_1\cup\{\langle$solverd;solverc$\rangle,\langle$solvere;solverc$\rangle,\langle$solverc_to_solvere;solverc$\rangle,$ $\langle$solverc_to_solverd;solverc$\rangle,\langle$solvere_to_solverd;solverc$\rangle,\langle$solverd_to_solverc;solverc$\rangle\}$;
- $dom_2=dom_1\cup\{\langle$solverc_to_solvere;solverc$\rangle,\langle$solverc_to_solverd;solverc$\rangle,$ $\langle$solvere_to_solverd;solvere$\rangle,\langle$solverd_to_solverc;solverd$\rangle\}$;

- $cdom_2 = cdom_1 \cup \{\langle$solverc_to_solvere;solvere$\rangle, \langle$solverc_to_solverd;solverd$\rangle, \langle$solvere_to_solverd;solverc$\rangle, \langle$solverd_to_solverc;solverc$\rangle\}$.

For the components *solverd* and *solvere*, the following signatures are defined (the signature for *solverc* remains unchanged; it is repeated here for ease of reference):

CONCURRENT METATEM AGENT DECLARATION	SIGNATURE
solverc(problem1)[solution1]:	$\Sigma_{\text{solverc}} = \langle$problem1$\}; \emptyset; \{$solution1$\}\rangle$
solverd(problem1,solution1.2)[solution1]:	$\Sigma_{\text{solverd}} = \langle$problem1,solution1.2$\}; \emptyset; \{$solution1$\}\rangle$
solvere(problem1)[solution1.2]:	$\Sigma_{\text{solvere}} = \langle$problem1$\}; \emptyset; \{$solution1.2$\}\rangle$

12.1.3 Semantics

The semantics of Concurrent MetateM presented in (Fisher & Wooldridge, 1997) follows the commonly used possible-worlds semantics for modal (and, more specifically, temporal) logic. As a starting point for the semantics, the logical language available to the user for the specification of the behaviour of an individual agent (i.e., the language for the rules given in Figure 12.1) is extended with an epistemic modality. (The epistemic modality can only be applied to non-temporal formulas.) The resulting temporal logic is called TBL (Temporal Belief Logic). This logic is interpreted over time frames with a discrete order. The precise behaviour of the agents is constrained by a fixed set of temporal logic axioms. These axioms state several properties of agent interactions, e.g. an agent only believes a proposition p if p is an output proposition of another agent and p was true at the previous moment in time. Fisher and Wooldridge explicitly require communication to be lossless with their axiom (14) in (Fisher & Wooldridge, 1997). The order-preserving transmission property is not imposed by Fisher and Wooldridge (that is, at least not formally. However, in Fisher (1995b) it is explicitly stated that this property is *not* assumed).

At least two alternative semantics for Concurrent MetateM are reported. In (Reynolds, 1995), a first-order variant of the semantics for Concurrent MetateM is developed. In (Fisher, 1995b), a dense-time semantics is presented. The dense time semantics is basically defined in the same way as the semantics presented in (Fisher & Wooldridge, 1997). However, the frames for the interpretation of temporal formulae consist of an (infinite) set of time points with a dense order. (In fact, the set of real numbers is used.) However, the use of a dense order for the time frames has far-reaching consequences: it is not possible to express a next moment in time to interpret the 'next'-modality, or a previous moment in time to interpret the 'previous'-modality. An alternative is to focus on *changes* of truth values between states. Barringer, Kuiper and Pnueli (1986) present a dense

temporal logic, called TLR (Temporal Logic of the Reals) which includes an 'until'-modality based on state changes. The dense-time semantics for Concurrent MetateM is, in fact, a translation of Concurrent MetateM to TLR. The main assumption applied in this translation is as follows: individual agents in a multi-agent system are themselves *discrete* systems, the (discrete) behaviour of which occurs in densely ordered time, possibly simultaneously with discrete behaviour of other agents. A predicate act is introduced to represent their discrete behaviour, for each agent, which alternates between intervals in which it is true and false. Only if act is true, the truth values of other agent predicates may change. Consecutive intervals in which act is true define a 'next' relation for the agent.

Similar to standard temporal logic, dense time temporal logic does not have inherent support for true concurrency. However, dense-time temporal logic is viewed as a better approximation of true concurrency than discrete-time temporal logic for three reasons. First, real time is also densely ordered, so dense time frames are less artificial than discrete time frames. Second, it is possible to model concurrent actions as actions that occur infinitely close to one another. Third, and most important, in many dense-time temporal logic, all behaviour is modelled as consisting of intervals of (real) time. This enables modelling concurrent actions as overlapping intervals, possibly with the start of both intervals infinitely close to one another, and possibly likewise for the end of the intervals.

A number of differences between the semantic structure developed in this thesis and the approaches followed in the semantics of Concurrent MetateM can be distinguished:

- Both the discrete and dense time versions of the semantics of Concurrent MetateM assume that global time is available. As a result, all observers of a multi-agent system necessarily observe exactly the same behaviour in terms of the order of global states of the multi-agent system. The semantic structure developed in this thesis does not assume that global time is available. Different observers may observe different orders of global state, as explained in Chapter 7;
- Both the discrete and dense time versions of the semantics of Concurrent MetateM describe the behaviour of a multi-agent system in terms of global states of the entire multi-agent system. There is no support for locality;
- In Concurrent MetateM, only one view is provided, which completely describes the behaviour of a multi-agent system. Thus, this view is comparable to the glass box view. The semantic structure developed in this thesis provides three views on the behaviour of a multi-agent system;
- In the semantic structure developed in this thesis, the behaviour of compositions of components is described in terms of the behaviour of these components by incorporating the behaviour of these components in compatible multitraces. Multitraces retain the hierarchical structure of components. In Concurrent MetateM, the behaviour of compositions of

components is described by models of a global set of formulae, composed of the formulae that describe the behaviour of individual objects. Moreover, objects in Concurrent MetateM are not hierarchical, i.e., objects cannot consist of other objects.

12.2 Object Specification Logic (OSL)

Object Specification Logic (OSL, Sernadas, Sernadas & Costa, 1995) is a logic designed for the specification and analysis of object-oriented models of information systems. OSL is of interest for this thesis because separation of local and global reasoning has been an important requirement in the design of OSL. As a result, OSL supports locality, and, to a limited extent, compositionality. Locality and compositionality are important features of the semantic structure developed in this thesis. Moreover, OSL is state-based, as is the semantic structure developed in this thesis, and employs temporal logic, as does DESIRE. The description of OSL in this chapter is taken from (Eck, Engelfriet, Fensel, Harmelen, Venema & Willems, in press).

OSL consists of two levels: a local and a global level. The local level is concerned with the definition of the local state and behaviour of an object (the description of which is called an *aspect* of an object), specified by *aspect templates* in a local specification language. (For each aspect template, a different local language is defined.) At the global level, the different aspects are related, forming specialisations (to represent inheritance) and aggregations (to represent composite objects). In this section, the notation and terminology used in (Jungclaus, 1993) is adopted, which differs from the (more complex) notation in (Sernadas *et al.*, 1995). Section 12.2.1 presents the local specification language, used to specify object aspects. Section 12.2.2 describes the semantics of the local level. Section 12.2.3 proceeds to discuss morphisms, which connect the local and the global level. Section 12.2.4 concludes the description of OSL by discussing the global level and its semantics. Section 12.2.5 presents a brief summary of the preceding sections on OSL. Section 12.2.6 compares the semantics of OSL to the semantic structure developed in this thesis.

12.2.1 Local Syntax

At the local level, states and transitions are described using many-sorted first-order local temporal predicate languages. These languages are defined using (local) signatures $\Theta=\langle \mathcal{LS};\alpha;ATT;EVT\rangle$. A local signature consists of a partially ordered set[4] $\mathcal{LS}$ of identifier sorts (sorts containing object identifiers), a distinguished sort $\alpha\in\mathcal{LS}$ (the local sort), which is used for identifying instances of the class (aspect)

[4] Actually, $\mathcal{LS}$ is the Cartesian category freely generated from the partial order. In this section, OSL is presented without reference to the category structure of the sorts, similar to the description in (Jungclaus, 1993).

specified by Θ, a set of attribute symbols ATT and a set of event symbols EVT. Based on a signature Θ, a set of predicates $\mathcal{P}(\Theta)$ is defined, consisting of predicates $\triangleleft e(x_1,\ldots,x_n)$ and $\otimes e(x_1,\ldots,x_n)$ for every event symbol $e \in EVT$, and $\triangleleft a(x_1,\ldots,x_n)$ for every attribute symbol $a \in ATT$. The intended meaning of the predicates $\triangleleft e(x_1,\ldots,x_n)$ and $\triangleleft a(x_1,\ldots,x_n)$ is that event e is enabled (i.e., *can* occur (it does not have to occur) in the current state) and that a certain value for attribute a is observable. The predicate $\otimes e(x_1,\ldots,x_n)$ indicates that event e occurs in the current state. From these predicates, a local specification language is defined, with the usual logical connectives and three temporal operators: $\bigcirc$ ('next'), $\Box$ ('always') and $\Diamond$ ('sometimes'). In this local language, predicates are localised by prefixing them with a variable with as sort the sort α. A *local specification* is a set of formulas (the local axioms) in this language. State transitions are described by axioms that define admissible behaviour by relating current and future states.

12.2.2 Local Semantics

Similar to the syntax of OSL, semantically local and global states are distinguished. The local interpretation of formulas assumes the existence of a fixed data universe (an algebra of data types), providing a carrier set $A(s)$ for each sort s. Formulas in the local specification language are interpreted over local interpretation structures that consist of a family of carrier sets for the sorts used and a sequence of states $(\sigma_k)_{k \in N}$ (discrete linear time). States are sets of predicates that describe which observations are possible, which events are enabled, and which events actually occur in that state. Formally, states are elements of the set of all possible local states $\Pi = \{p(x_1,\ldots,x_n) \mid p \in \mathcal{P}(\Theta), x_i \in A(s_i) \text{ for } i=1,\ldots,n\}$. A sequence of states has additional semantic constraints, among which is the following frame assumption: only the occurrence of an event can change the set of observables and enabled events (i.e., only an event occurrence can change a state). Another constraint is that attributes are functional in OSL: if $\triangleleft a(x) \in \sigma$ and $\triangleleft a(x') \in \sigma$ for some state σ and for some $a \in ATT$, then $x=x'$. As a consequence, attributes are interpreted functionally, as usual.

12.2.3 Template Morphism Syntax: Bridge Between Local and Global Level

As stated before, objects can be composed to form complex objects. In OSL, this is specified using *template morphisms* to compose complex signatures from which formulae describing the complex objects are generated. A template morphism σ: $\Theta \rightarrow \Theta'$ between two signatures $\Theta = \langle IS; \alpha; ATT; EVT \rangle$ and $\Theta' = \langle IS'; \alpha'; ATT'; EVT' \rangle$, is a tuple $\langle \sigma_{IS}; \omega_\alpha; \sigma_{ATT}; \sigma_{EVT} \rangle$, with σ_{IS} a mapping of identifier sorts, σ_{ATT} and σ_{EVT} mappings for the attribute and event symbols, respectively, and ω_α: $\alpha' \rightarrow \sigma_{IS}(\alpha)$ an operator used to distinguish kinds of morphisms in a way that is not relevant here. There are two kinds of morphisms: *inclusion* of Θ in Θ' and *injection* of Θ in Θ'. With the former, additional signature elements can be added to Θ, making

inclusion morphisms suitable for modelling specialization. With the latter, Θ can be incorporated in a more complex signature, making it suitable to model aggregations of objects in composed objects. Using template morphisms, different local signatures can be combined. The resulting signatures are used to compose formulae in the global language.

12.2.4 Global Language and Semantics

The global language is based on a signature that consists of all local signatures, generated by inclusion and injection template morphisms. The language defined over this signature consists of formulas of the form $\varphi \Rightarrow \psi$, where φ and ψ are local-language interaction formulas over different signatures. Using this language, relations between (the behaviour of) objects are described. In a formula $\varphi \Rightarrow \psi$, the occurrence of events described in φ implies the simultaneous occurrence of the events described in ψ ('event calling'). The semantics of the global language is similar to the semantics of local languages. (The global language is generated from the global signature in the same way as local languages are generated from local signatures. The semantics of the global language can therefore be defined in the same way as the semantics of the local language.) The global language is thus interpreted over sequences of states, each of which is a set of predicates of the global signature. These states are global states.

There is, however, one difference between the global level and the local level. The global signature consists of local signatures, related by template morphisms. The semantics of the global language not only consists of sequences of states generated from the global signature, but also of an indexed set of all local interpretation signatures. Similar to the semantic structure developed in this thesis, only combinations of local and global interpretation structures that respect specific constraints are allowed. (See (Sernadas, Sernadas & Costa, 1995, Definition 5.4, second bullet, clause 2, page 621. In this thesis, these constraints are modelled by compatibility relations.) However, in OSL, local interpretation structures are only related to the global interpretation structure, which needs to be defined to interpret formulae in the global language. In the semantic structure developed in this thesis, compatibility relates local component and link traces, and no global traces are necessary.

12.2.5 Intuition Behind OSL

Consider a global language formula $\varphi \Rightarrow \psi$. Both φ and ψ are formulae of the global signature, containing subformulae that refer to different objects in the system. At the global level, with formulae like $\varphi \Rightarrow \psi$, it is only possible to formulate expressions like 'if something happens in the life of an object referred to in φ, then something else has to happen in the life of an object referred to in ψ at the same moment'. For such an expression to have a meaning, it is necessary to specify the

lives of both objects. This is done by the full temporal logic subformulae of φ and ψ.

12.2.6 Comparison

The semantics of OSL resembles the semantic structure developed in this thesis in its attention for locality. As in the semantic structure developed in this thesis, OSL starts from local perspectives on the behaviour of parts of a system (objects in the case of OSL, components in the case of the semantic structure developed in this thesis). A number of differences, however, can be identified:

- OSL constructs a global language to describe the behaviour of a system as a whole, which is not the case for the semantic structure developed in this thesis. (As OSL is a language, the only way to relate different local languages is to define a global language.)
- OSL supports compositionality only to a limited extent. In OSL, it is possible to specify objects that consist of other objects (aggregation). However, the behaviour of a composite object is not described in terms of the behaviour of its constituent objects.
- In OSL, only two levels of locality are distinguished: the object level and the system level. In the semantic structure developed in this thesis, three views on the behaviour of (components in) a compositional system are distinguished, which correspond to three levels of locality. As these views can be applied to any component at any level in a compositional system, many different levels of locality can be distinguished.
- The semantics of OSL is defined in terms of global states and implies that a notion of global time is available, which is not the case for the semantic structure developed in this thesis.

Interaction is modelled in OSL in a way that resembles interaction in the semantic structure developed in this thesis. In OSL, interaction is modelled by *event calling* as explained above. Event calling is a form of implication: the occurrence of an event described by φ in the event calling formula $\varphi \Rightarrow \psi$ implies the occurrence of an event described by ψ in another object. This is similar to the interpretation of information link mappings as described in Chapter 6: the occurrence of specific states in a local component trace of the domain of a link implies the occurrence of specific states in a local component trace of the co-domain of the link.

12.3 Local Model Semantics for Contextual Reasoning

An important topic in the study of common-sense reasoning in Artificial Intelligence is the formalisation of *contextual reasoning*. In this section, a formalisation of contextual reasoning developed by Giunchiglia and Ghidini (1998)

is compared to the semantic structure developed in this thesis. Contextual reasoning is related to multi-agent systems as follows. An agent's observations are often constrained, which precludes the agent from obtaining a complete model of its environment. An agent may be aware of the fact that it (temporarily) cannot obtain observations of a specific aspect of its environment. In this case, the absence of information on this aspect can be modelled using epistemic logic or three-valued logic. However, an agent may not even be aware of the *existence* of specific aspects of its environment. Its view of its environment is local in the sense that its view cannot accommodate knowledge on the aspects of which it is not aware. The agent reasons about its environment from the context consisting of (partial or complete) knowledge of those aspects of which it is aware. This is called the *principle of locality* in (Giunchiglia & Ghidini, 1998).

In their formalisation, Giunchiglia and Ghidini (1998) only consider objective information about the environment available to agents in their context (subjective interpretations are not considered). As several agents reason about the same environment, parts of the different agents' objective information overlap. The overlapping parts of different contexts must match, as they are local views of the same part of the environment. This is called the *principle of compatibility* in (Giunchiglia & Ghidini, 1998).

Giunchiglia and Ghidini (1998) present a semantics that can be used to capture contextual reasoning according to the principles of locality and compatibility. Their approach is related to the semantic structure presented in this thesis, as locality and compatibility are the basic principles employed in both approaches.

As a starting point, Giunchiglia and Ghidini (1998) assume that local reasoning within a context is described by (first-order) local languages, a (possibly different) language L_i for each agent i. Similar to the languages presented in Chapter 9, each language only contains terms and predicates to describe the aspects that are known to exist in the local view. Associated with each local language L_i is a class of standard first-order interpretation structures $\underline{M}_i$. The formulae that describe the knowledge available in a specific local view i denote a set of local models $M_i \subseteq \underline{M}_i$ that classically satisfy those formulae.

Giunchiglia and Ghidini (1998) introduce (non-temporal) compatibility relations as relations on the sets of local models. Similar to (Giunchiglia & Ghidini, 1998), in this description of local model semantics, all definitions are presented for two agents. (It is straightforward to extend the definitions to the general case for more than two agents.) A compatibility relation is defined as a relation $\boldsymbol{C} \subseteq \mathcal{P}(\underline{M}_1) \times \mathcal{P}(\underline{M}_2)$. ($\mathcal{P}(S)$ denotes the power set of S). For convenience, Giunchiglia and Ghidini use the notation $\boldsymbol{c}_i$ for the i-th element of a tuple $\boldsymbol{c} \in \boldsymbol{C}$. A compatibility relation $\boldsymbol{C}$ is called a *model* for $\{L_1, L_2\}$ if $\boldsymbol{C} \neq \emptyset$ and $\langle \emptyset; \emptyset \rangle \notin \boldsymbol{C}$. A *context* C_i is defined as the set of local models in the compatibility relation, or formally, the set of local models $m \in \underline{M}_i$ such that $m \in \boldsymbol{c}_i$ for $\boldsymbol{c} \in \boldsymbol{C}$. In other words, a context is the

set of local models that take overlap with other context as modelled by a compatibility relation, into account.

The framework described in (Giunchiglia & Gidini, 1998) is very general. Compatibility relations are not further defined as specific (proper) subsets of $\mathcal{P}(\underline{M_1})\times\mathcal{P}(\underline{M_2})$ (e.g., for specific applications). The role of compatibility relations is to model constraints on local models imposed by non-local dependencies. The constraints modelled by compatibility have, in the words of Giunchiglia and Ghidini (1998), "the structural effect of changing the set of local models defining each context. It forces local models to agree up to a certain extent". This structural effect is made explicit in the notion of satisfiability of a (local) formula:

Definition 12.2. (Satisfaction (Giunchiglia & Gidini, 1998), slightly different notation). *Let $\boldsymbol{C}$ be a model and let $\varphi\in L_i$ be a formula. Then $\boldsymbol{C}$* satisfies *φ iff for all $\boldsymbol{c}\in\boldsymbol{C}$, $\boldsymbol{c}_i$ satisfies φ, where $\boldsymbol{c}_i$ satisfies φ iff for all $m\in\boldsymbol{c}_i$, m classically satisfies φ.*

Thus, local models m that do not occur in pairs in the compatibility relation $\boldsymbol{C}$ cannot satisfy φ.

It is interesting to conclude the description of local model semantics with Giunchiglia and Ghidini's notion of logical consequence in contextual reasoning, which is the foundation for reasoning about agents that maintain contextual views of their environment:

Definition 12.3. (Logical consequence (Giunchiglia & Gidini, 1998), slightly different notation). *Let Γ_1 and Γ_2 be sets of formulae of L_1 and L_2, respectively. A formula $\varphi\in L_i$ is a* logical consequence *of a set of formulae $\Gamma=\Gamma_1\cup\Gamma_2$ iff for all models $\boldsymbol{C}$ and for all $\boldsymbol{c}\in\boldsymbol{C}$, if for all $j\in\{1,2\}$, $j\neq i$, $\boldsymbol{c}_j$ satisfies Γ_j, then for all $m\in\boldsymbol{c}_i$, if m classically satisfies Γ_i, then m classically satisfies φ.*

The basic principles locality and compatibility in the semantic structure developed in this thesis are very similar to the notions of locality and compatibility employed by Giunchiglia and Ghidini. The examples and discussions of compatibility relations in (Giunchiglia & Gidini, 1998) indicate that compatibility relations, in their approach, are constructed in the same way as compatibility relations and constructs in the semantic structure presented in this thesis. Moreover, compatibility relations have exactly the same role: to model constraints on local models (local component and link traces in this thesis) imposed by non-local dependencies (information transmission in this thesis). However, a number of differences between local model semantics and the semantic structure developed in this thesis can be identified:

- The approach presented in (Giunchiglia & Ghidini, 1998) only covers static aspects of contextual reasoning. However, in this thesis, the principles of locality and compatibility relations are extended for dynamic domains and applied to the dynamics of multi-agent systems.

- As the definition of compatibility presented above indicates, Giunchiglia and Ghidini relate *sets* of local models in compatibility relations. (A compatibility relation is a subset of $\mathcal{P}(\underline{M_1}) \times \mathcal{P}(\underline{M_2})$, not of $\underline{M_1} \times \underline{M_2}$.) Contexts consist of sets of local models, which enables the representation of partial information. As Definition 5.18 indicates, a compatibility relation consists of triples of single local component or link traces. Partiality may be modelled by using partial states inside local component and link traces (as in the DESIRE modelling framework presented in Chapter 9);
- Compatibility in local model semantics model constraints imposed by overlapping parts of local models. In the semantic structure developed in this thesis, local component and link traces do not overlap. A local component or link trace only consists of states of a single component or link. However, information transmission introduces similar constraints between local models as the overlap of local models in contextual reasoning, giving rise to a notion of (temporal) compatibility;

12.4 Other Approaches

The semantic structure developed in this thesis can be compared with many more approaches in various areas of research. A number of suggestions for further comparison are presented:

- Within the area of multi-agent systems, there is considerable attention for the specification of multi-agent system dynamics. A number of approaches are of particular importance to consider for a comparison with the semantic structure developed in this thesis. Kiss (1996) presents a very general overview of agent dynamics and its relations with traditional Distributed Artificial Intelligence topics such as planning and rationality. A semantics for an abstract agent programming language (for single agents) is presented in (Hindriks, Boer, Hoek & Meyer, 1998). In (Eijk, Boer, Hoek & Meyer, 1998), information exchange in a multi-agent system is formalised. Situation calculus (McCarthy & Hayes, 1969) is used for the specification of (multi-)agent systems using the Congolog framework in (Lespérance, Levesque, Lin, Marcu, Reiter & Scherl, 1996; Lespérance, Levesque & Ruman, 1997). Burkhard (1993) studies liveliness and safety properties in multi-agent systems. Burkhard's (1993) approach is entirely event-based: the behaviour of a single agent is modelled by a set of strings of events, which constitutes a formal language. Liveliness and safety properties for a multi-agent system are analysed in terms of operations on formal languages.
- Chapter 7 presented a detailed comparison with the representation of concurrency in distributed systems based on the notion of dependence (Lamport, 1978). Candidates for further comparison are more algebraic approaches such as event structures (Winskel, 1989) or Chu spaces

(Pratt, 1995), or Petri net based approaches such as (Moldt & Weinberg, 1997). Ladkin and Leue (1995) and Damm and Harel (1999) study the relation between event-based and state-based descriptions of dynamics in the context of timing diagrams or message sequence diagrams. As Ladkin and Leue observe, diagrams similar to Figure 2.9 and Figure 7.4 play an important role in various research areas, including Hardware Design, Computer Networks (Tanenbaum, 1996) and Object Orientation (e.g., message sequence charts in the UML (Booch, Rumbaugh & Jacobson, 1998)).

- Semantic structures for co-ordination languages (Papadopoulos & Arbab, 1998; Ciancarini & Wolf, 1999) form another category of interesting approaches for comparison with the semantic structure developed in this thesis. As stated in Chapter 2, most co-ordination languages completely abstract from computations in a compositional system by only distinguishing a number of components, each of which contains a number of computational processes. Co-ordination languages focus on the specification of systems of such components and information exchange between them, which is similar to the semantic structure developed in this thesis. The semantical description of Manifold (Bonsangue, Arbab, Bakker, Rutten, Scutellà & Zavattaro, 1998) is of particular interest as it has a number of characteristics in common with the semantic structure developed in this thesis. In Manifold, communication channels between components are explicitly modelled. Moreover, these channels have their own state (similar to commitment presented in Section 2.2.3) and are autonomous.

12.5 Further Research

The semantic structure developed in this thesis provides a ground for further research in a number of areas. First, the semantic structure can be extended, i.e. by including additional facilities for the specification of real-time behaviour (see also Section 12.5.1), process creation, or other phenomena that have been studied in for instance the context of Process Algebra (Bergsta & Klop, 1985). Second, verification and validation of multi-agent systems represented as compositional systems using the semantic structure can be investigated. Formal verification requires a formal proof system. Section 12.5.2 sketches how two existing logics can possibly be used for reasoning about properties of multi-agent systems in the context of the semantic structure. As an aside, validation and verification of properties of multi-agent systems do not require the use of mechanical proof creation or checking. In (Jonker, Treur & Vries, 1998), an approach is presented for the verification of multi-agent systems using rigorous mathematical reasoning. As the semantic structure developed in this thesis is itself defined in mathematical terms, the approach described in (Jonker, Treur & Vries, 1998) may be particularly suitable. However, this form of verification is difficult to support directly by mechanical means, such

as for instance a proof checker for general first-order logic. Third, some common topics in concurrency theory, such as notions of equivalence of behaviour, can be investigated in the context of the semantic structure. In Section 12.5.3, some directions for such investigations are briefly sketched. Fourth, additional applications of the semantic structure can be investigated. In Section 12.5.4, further research with respect to the DESIRE modelling framework is sketched.

12.5.1 Real-time Logics and Fictitious Clocks

Standard temporal logic can be used to express qualitative statements about time, but not to express quantitative statements. Real-time logics are extensions of temporal logic that support quantitative statements about time. Most real-time logics are either based on dense time (as presented in Section 12.1.3) or on fictitious clocks (Raskin & Schobbens, 1997). In the fictitious clocks approach, a global fictitious clock is introduced that is assumed to generate clock ticks at a fixed rate. The clock ticks are represented in the sequence of discrete, global states of a system. Time is measured by counting the number of states between two states in which a clock tick occurs. When dense-time temporal logic is used as a real-time logic, a metric on the dense set of time points (e.g., the standard metric on the set of real numbers) is used for quantitative statements about time.

In (Raskin & Schobbens, 1997), the relation between the dense time and fictitious clocks approaches to real-time logics is investigated. Raskin and Schobbens (1997) present one temporal logic language with two different interpretations (dense time and fictitious clocks). The most imporant connective in the logic is a special form of until, which is parameterised by an amount of time. For instance, $\phi \mathcal{U}_{\leq 3} \varphi$ is true at a time point t iff ϕ is true at t and stays true for at most three units of time, after which φ is true.

The fictitious clock interpretation of their language is based on models consisting of a discrete sequence of global states. A global state itself is a set of proposition symbols that are true in that state. In a subset of these states, the proposition symbol tick is true, which indicates that a tick of the fictitious clock occurs between that state and the next one. A formula $\phi \mathcal{U}_{\leq 3} \varphi$ is true at state s in such a model iff ϕ is true in state s and in the sequence of states between s and the state in which ϕ is no longer true, but φ is, at most three states in which 'tick' is true appear.

Raskin and Schobbens do not introduce a formal framework with a specific computational model, such as DESIRE or Concurrent MetateM. Instead, they only introduce the temporal logic mentioned above. However, their work is intended for the specification of properties of computer systems (as stated on page 166 of their paper). The dense time interpretation of the language is therefore based on models that consist of consecutive intervals of real time in which the (global) state of a system remains constant. The union of these intervals is required to cover an

unbounded amount of time, which ensures that there cannot be an infinite number of intervals in which the state is distinct in a finite amount of time (non-Zenoness).

The semantic structures and logic language presented in (Raskin & Schobbens, 1997) are standard. The main result of the paper is a precise definition of a relationship between dense time and fictitious clock semantics. Using this relation, approximate answers to satisfiability questions for the dense time semantics (which is undecidable) can be found: dense time formulae are abstracted to fictitious clock interpretation, which is decidable. This approach can be employed at the local level in the semantic structure to mechanically determine whether specific local component or link traces satisfy a local formula.

Many differences can be distinguished between the approach by Raspin and Schobbens, and the semantic structure developed in this thesis. On the one hand, both the dense time semantics and in the fictitious clocks semantics assume that global time exists. Moreover, the approach by Raskin and Schobbens is based on a notion of global state. Locality, compositionality and hierarchical composition are not treated at all. On the other hand, the approach by Raskin and Schobbens provides much more detail with respect to the representation of real-time systems than the semantic structure presented in this thesis. Also the DESIRE modelling framework presented in Chapter 9 has no built-in facilities for expressing real-time requirements.

A possible further research question is: can the approach of Raspin and Schobbens be used to specify sets of local behaviour of individual components, and if so, is this beneficial? The definitions of local component and link traces presented in Chapter 5 enable the use of dense time or fictitious clock real time logic: for the time frames on which these traces are based, dense time or fictitious clock structures as described in (Raskin & Schobbens, 1997) can be used. A logic language for real time, such as the logic presented in (Raskin & Schobbens, 1997), could be used to specify the sets of local component and link traces.

12.5.2 Verification

The formally defined semantic structure presented in this thesis enables precise, mathematical proofs of properties of multi-agent systems. It is often desirable to mechanically verify or generate such proofs, which requires a formal logic specifically suited for the semantic structure developed in this thesis. The application of the semantic structure in the context of DESIRE presented in Chapter 9 shows how temporal logic can be used for the specification of local behaviour. The application of formal logic for reasoning at a more global level is an area for further research. Two approaches that seem to be particularly suitable for this purpose are Multi-Context Systems (Giunchiglia, 1993; Giunchiglia & Serafini, 1994) and Interleaving Set Temporal Logic (ISTL; Katz & Peled, 1990), which are both described below.

12.5.2.1 Multi-Context Systems

A Multi-Context System (Giunchiglia, 1993; Giunchiglia & Serafini, 1994), also called a Multi-Language System, is a set of (possibly different) logic languages, together with bridge rules which relate formulae in these languages. The use of different logic languages is the same as in Local Model Semantics (discussed in Section 12.3): each language is a local language that describes a specific context. A bridge rule is an inference rule in which the language of the condition of the rule is not the same as the language of the conclusion of the rule. A bridge rule states that if a formula is proven that matches the condition of the bridge rule in a derivation in one context, then the conclusion of the rule can be introduced as an assumption in a derivation in another context. Bridge rules are not the same for all Multi-Context System applications; instead they differ from system to system, depending on the application.

The Multi-Context Systems approach is a likely candidate for a formal logic that enables reasoning at the global level in applications of the semantic structure for the following reasons:

- Similar to the semantic structure developed in this thesis, the Multi-Context Systems approach focuses on locality and compositionality. In a Multi-Context System, only local languages are defined, and only local proofs can be derived. Proofs of global properties are composed of these local proofs;
- Multi-Context Systems may be considered the syntactical counterpart of Local Model Semantics (Giunchiglia & Gidini, 1998) as presented in Section 12.3, although Giunchiglia and Gidini (1998) do not investigate the relationship between Multi-Context Systems and Local Model Semantics.

The Multi-Context Systems approach is supported by the proof checker GETFOL (Giunchiglia, 1994,1993). The availability of a proof checker contributes greatly to the attractiveness of the Multi-Context Systems approach for the semantic structure developed in this thesis. However, as yet there are no research results with respect to the application of the Multi-Context Systems approach for the semantic structure. In this section, only a sketch is provided of how an application of the Multi-Context Systems approach for proving properties of multi-agent systems could be envisioned.

Figure 12.4 shows a derivation of the property $\boldsymbol{start} \rightarrow \boldsymbol{F}\mathsf{solved1}$ for the distributed problem solving system used in the discussion of Concurrent MetateM in the first section of this chapter. (In (Fisher & Wooldridge, 1997), the same property is proven using a Hilbert-style temporal logic.) Please take note that the derivation shown in Figure 12.4 is tentative and most probably unsound. The derivation is only presented to sketch the flavour of an application of the Multi-Context Systems approach. The notation used in Figure 12.4 is the same as in (Giunchiglia, 1993).

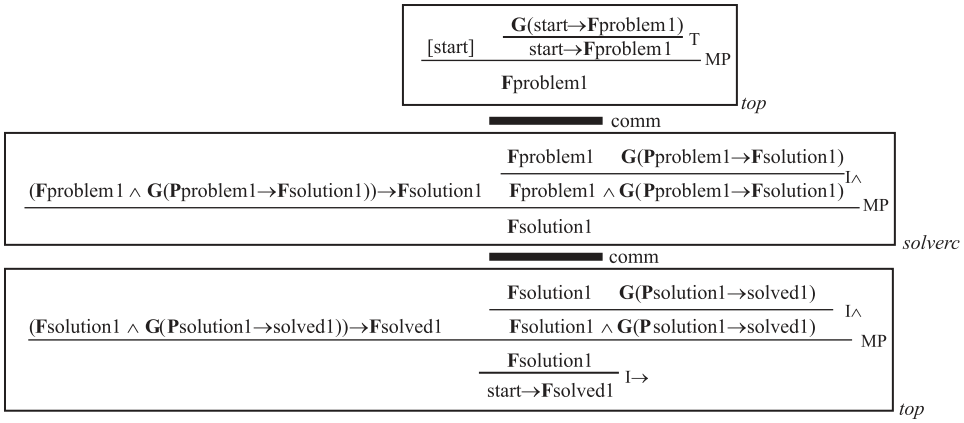

Figure 12.4: Tentative sketch of a derivation of ***start***$\rightarrow$***F***solved1.

In the derivation shown in Figure 12.4, each box corresponds with a component: both boxes labelled *top* correspond to component *top*, and the box labelled *solverc* corresponds to component *solverc*. Each box contains a sequent calculus deduction using only propositional symbols defined for the corresponding component. In other words, each box contains a derivation in the context of its corresponding component, using the local language of that component. In the derivations, three kinds of open assumptions can be distinguished:

- Assumptions of the form $\boldsymbol{G}(\rho)$, where ρ is a rule occuring in the specification of the corresponding component. (See Figure 12.1 for the specification of components *top* and *solverc*.) These assumptions can be considered to be axioms which state that in the component, all rules by which it is specified are always applicable;
- Assumptions of the form $(\boldsymbol{F}\varphi \wedge \boldsymbol{G}(\boldsymbol{P}\varphi\rightarrow\psi))\rightarrow\boldsymbol{F}\psi$ and $(\boldsymbol{F}\varphi \wedge \boldsymbol{G}(\boldsymbol{P}\varphi\rightarrow\boldsymbol{F}\psi))\rightarrow\boldsymbol{F}\psi$, where $\boldsymbol{P}\varphi\rightarrow\psi$ is a rule in the specification of the corresponding component. Assumptions of this form, in general, express *local* axioms of computation such as, e.g., liveliness. For instance, an axiom of the form $(\boldsymbol{F}\varphi \wedge \boldsymbol{G}(\boldsymbol{P}\varphi\rightarrow\psi))\rightarrow\boldsymbol{F}\psi$, where $\boldsymbol{P}\varphi\rightarrow\psi$ is a rule, states that if eventually the condition of the rule is true, then the conclusion will also eventually be true. Other assumptions in this category may for instance be frame axioms, or, in the context of Concurrent MetateM's TBL (see Section 12.1.3), axioms of the underlying epistemic logic. Further research is needed to identify which axioms are relevant in the context of multi-agent systems;
- Assumptions introduced by bridge rules, such as for example ***F***problem1 in the box labelled *solverc*. In the derivation shown in Figure 12.4, only one bridge rule is applied, called *comm* and denoted by a heavy, solid line in the figure. Bridge rules differ for specific applications of the multi-context systems approach. For the semantic structure developed in this thesis, a

possible bridge rule may state that if an information link exists from a component C to a component D that links φ to ψ, then if $\boldsymbol{F}\varphi$ is derived in component C, $\boldsymbol{F}\psi$ can be introduced as an assumption in a derivation in the context of component D. Bridge rules in this application of the Multi-Context Systems approach are closely related to properties of information transmission as presented in Chapter 6, and to information link mapping descriptions, which specify how states in various components are related.

The tentative derivation presented in Figure 12.4 suggest an approach for proving properties of a multi-agent system using temporal local model semantics. The basic intuition that forms the basis of this approach is as follows. As the notion of compatibility relations does not enforce any relation on the local clocks of components C and D, proofs have to be based on local properties of components that state that relevant events *eventually* happen. Bridge rules ensure that these relevant events *actually* happen if, as proven for another component, a related event happens in that component. It remains to be investigated whether this is sound and sufficient for proving all valid properties.

12.5.2.2 ISTL—Interleaving Set Temporal Logic

Another likely candidate for a formal logic that enables proving properties of compositional systems is Interleaving Set Temporal Logic (ISTL; Katz & Peled, 1990). Syntactically, ISTL is similar to branching-time temporal logics such as CTL* (Emerson & Halpern, 1986). However, the interpretation of ISTL differs from most other temporal logics: in ISTL, the two sources of non-determinism found in concurrent systems are not identified, as explained below.

In a concurrent system in which no global clock is assumed, two sources of non-determinism can be distinguished. First, specific processes in a concurrent system may be non-deterministic with respect to their reactions to information received from other processes. Although it is not known beforehand which reaction will be chosen by the process, all observers of the process observe the same reaction after a choice is made. Second, due to the absence of global time, different observers observe different behaviour of the same process even if this process always reacts in the same way. In most temporal logics, both forms of non-determinism are represented in exactly the same way: as branching points in their interpretation structures. As a consequence, the two forms of non-determinism cannot be distinguished in these logics. Moreover, in these logics, the behaviour of one concurrent system is described by one branching-time interpretation structure.

In ISTL, both forms of non-determinism are distinguished. The behaviour of one concurrent system is described in ISTL by a *set* of branching-time interpretation structures. Each different branching-time interpretation structure in this set represents a different reaction to information received by a process in the system. Each branching point in one specific branching-time interpretation

structure represents non-determinisn with respect to observations due to the absence of global time.

Interpretation structures of ISTL are similar to the global view on the behaviour of a compositional system presented in Chapter 7. As explained in Chapter 7, global states for one specific multitrace are partially ordered. The partial order of global states represents non-determinism of observations caused by the absence of global time. If a component in a compositional system is non-deterministic with respect to its reactions to information received from other components, the different behavious are represented by different multitraces, which leads to different sets of partially ordered global states.

It may be possible to prove that the class of interpretation structures formally defined in (Katz & Peled, 1990) for ISTL is exactly the same as the class of partially ordered sets of strict global states defined in Chapter 7. If this is the case, than ISTL is suitable as a logic to reason about global properties of a compositional system. If this is not the case, further research may try to adapt ISTL's inference rules to the semantic structure developed in this thesis.

A specific question that has to be answered in the development of formal logic systems for the semantic structure is the question of fairness (Francez, 1986; Manna & Pnueli, 1992). In fact, it may come as a surprise that fairness is not discussed in this thesis, as specific semantic structures often commit to a specific notion of fairness. However, in this thesis, the view of OSL is adopted, which states that fairness is a property of an *application* of the semantic structure and may differ for different applications. In the words of the developers of OSL: "The question of fairness was completely disregarded in this paper. Indeed, we assume that the task of imposing fairness or justice requirements is left to the specifier (to the point of inconsistency if by mistake too much is required)." (Sernadas, Sernadas & Costa, 1995, p. 627).

12.5.3 Concurrency Theory

As stated in Section 1.4.5, this thesis does not make any claim with respect to the applicability of the semantic structure as a general theory of concurrency. However, the question whether the semantic structure is applicable outside the area of multi-agent systems may be interesting for further research. A general topic in Concurrency Theory is equivalence of computations. For instance, in Process Algebra (Bergstra & Klop, 1985) approaches, a process and its desired properties are both specified using a process algebra. The question whether a property holds for the process is reduced to the question whether the computations denoted by the two specifications are equivalent. The development of equivalence notions for process algebra specifications is therefore an important research area. It would be interesting to develop similar notions for the semantic structure presented in this thesis.

12.5.4 Applications of the Semantic Structure

Some directions for further research can be identified in the area of applications of the semantic structure:

- The semantic structure fully abstracts from specific agent processes such as co-operation and maintenance of mental notions such as beliefs, desires, intentions and commitments. The specification of, for instance, a BDI architecture for a single agent, or of inter-agent commitments in a multi-agent system, raises interesting questions with respect to the application of the semantic structure. In Chapter 10 and Chapter 11, applications in these areas are presented. In these applications, the behaviour of agents does not change during the lifetime of agents. Further research may investigate how the semantic structure can support adaptive behaviour and learning;
- In the context of DESIRE, at least two directions for further research can be identified. First, DESIRE has recently been extended with support for dynamic adaptation of agent models and for additional forms of information transmission such as broadcasting. Further research may extend the formal description of DESIRE presented in Chapter 9 to include these new facilities. Second, some research questions with respect to implementation generators for DESIRE can be identified. To support the design process for multi-agents systems using DESIRE, an implementation generator has been developed. Currently, the implementation generator executes automatically generated prototype implementations of multi-agent systems on a single processor using pseudo-concurrency. To do full justice to the specifications a, prototype implementation should ideally run in a distributed environment. In the near future, the implementation generators will be augmented, providing support for concurrent execution. A possible research question is: is it possible to develop or validate such an implementation generator directly from the description of the DESIRE behaviour as presented in Chapter 9?

12.6 Conclusions

The aim of the research presented in this thesis is formulated in Chapter 1 as the development of a formal, compositional, semantic structure for multi-agent systems dynamics. Section 1.4.3 lists four requirements for the semantic structure: (1) both *deliberation and interaction* in a multi-agent system should be explicitly represented, (2) the semantic structure should support the *compositionality principle*, (3) an agent's dynamics should be described in terms of its *state and state transformations*, and (4) the semantic structure should support *locality*. This final section of the thesis describes the extent to which these requirements have been fulfilled.

The research presented in this thesis is based on a central assumption, introduced in Chapter 1: multi-agent systems are represented as compositional systems. As a consequence, the main building blocks of the semantic structure are components and information links between components. Studies in the context of DESIRE have shown that multi-agent systems can successfully be represented as compositional systems (Brazier, Dunin-Keplicz, Jennings & Treur, 1997; Brazier, Eck & Treur, 1997b,2001a; see also the concluding remarks with respect to DESIRE presented in Section 9.4). Moreover, existing generic representations (models) of multi-agent systems can be (re)used for agents and components of agents, such as the generic agent model presented in (Brazier, Jonker & Treur, 2000). In Chapter 3, modelling choices with respect to how a multi-agent system can be represented as a compositional system, are discussed. Chapter 2 discussed compositional systems and presented commitments with respect to properties of compositional systems that can be described using the semantic structure developed in this thesis.

The main constructs that comprise the semantic structure are mathematically defined in Chapter 5. The static compositional structure of components is described by *structure hierarchies*. Three views on the behaviour of a compositional system (described by a structure hierarchy) are presented: the *black box view*, the *white box view*, and the *glass box view*. Each view is a set of compatible *multitraces*. A multitrace is an indexed set of *local component and link traces* of (a subset of) the components and links in the compositional system, in which a partial order on the index set represents the compositional structure of the system. A local component or link trace consists of *local component or link states*. A local state is fully determined by one single component and link (hence the name). For each component and link, a set of local component traces is assumed to be given. Such a set contains all local component and link traces that describe possible behaviour without taking constraints imposed by information transmission into account. *Compatibility relations* group triples consisting of a local link trace for an information link I, a local component trace of the domain of I, and a local trace of the co-domain of I. Such triples are related by compatibility if the local traces in the triple respect constraints imposed by information transmission, e.g., if a state occurs in a local component trace of the domain in which information has to be transmitted, then a related state occurs in the co-domain of the link in which the transmitted information has just been received. A *compatible multitrace* contains only local component and link traces that occur in compatibility relations. Properties of information transmission such as the lossless transmission property are captured as properties of compatibility relations. The definitions of these properties demonstrate how compatibility relations relate local component traces according to *information link mappings*.

The four requirements for the semantic structure described in Chapter 1 and mentioned at the beginning of this section are met to the following extent:

- The first requirement (deliberation and interaction) is met to the following extent. Local component traces describe processes within primitive

components that use information received from other components and produce information intended to be transmitted to other components. Local link traces and compatibility relations describe the information transmission processes needed to actually transmit information. As explained in Chapter 3, an agent's deliberation is represented by internal processes inside components, while an agent's interaction with other agents and with its environment is represented by information transmission. The semantic structure thus supports both kinds of agent activities;

- With respect to the second requirement (the compositionality principle), it is important to note that the adjective 'compositional' refers to three different but related notions in the literature. These notions are supported by the semantic structure developed in this thesis in the following way:
  - First, the adjective 'compositional' simply denotes that a system consists of components. First and foremost, the semantic structure focuses on compositional systems: systems consisting of components and information links between components. Components and information links are the two most important constructs that constitute the semantic structure. In Chapter 2, compositional systems are (informally) defined, commitments with respect to properties of compositional systems adopted in this thesis are presented and alternative properties are described. The notion of a compositional system is also discussed from the perspective of areas of research such as Software Engineering and Co-ordination Languages;
  - Second, the adjective 'compositional' is used in the following way: a compositional system is a system in which the dynamics of a system composed of a number of components is defined by a composition relation. The composition relation itself defines the dynamics of a system composed of a number of components in terms of the dynamics of those components. The definitions of the three views on the behaviour of a composed component presented in Chapter 5 (the black box, glass box and white box views) clearly indicate that compositional systems in the semantic structure adopt this compositionality property: each view is built from local component and link traces or from multitraces that describe the behaviour of lower level components. Moreover, in Chapter 5, propositions are presented that specify how more global views can be composed of local views on the behaviour of a component;
  - Third, the adjective 'compositional' may apply to proofs of properties of a system that itself may or may not be compositional (in the previous meanings). In the preface to (Roever *et al.*, 1998), the term 'compositional method' refers to "Any method by which the properties of a system can be inferred from the properies of its constituents, without additional information about the internal structure of the constituents". This

meaning of 'compositional' is not directly applicable to the semantic structure developed in this thesis, as the semantic structure is not equipped with a formal logic. However, the focus of the semantic structure on other notions of compositionality most probably will facilitate the development of compositional proof methods for the semantic structure. Foster (1996) connects this view on the term 'compositional' with the previous one as follows: "A compositional programming system is one in which properties of program components are preserved when those components are composed in parallel with other program components." Foster's perspective on compositionality is difficult to combine with the property of locality in a hierarchical system, as properties of lower-level components need to be expressible at higher levels for Foster's perspective to apply.

In the semantic structure, not only primitive components, but also composed components are associated with a notion of local behaviour. This local behaviour is, constrained by information transmission, one of the building blocks of the actual behaviour of a component. It can be used for management of its input and output state and for accumulation and generalisation of information provided by subcomponents. This local behaviour of a composed component is in general completely independent of the behaviour of its subcomponents;

- According to the third requirement described in Chapter 1, the semantic structure should be state-based. As indicated in Chapter 1, agents in a multi-agent system are often described and analysed in terms of their mental state. Therefore, the dynamics of a multi-agent system should be described first and foremost in terms of state and state transformation, as opposed to actions or events. The semantic structure as presented in Chapter 5 and Chapter 6 is entirely defined in terms of states and traces consisting of states. Nevertheless, as described in Chapter 7, (mostly global) event-based notions for the description of dynamics, such as the notions presented in (Lamport, 1986; Charron-Bost *et al.*, 1996) are applicable within the state-based semantic structure;
- The fourth requirement states that the semantic structure should support locality. Locality is at the heart of the semantic structure. The basic units for the representation of dynamics are local component and link traces, which consist of local component and link states. The black box view and white box view provide different levels of locality with respect to the description of the behaviour of composed components. Global views are defined (in the form of the glass box view defined in Chapter 5 as well as the notion of global state defined in Chapter 7). These global views are defined in terms of local views and not the other way around. Commitments presented in Chapter 2 with respect to compositional systems also ensure that

information is localised in components. Only information in the input and output interfaces of components is visible to other components. Moreover, this information is only visible to a limited set of agents: as the restrictions on information transmission presented in Chapter 2 indicate, only links between a component and the parent components and subcomponents of the parent component are allowed. As a result, information is kept as local as possible: information can only be distributed in a compositional system via parent components that may control which information is distributed to other components.

The semantic structure as presented in Chapter 5 and Chapter 6 provides all facilities needed to represent the control phenomenon. Chapter 8 describes the phenomenon of control in multi-agent systems and compositional systems in more depth. Some constructs of the semantic structure introduced in Chapter 5 and Chapter 6, however, needed to be refined in Chapter 8 to facilitate the representation of control in a *separated*, *domain-independent* way.

Chapters 9, 10 and 11 show how the semantic structure can be applied for different purposes. In Chapter 9, the semantic structure is used to describe the dynamics of models of multi-agent systems in the DESIRE modelling framework. In Chapter 10 and Chapter 11, two example multi-agent systems modelled using DESIRE are presented. The example systems show how the dynamics of complex multi-agent systems can be described by the semantic structure and how locality and compositionality help to reduce the complexity of such systems.

As indicated in Section 12.5, there are different directions for further research. The ultimate goal of all such research should be to meet the promise of multi-agent systems sketched in Chapter 1. The research presented in this thesis shows that the semantic structure is a first step towards rigorous, formal design and analysis of multi-agent systems necessary to reach this goal.

Bibliography

Abraham, U., Ben-David, S., and Magidor, M. (1990). On global time and inter-process communication. In Kwiatkowska, M., Shields, M., and Thomas, R., editors, *Semantics for Concurrency, Leicester 1990*, Workshops in Computing, pages 311-323. Springer-Verlag.

Axelrod, R. M. (1997). *The Complexity of Cooperation. Agent-Based Models of Competition and Collaboration*. Princeton University Press.

Barringer, H., Fisher, M., Gabbay, D., Owens, R., and Reynolds, M., editors (1996). *The Imperative Future: Principles of Executable Temporal Logic*, volume 4 of Advanced Software Development Series, Research Studies Press, Taunton.

Barringer, H., Kuiper, R., and Pnueli, A. (1986). A really abstract concurrent model and its temporal logic. In *Conference Record of the 13th ACM Symposium on the Principles of Programming Languages (POPL)*, pages 173-183, New York. ACM Press.

Bass, L., Clements, P., Kazman, R., and Bass, K. (1998). *Software Architecture in Practice*. SEI Series in Software Engineering. Addison-Wesley.

Bergsta, J. A. and Klop, J.-W. (1985). Algebra of communicating processes with abstraction. *Theoretical Computer Science*, 37:77-121.

Birkhoff, G. (1933). On the combination of subalgebras. *Proc. Cambridge Phil. Soc.*, 29:441-464.

Bonsangue, M. M., Arbab, F., Bakker, J. W. de, Rutten, J. J. M. M., Scutellà, A., and Zavattaro, G. (1998). *A transition system semantics for the control-driven coordination language MANIFOLD*. Report SEN-R9829, Centrum voor Wiskunde en Informatica.

Booch, G., Rumbaugh, J., and Jacobson, I. (1998). *The Unified Modeling Language User Guide*. Addison-Wesley.

Brazier, F. M. T., Cornelissen, F., Gustavsson, R., Jonker, C. M., Lindeberg, O., Polak, B. C. G., and Treur, J. (2000). A multi-agent system performing one-to-

many negotiation for load balancing of electricity use. *International Journal of Electronic Commerce*.

Brazier, F. M. T., Cornelissen, F., Jonker, C. M., and Treur, J. (2000). Compositional specification of a reusable co-operative agent model. *International Journal of Cooperative Information Systems*, 9:171-207.

Brazier, F. M. T., Dunin-Keplicz, B. M., Jennings, N. R. and Treur, J. (1997). DESIRE: modelling multi-agent systems in a compositional formal framework. In Huhns, M. and Singh, M., editors, *International Journal of Cooperative Information Systems, special issue on Formal Methods in Cooperative Information Systems: Multiagent Systems*, 6(1):67-94.

Brazier, F. M. T., Dunin-Keplicz, B. M., Treur, J., and Verbrugge, L. C. (1999). Modelling internal dynamic behaviour of BDI agents. In (Meyer & Schobbens, 1999). pages 36-56.

Brazier, F. M. T., Eck, P. A. T. van, and Treur, J. (1996). Design of a modelling framework for multi-agent systems. In Albrecht, R. and Herre, H., editors, *Trends in Theoretical Informatics*, Band 89 in Schriftenreihe der Österreichischen Computer Gesellschaft, pages 173-191, R. Oldenbourg, Wien, München. ISBN: 3-486-23809-4 (München), 3-7029-0414-X (Wien), 3-85403-089-4 (Österr. Computer Ges.)

Brazier, F. M. T., Eck, P. A. T. van, and Treur, J. (1997a). Modelling a society of simple agents: from conceptual specification to experimentation. In Conte, R., Hegselmann, R., and Terna, P., editors, *Modelling Social Phenomena*, volume 456 of Lecture Notes in Economics and Mathematical Systems, pages 103-109. Springer-Verlag, Berlin.

Brazier, F. M. T., Eck, P. A. T. van, and Treur, J. (1997b). Modelling competitive co-operation of agents in a compositional multi-agent framework. In Plaza, E. and Benjamins, R., editors, *Knowledge Acquisition, Modelling and Management. Proceedings of the 10th European Knowledge Acquisition Workshop, EKAW'97*, volume 1319 of Lecture Notes in Artificial Intelligence, pages 317-322. Springer-Verlag, Berlin.

Brazier, F. M. T., Eck, P. A. T. van, and Treur, J. (2001a). Modelling a society of simple agents: From conceptual specification to experimentation. *Applied Intelligence*, 14(2):161-178.

Brazier, F. M. T., Eck, P. A. T. van, and Treur, J. (2001b). Semantic formalisation of emerging dynamics of compositional agent systems. In Gabbay, D. and Smets, P., editors, *Agent-Based Defeasible Control in Dynamic Environments*, volume 7 of *Handbook Series on Defeasible Reasoning and Uncertainty Management Systems (DRUMS)*. Kluwer Academic Publishers.

Brazier, F. M. T., Jonker, C. M., Jungen, F. J., and Treur, J. (1999). Distributed scheduling to support a call centre: A co-operative multi-agent approach. *Applied Artificial Intelligence Journal*, 13:65-90. H. S. Nwana and D. T. Ndumu, editors, Special Issue on Multi-Agent Systems.

Brazier, F. M. T., Jonker, C. M. and Treur, J. (1997). Formalization of a cooperation model based on joint intentions. In (Müller, Wooldridge & Jennings, 1997) pages 141-155.

Brazier, F.M.T., Jonker, C. M., and Treur, J. (1998). Principles of compositional multi-agent system development. In Cuena, J., editor, *Proceedings of the 15th IFIP World Computer Congress, WCC'98, Conference on Information Technology and Knowledge Systems, IT&KNOWS'98*, pages 347-360.

Brazier, F. M. T., Jonker, C. M., and Treur, J. (2000). Compositional design and reuse of a generic agent model. *Applied Artificial Intelligence Journal*, 14:491-538.

Brazier, F. M. T., Jonker, C. M., Treur, J. and Wijngaards, N. J .E. (1998a). Compositional design of a generic design agent. In Luger, G. and Interrante, L., editors, *Proceedings of the AAAI Workshop on Artificial Intelligence and Manufacturing: State of the Art and State of Practice*, pages 30-39. AAAI Press, Menlo Park.

Brazier, F. M. T., Jonker, C. M., Treur, J., and Wijngaards, N. J. E. (2000). Deliberate evolution in multi-agent systems. In Gero, J., editor, *Proceedings of the Sixth International Conference on AI in Design, AID'2000*. Kluwer Academic Publishers.

Brazier, F. M. T. and Ruttkay, Zs. (1993). Modelling collective user satisfaction. In *Proceedings of HCI International'93*, pages 672-677, Elsevier, Amsterdam.

Brazier, F. M. T., Treur, J., and Wijngaards, N. J. E. (1996). Modelling interaction with experts: the role of a shared task model. In Wahlster, W., editor, *Proceedings of the European Conference on Artificial Intelligence, ECAI'96*, pages 241-245. Wiley and Sons, Chichester.

Brazier, F. M. T., Treur, J., Wijngaards, N. J. E., and Willems, M. (1999). Temporal semantics of tasks models and problem solving methods. *Data and Knowledge Engineering*, 29(1):17-42.

Briot, J.-P. and Gasser, L. (1998). Agents and concurrent objects. Jean-Pierre Briot interviews Les Gasser. *IEEE Concurrency*, 6(4).

Burkhard, H.-D. (1993). Liveness and fairness properties in multi-agent systems. In Bajcsy, R., editor, *Proceedings of the 13th International Joint Conference on Artificial Intelligence, IJCAI'93*, pages 325-330. Morgan Kaufmann, San Mateo.

Carriero, N. and Gelernter, D. (1992). Coordination languages and their significance. *Communications of the ACM*, 35(2):97-107.

Castelfranchi, C. (1995). Guarantees for autonomy in cognitive agent architecture. In (Wooldridge & Jennings, 1995a), pages 56-70.

Castelfranchi, C. and Conte, R. (1996). Distributed artificial intelligence and social science: Critical issues. In (O'Hare & Jennings, 1996), chapter 20.

Castellano, L. Michellis, G. de, and Pomello, L. (1987). Concurrency vs. interleaving: an instructive example. *Bulletin of the EATCS*, 31:12-15, February.

Cesta, A., Miceli, M., and Rizzo, P. (1996a). Effects of different interaction attitudes on a multi-agent system performance. In (Velde & Perram, 1996), pages 128-138.

Cesta, A., Miceli, M., and Rizzo, P. (1996b). Help under risky conditions: Robustness of the social attitude and system performance. In Tokoro, M., editor, *Proceedings of the Second International Conference on Multi-Agent Systems, ICMAS'96*, pages 18-25, AAAI Press, Menlo Park.

Chaib-draa, B. and Vanderveken, D. (to appear). On the success and satisfaction of speech acts for computational agents. In Vanderveken, D. and Kudo, S., editors, *Essays in Speech Act Theory*. Benjamin, Amsterdam/Philadelphia, in press.

Chandrasekaran, B. (1994). Understanding control at the knowledge level. AAAI Fall Symposium. http://www.cis.ohio-state.edu/~chandra/intelligent-control.pdf.

Chandrasekaran, B. (1999). Homepage. http://www.cis.ohio-state.edu/~chandra/

Chandrasekaran, B., Johnson, T. R., and Smith, J. W. (1992). Task-structure analysis for knowledge modeling. *Communications of the ACM*, 35(9):124-137.

Chandrasekaran, B., Josephson, J. R. and Benjamins, V. R. (1998). The ontology of tasks and methods. In Gaines, B. and Musen, M., editors, *Proceedings of the 11th Banff Knowledge Acquisition for Knowledge-Based Systems Workshop, KAW'98*, University of Calgary.

Charron-Bost, B., Mattern, F., and Tel, G. (1996). Synchronous, asynchronous, and causally ordered communications. *Distributed Computing*, 9:173-191.

Chomsky, N. (1988). *Language and Problems of Knowledge: the Nicaraguan Lectures*. MIT Press.

Ciancarini, P. and Wolf, A. L., editors (1999). *Coordination Languages and Models*, volume 1594 of Lecture Notes in Computer Science. Springer-Verlag, Berlin.

Clancey, W. J. (1983). The advantages of abstract control knowledge in expert system design. In *Proceedings AAAI-83*, pages 74-78. AAAI Press.

Clancey, W. J. (1992). Model construction operators. *Artificial Intelligence*, 53:1-115.

Cockburn, D. and Jennings, N. R. (1996). ARCHON: A distributed artificial intelligence system for industrial applications. In (O'Hare & Jennings, 1996), chapter 12.

Damm, W. and Harel, D. (1999). LSCs—breathing life into message sequence charts. In *Third International Conference on Formal Methods for Open Object-Based Distributed Systems, FMOODS'99*. IFIP TC6/WG6.1.

Davey, B. A. and Priestley, H. A. (1992). *Introduction to Lattices and Order.* Cambridge Mathematical Textbooks.

Dennett, D. C. (1984). *Elbow Room: The Varieties of Free Will Worth Wanting*. Clarendon Press.

D'Souza, D. F. and Wills, A. C. (1998). *Objects, Components, and Frameworks with UML. The Catalysis Approach.* Addison-Wesley.

Eck, P. A. T. van (1998). AI and computer simulation in the social sciences—a mutually beneficial relationship? *NVKI Nieuwsbrief*, 15(1):9-10.

Eck, P. A. T. van, Engelfriet, J., Fensel, D., Harmelen, F. van, Venema, Y. and Willems, M. (1998). Specification of dynamics for knowledge-based systems. In: Freitag, B., Decker, H., Kifer, M., and Voronkov, A., editors, *Transactions and Change in Logic Databases*, volume 1472 of Lecture Notes in Computer Science, pages 37-68. Springer-Verlag, Berlin.

Eck, P. A. T. van, Engelfriet, J., Fensel, D., Harmelen, F. van, Venema, Y. and Willems, M. (in press). A survey of languages for specifying dynamics: A knowledge engineering perspective. *IEEE Transactions on Knowledge and Data Engineering*. Also appeared as: Technical Report IR-447, Vrije Universiteit, Faculty of Sciences, Amsterdam. Shorter version appeared as: (Eck, Engelfriet, Fensel, Harmelen, Venema & Willems, 1998).

Eijk, R. van, Boer, F. S. de, Hoek, W. van der, and Meyer, J.-J. C. (1998). Systems of communicating agents. In (Prade, 1998), pages 293-297.

Emerson, E. A. and Halpern, J. Y. (1986). "Sometimes" and "not never" revisited: On branching versus linear time temporal logic. *Journal of the ACM*, 33(1):151-178.

Engelfriet, J. (1999). *The Dynamics of Reasoning*. Ph.D. Thesis, Vrije Universiteit Amsterdam.

Engelfriet, J., Jonker, C. M. and Treur, J. (1999). Compositional verification of multi-agent systems in temporal multi-epistemic logic. In (Müller, Singh & Rao, 1999), pages 177-194.

Engelfriet, J. and Treur, J. (1994). Temporal theories of reasoning. In: MacNish, C., Pearce, D. and Pereira, L. M., editors, *Logics in Artificial Intelligence. Proc. of the 4th European Workshop on Logics in Artificial Intelligence, JELIA '94*, volume 838 of

Lecture Notes in Artificial Intelligence, pages 279-299. Springer-Verlag, Berlin. Also in: *Journal of Applied Non-Classical Logics*, 5(2):239-261, 1995.

Fagin, R., Halpern, J. Y., Moses, Y., and Vardi, M. Y. (1995). *Reasoning about Knowledge*. MIT Press.

Finin, T., Labrou, Y. and Mayfield, J. (1997). KQML as an agent communication language. In Bradshaw, J., editor, *Software Agents*, MIT Press.

[FIPA] *FIPA '97 Specification*. http://www.fipa.org/.

Fisher, M. (1995a). Representing and executing agent-based systems. In (Wooldridge & Jennings, 1995a), pages 307-323.

Fisher, M. (1995b). Towards a semantics for Concurrent MetateM. In (Fisher & Owens, 1995), pages 86-102.

Fisher, M. and Owens, R., editors (1995). *Executable Modal and Temporal Logics*, volume 897 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag.

Fisher, M. and Wooldridge, M. (1997). On the formal specification and verification of multi-agent systems. *International Journal of Cooperative Information Systems*. 6(1):37-65.

Foster, I. (1996). Compositional parallel programming languages. *ACM Transaction on Programming Languages and Systems*, 18(4):454-476.

Francez, N. (1986). *Fairness*. Springer-Verlag.

Franklin, S. and Graesser, A. (1997). Is it an agent, or just a program?: A taxonomy for autonomous agents. In (Müller, Wooldridge & Jennings, 1997), pages 21-35.

Fromentin, E. and Raynal, M. (1994). Local states in distributed computations: a few relations and formulas. *ACM Operating Systems Review*, 28(2):65-72.

Fromentin, E. and Raynal, M. (1995). Characterizing and detecting the set of global states seen by all the observers of a distributed computation. In *Proceedings of the 15th International IEEE Conference on Distributed Computer Systems*, pages 431-438. IEEE Computer Society Press.

Gamma, E., Helm, R., Johnson, R., and Vlissides, J. (1995). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

Gaspari, M. (1998). Concurrency and knowledge-level communication in agent languages. *Artificial Intelligence*, 105:1-45.

Gavrila, I. S. and Treur, J. (1994). A formal model for the dynamics of compositional reasoning systems. In: Cohn, A. G., editor, *Proceedings of the 11th European Conference on Artificial Intelligence, ECAI'94*, pages 307-311. Wiley and Sons. Extended version: Report IR-323, Vrije Universiteit Amsterdam, Department of Mathematics and Computer Science.

Gaylford, R. J. and D'Andria, L. J. (1998). *Simulating Society: A Mathematical Toolkit for Modelling Socioeconomic Behavior*. Springer-Verlag.

Giunchiglia, F. (1993). Contextual reasoning. *Epistemologia*, special issue on I Linguaggi e le Macchine, XVI:345-364. Short version in *Proceedings IJCAI'93 Workshop on Using Knowledge in its Context*, Chambéry, France, 1993, pp. 39-49. Also IRST-Technical Report 9211-20, IRST, Trento, Italy.

Giunchiglia, F. (1994). *GETFOL manual—GETFOL version 2.0*. Technical Report 92-0010, Università di Genova, Dipartimento di Informatica Sistemistica e Telematica.

Giunchiglia, F. and Serafini, L. (1994). Multilanguage hierarchical logics (or: how we can do without modal logics). *Artificial Intelligence*, 65:1-42.

Giunchiglia, F. and Ghidini, C. (1998). Local models semantics, or contextual reasoning = locality + compatibility. In Cohn, A. G., Schubert, L. K., and Shapiro, S. C., editors, *Proceedings of the Sixth International Conference on Principles of Knowledge Representation and Reasoning (KR'98)*, pages 282-291. Morgan Kaufmann.

Glabbeek, R. van and Goltz, U. (1989). Partial order semantics for refinement of actions—neither necessary nor always sufficient but appropriate when used with care. *Bulletin of the EATCS*, 38:154-163, June.

Gray, R. S., Cybenko, G., Kotz, D., and Rus, D. (1997). Agent Tcl. In Cockayne, W. and Zypa, M., editors, *Itinerant Agents: Explanations and Examples with CDROM*. Manning.

Hanks, S., Pollack, M., and Cohen, P. (1993). Benchmarks, testbeds, controlled experimentation, and the design of agent architectures. *AI Magazine*, 14(4):17-42.

Hindriks, K. V., Boer, F. S. de, Hoek, W. van der, and Meyer, J.-J. C. (1998). A formal semantics for an abstract agent programming language. In (Singh, Rao & Wooldridge, 1998), pages 215-230.

Hoare, C. A. R. (1978). Communicating sequential processes. *Communication of the ACM*, 21(8):666-677.

Iglesias, C. A., Garijo, M., González, J. C. and Velasco, J. R. (1998). Analysis and design of multiagent systems using MAS-CommonKADS. In (Singh, Rao & Wooldridge, 1998), pages 313-328.

Iglesias, C. A., Garijo, M. and González, J. C. (1999). A survey of agent-oriented methodologies. In (Müller, Singh & Rao, 1999), pages 317-330.

Jennings, N. R. (1999). Agent-oriented software engineering. In F. J. Garijo and M. Boman, editors, *Multi-Agent System Engineering. Ninth European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW'99*, volume 1647 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin.

Johansen, D., van Renesse, R., and Schneider, F. B. (1997). Operating system support for mobile agents. In Huhns, M. N. and Singh, M. P., editors, *Readings in Agents*. Morgan Kaufmann.

Jonker, C. M., Lam, R. A., and Treur, J. (1999). A multi-agent architecture for an intelligent website in insurance. In Klusch, M., Shehory, O., and Weiss, G., editors, *Cooperative Information Agents III. Proceedings of the Third International Workshop on Cooperative Information Agents, CIA'99*, volume 1652 of Lecture Notes in Artificial Intelligence, pages 86-100. Springer-Verlag, Berlin.

Jonker, C. M. and J. Treur. (1997). Modelling an agent's mind and matter. In Boman, M. and Velde, W. van de, editors, *Multi-Agent Rationality. Proceedings of the Eighth European Workshop on Modelling Autonomous Agents in a Multi- Agent World, MAAMAW'97*, volume 1237 of Lecture Notes in Artificial Intelligence, pages 210-233. Springer-Verlag, Berlin.

Jonker, C. M. and Treur, J. (1998a). Compositional verification of multi-agent systems: A formal analysis of pro-activeness and reactiveness. In (Roever et al., 1998), pages 350-380.

Jonker, C. M. and Treur, J. (1998b). *Agent-Based Simulation of Animal Behaviour*. Centre for Mathematics and Computer Science, Amsterdam, Technical Report, pp. 33.

Jonker, C. M. and Treur, J. (1998c). A generic architecture for broker agents. In Nwana, H. S. and Ndumu, D.T., editors, *Proceedings of the Third International Conference on the Practical Application of Intelligent Agents and Multi-Agent Technology, PAAM'98*, pages 623-624. The Practical Application Company Ltd.

Jonker, C. M., Treur, J. and Vries, W. de (1998). Compositional verification of agents in dynamic environments: A case study. In Harmelen, F. van, editor, *Proceedings of the KR'98 Workshop on Verification and Validation of Knowledge-Based Systems*. Trento, Italy, June 6-8, 1998.

Jonker, C. M., Treur, J. and Wijngaards, W. C. A. (2000). An executable model of the interaction between verbal and non-verbal communication. In Dignum, F., and Greaves, M., editors, *Issues in Agent Communication: Proceedings of the Agent Communication Languages Workshop 1999, ACL'99,* volume 1916 of Lecture Notes in Artificial Intelligence, pages 331-350. Springer-Verlag, Berlin.

Jungclaus, R. (1993). *Modeling of Dynamic Object Systems—A Logic-based Approach*. Advanced Studies in Computer Science. Vieweg.

Jungclaus, R., Saake, G., Hartmann, T., and Sernadas, C. (1996). TROLL—A language for object-oriented specification of information systems. *ACM Transactions on Information Systems*. 14(2):175-211.

Katz, S. and Peled, D. (1990). Interleaving set temporal logic. *Theoretical Computer Science*, 75:263-287.

Kiss, G. (1996) *Agent dynamics*. In (O'Hare & Jennings, 1996), chapter 9.

Kowalski, R. (1979). Algorithm=logic+control. *Communications of the ACM*, 22(7):424-436.

Kshemkalyani, A. D. (1998). Causality and atomicity in distributed computations. *Distributed Computing*, 11(4):169-189.

Ladkin, P. B. and Leue, S. (1995). Interpreting message flow graphs. *Formal Aspects of Computing*, 7(5):473-509.

Lamport, L. (1978). Time, clocks and the ordering of events in a distributed system. *Communications of the ACM*, 21(7):558-565.

Lamport, L. (1986). On interprocess communication, part I—basic formalism. *Distributed Computing*, 1:77-85.

Lamport, L. (1994). The temporal logic of actions. *ACM Transactions on Programming Languages and Systems*, 16(3):872-923.

Lamport, L. (1995). How to write a proof. *American Mathematical Monthly*, 102(7):600-608, August-September.

Langevelde, I. A. van, Philipsen, A. W., and Treur, J. (1992). Formal specification of compositional architectures. In Neumann, B., editor, *Proceedings of the 10th European Conference on Artificial Intelligence, ECAI'92*, pages 272-276. Wiley and Sons.

Langholm, T. (1988). *Partiality, Truth and Persistence*. CSLI lecture notes, number 15.

Lassila, O. (1998). Web metadata: a matter of semantics. *IEEE Internet Computing*, 2(4). See also: http://www.w3.org/RDF.

Lespérance, Y., Levesque, H. J., Lin, F., Marcu, D., Reiter, R., and Scherl, R. B. (1996). Foundations of a logical approach to agent programming. In (Wooldridge, Müller & Tambe, 1996), pages 331-346.

Lespérance, Y., Levesque, H. J., and Ruman, S. J. (1997). An experiment in using Golog to build a personal banking assistant. In Cavedon, L., Rao, A., and Wobcke, W., editors, *Intelligent Agent Systems—Theoretical and Practical Issues*, volume 1209 of Lecture Notes in Artificial Intelligence, pages 27-43, Springer-Verlag, Berlin.

Liskov, B., Moss, E., Schaffert, C., Scheifler, R., and Snyder, A. (1981). *CLU Reference Manual*. Springer-Verlag, New York.

Luck, M. and d'Inverno, M. (1995). A formal framework for agency and autonomy. In Lesser, V., editor, *Proceedings of the First International Conference on Multi-Agent Systems, ICMAS'95*. AAAI Press, Menlo Park, pages 254-260.

Malone, T. W. and Crowston, K. (1994). The interdisciplinary study of coordination. *ACM Computing Surveys*, 26(1):87-119.

Manna, Z. and Pnueli, A. (1992). *The Temporal Logic of Reactive and Concurrent Systems*, volume 1 (Specification). Springer, New York.

Mattern, F. (1992). On the relativistic structure of logical time in distributed systems. *Datation et Controle des Executions Reparties, Bigre 78*, pages 3-20. ISSN 0221-525. http://www.informatik.tu-darmstadt.de/VS/Publikationen/papers/relativistic_time.ps.

McCarthy, J. (1987). *Ascribing Mental Qualities to Machines*. Technical Report, Stanford University AI Lab, Stanford, CA 94305.

McCarthy, J. and Hayes, P. J. (1969). Some philosophical problems from the standpoint of Artificial Intelligence. In Meltzer, B. and Michie, D., editors, *Machine Intelligence 4*. Edinburgh University Press.

Meyer, J.-J. C. and Schobbens, P.-Y., editors (1999). *Formal Models of Agents*, volume 1760 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag.

Milner, R. (1980). *A Calculus of Communicating Systems*, volume 92 in Lecture Notes in Computer Science. Springer-Verlag, Berlin.

Moldt, D. and Weinberg, F. (1997). Multi-agent systems based on coloured petri nets. In Azema, P. and Balbo, G., editors, *Applications and Theory of Petri Nets*, volume 1248 of Lecture Notes in Computer Science, pages 82-101. Springer-Verlag.

Montgomery, T. A. and Durfee, E. H. (1990). Using MICE to study intelligent dynamic coordination. In *Proceedings of the IEEE Conference on Tools for Artificial Intelligence*, IEEE Computer Society Press.

Moss, S., Gaylard, H., Wallis, S. and Edmonds, B. (1998). SDML: A multi-agent language for organizational modelling. *Computational and Mathematical Organization Theory*, 4(1):43-70.

Mulder, M., Treur, J. and Fisher, M. (1998). Agent modelling in MetateM and DESIRE. In (Singh, Rao & Wooldridge, 1998), pages 193-207.

Müller, J. P., Singh, M. P., and Rao, A.S., editors (1999). *Intelligent Agents V. Proceedings of the Fifth International Workshop on Agent Theories, Architectures, and Languages, ATAL'98,* volume 1555 of Lecture Notes in Artificial Intelligence, Springer Verlag, Berlin.

Müller, J. P., Wooldridge, M., and Jennings, N.R., editors (1997). *Intelligent Agents III. Proceedings of the Third International Workshop on Agent Theories, Architectures, and Languages, ATAL'96*, volume 1193 of Lecture Notes in Artificial Intelligence, Springer-Verlag, Berlin.

Neches, R. Swartout, W. R. & Moore, J. (1985). Explainable (and maintainable) expert systems. In *Proceedings of the International Joint Conference on Artificial Intelligence, IJCAI'85*, volume 1, pages 382-389.

Newell, A. (1984). The knowledge level. *Artificial Intelligence*, 18(1):87-127.

Nii, H. P. (1986a). Blackboard systems: The blackboard model of problem solving and the evolution of blackboard architectures. Part I. *AI Magazine*, 7(2):38-53.

Nii, H. P. (1986b). Blackboard systems: Blackboard application systems, blackboard systems from a knowledge engineering perspective. Part II. *AI Magazine*, 7(3):82-106.

O'Hare, G. M. P. and Jennings, N. R., editors (1996). *Foundations of Distributed Artificial Intelligence*. Sixth Generation Computer Technology Series. John Wiley & Sons, New York.

Papadopoulos, G. A. and Arbab, F. (1998). *Coordination Models and Languages*. Technical Report SEN-R9834, Centrum voor Wiskunde en Informatica, Amsterdam, The Netherlands.

Pednault, E. P. (1987). Formulating multiagent, dynamic-world problems in the classical planning framework. In Georgeff, M. and Lansky, A., editors, *Reasoning about Actions and Plans*.

Perry, D. E. and Wolf, A. L. (1992). Foundations for the study of software architecture. *Software Engineering Notes*, 17(4):40-52.

Petri, C. A. (1962). *Kommunikation mit Automaten*. Ph.D. Thesis, Institut für Instrumentelle Mathematik, Bonn.

Plotkin, G. D. (1981). *Structured Approach to Operational Semantics*. Technical Report DAIMI FN-19, Computer Science Department, Aarhus University.

Plotkin, G. D. (1982). An operational semantics for CSP. In Bjorner, D., editor, *Formal Description of Programming Concepts II*, North-Holland, pages 199-225.

Pnueli, A. (1977). The temporal logic of programs. In *18th Annual Symposium on Foundations of Computer Science*, Providence, pages 46-57.

Prade, H., editor (1998). *Proceedings of the 13th European Conference of Artificial Intelligence, ECAI 98*. John Wiley & Sons.

Pratt, V. R. (1986). Modeling concurrency with partial orders. *Int. J. of Parallel Programming*, 15(1):33-71.

Pratt, V. R. (1991). Modeling concurrency with geometry. In *Proceedings of the 18th Annual ACM Symposium on Principles of Programming Languages*, pages 311-322.

Pratt, V. R. (1992). Event spaces and their linear logic. In *AMAST'91: Algebraic Methodology and Software Technology*, Workshops in Computing, pages 1-23. Springer-Verlag.

Pratt, V. R. (1995). Chu spaces and their interpretation as concurrent objects. In Leeuwen, J. van, editor, *Computer Science Today: Recent Trends and Developments*, volume 1000 of Lecture Notes in Computer Science, Springer-Verlag, pages 392-405.

Priestley, H. A. (1970). Representation of distributive lattices by means of ordered Stone spaces. *Bulletin of the London Mathematical Society*, 2:186-190.

Rao, A. S. and Georgeff, M. (1991). Modeling rational agents within a BDI-architecture. In Fikes, R. and Sandewall, E., editors, *Proceedings of the Second Conference on Knowledge Representation and Reasoning, KR'91*, pages 473-484. Morgan Kaufmann.

Raskin, J.-F. and Schobbens, P.-Y. (1997). Real-time logics: Fictitious clock as an abstraction of dense time. In Brinksma, E., editor, *Proceedings of TACAS'97: Tools and Algorithms for the Construction and Analysis of Systems*, volume 1217 of Lecture Notes in Computer Science, pages 165-182. Springer-Verlag, Berlin.

Rebecca-Thomas, S. (1995). The PLACA agent programming language. In (Wooldridge & Jennings, 1995a), pages 355-370.

Reynolds, M. (1995). Towards first-order concurrent metatem. In (Fisher & Owens, 1995), pages 118-143.

Ricart, G. and Agrawala, A. K. (1981). An optimal algorithm for mutual exclusion in computer networks. *Communications of the ACM*, 24(1):9-17, January.

Roever, W.-P. de (1998). The need for compositional proof systems: A survey. In (Roever, Langmaack & Pnueli, 1998), pages 1-22.

Roever, W.-P. de, Langmaack, H., and Pnueli, A. (1998). *Compositionality: The Significant Difference. International Symposium, COMPOS'97*, volume 1536 of Lecture Notes in Computer Science, Springer-Verlag, Berlin.

Schneider, F. B. and Lamport, L. (1984). Paradigms for distributed programming. In Alford, M. W., Ansant, J. P., Hommel, G., Lamport, L., Liskov, B., Mullery, G. P., and Schneider, F. B., editors, *Distributed Systems. Methods and Tools for Specification. An Advanced Course*, volume 190 of *Lecture Notes in Computer Science*, pages 431-480. Springer-Verlag.

Schreiber, A. Th., Akkermans, J. M., Anjewierden, A. A., Hoog, R. de, Shadbolt, N. G., Velde, W. van de and Wielinga, B. J. (1999). *Knowledge Engineering and Management. The CommonKADS Methodology*. MIT Press.

Schwarz, R. and Mattern, F. (1994). Detecting causal relationships in distributed computations: in search of the holy grail. *Distributed Computing*, 7:149-174.

Seel, N. (1989). *Agent Theories and Architectures*. Ph.D. Thesis, Surrey University, Guildford, UK.

Sernadas, A., Sernadas, C. and Costa, J.F. (1995). Object specification logic. *Journal of Logic and Computation*, 5(5):603-630, October. Slightly different version appeared in Research Report INESC/DMIST, Lisbon, 1992, same authors and title.

Singh, M. P. (1998). The intentions of teams: Team structure, endodeixis, and exodeixis. In (Prade, 1998), pages 303-307.

Singh, M. P., Rao, A. S., and Wooldridge, M. J., editors (1998). *Intelligent Agents IV. Proceedings of the Fourth International Workshop on Agent Theories, Architectures, and Languages, ATAL'97*, volume 1365 of Lecture Notes in Artificial Intelligence, Springer-Verlag, Berlin.

Shoham, Y. (1993). Agent-oriented programming. *Artificial Intelligence*, 60(1):51-92.

Smith, R.G. (1980). The contract net protocol: high-level communication and control in a distributed problem solver. *IEEE Transactions on Computers*, C29 (12).

Soneoka, T. and Ibaraki, T. (1994). Logically instantaneous message passing in asynchronous distributed systems. *IEEE Transactions on Computers*, 43(5):513-527.

Stone, M. H. (1936). The theory of representations for Boolean algebras. *Transactions of the American Mathematical Society*, 40:37-111.

Tanenbaum, A. S. (1992). *Modern Operating Systems*. Prentice Hall.

Tanenbaum, A. S. (1996). *Computer Networks, 3rd Edition*. Prentice Hall.

Turner, R. (1990). *Truth and Modality for Knowledge Representation.* Pitman.

Velde, W. van der and Perram, J. W., editors (1996). *Agents Breaking Away. Proc. 7th Eur. Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW'96*, volume 1038 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin.

Wieringa, R. J. (1995). *Requirements engineering: frameworks for understanding*. John Wiley & Sons.

Winskel, G. (1989). An introduction to event structures. In Bakker, J. W. de, Roever, W.-P. de and Rozenberg, G., editors, *Linear Time, Branching Time and Partial Order in Logics and Models of Concurrency*, volume 354 of Lecture Notes in Computer Science, pages 364-397. Springer-Verlag, Berlin.

Wirth, N. (1982). *Modula-2*. Springer-Verlag, New York.

Wooldridge, M. J. (1996). *Temporal belief logics for modeling distributed artificial intelligence systems*. In (O'Hare & Jennings, 1996), chapter 10.

Wooldridge, M. J. (1999). Intelligent agents. In Weiss, G., editor, *Multiagent Systems. A Modern Approach to Distributed Artificial Intelligence*. The MIT Press. Chapter 1.

Wooldridge, M. J. and Jennings, N. R., editors (1995a). *Intelligent Agents. Proceedings of the First International Workshop on Agent Theories, Architectures, and Languages, ATAL'94*, volume 890 of *Lecture Notes in Artificial Intelligence*. Springer-Verlag, Berlin.

Wooldridge, M. J. and Jennings, N. R. (1995b). Intelligent agents: Theory and practice. *The Knowledge Engineering Review*, 10(2):115-152.

Wooldridge, M. J., Müller, J. P., and Tambe, M., editors (1996). *Intelligent Agents II. Proceedings of the Second International Workshop on Agent Theories, Architectures, and Languages, ATAL'95*, volume 1037 of Lecture Notes in Artificial Intelligence. Springer-Verlag, Berlin.

Wooldridge, M. J. and Jennings, N. R. (1998). Pitfalls of agent-oriented development. In Sycara, K. P. and Wooldridge, M. J., editors, *Proceedings of the Second International Conference on Autonomous Agents*. ACM Press.

Summary

A Compositional Semantic Structure for Multi-Agent Systems Dynamics

Since a number of years, *multi-agent systems* have attracted considerable attention in Computer Science and Artificial Intelligence. A multi-agent system is a software (or hardware) system that consists of a set of co-operating, autonomous parts, called agents. For analysis and design, specification formalisms are needed that are especially suitable for such systems. To enable automated support for the analysis and design process, the semantics of such formalisms has to be described precisely.

The semantics of a specification formalism consists of a mapping of expressions of the formalism onto a set of concepts and relations between these concepts that is sufficiently rich to express the intended semantics. In this thesis, such a set is called a *semantic structure*. To enable the precision mentioned above, the semantic structure itself is described with the help of formal, mathematical notions.

The goal of the research described in this thesis is the development of a semantic structure for the description of the dynamics of multi-agent systems. Due to the specific characteristics of multi-agent systems, four requirements are stated with which the semantic structure should comply: (i) the semantic structure should support the description of both the internal dynamics of an agent and of the interaction between agents, (ii) the description of dynamics should be compositional (i.e., the behaviour of a system is described in terms that refer to the behaviour of the parts that make up the system), (iii) dynamics should be described in terms of states and state transitions of the agents, and (iv) the dynamics of a specific part of a system should be described in terms that only refer to this part (and possibly to its constituents). In other words, it is not possible to take a notion of state at a global or systems level as the basis for the semantic structure. These requirements are further described in Chapter 1.

The starting point for the development of the semantic structure is the assumption that a multi-agent system can be modelled as a *compositional system*. A compositional system is understood to be a system that consists of *components* and connections between these components: *information links*. A component is a locus of

information and computation that processes this information. A component offers services to other components, and is able to use services offered by other components. Information exchange necessary for the use of services takes place via the information links between components. A component may be composed of other components, which gives rise to a recursive system structure.

The starting point for the development of the semantic structure is further elaborated in Chapter 2 and Chapter 3. In Chapter 2, the notion of a component is described in more detail, and a number of choices with respect to the mechanism of information exchange between components are discussed. The development of the semantic structure requires that commitments are made to a number of these choices. Chapter 3 discusses how a multi-agent system can be modelled as a compositional system, again including a number of choices that have to be made. However, for the further development of the semantic structure, commitments to specific choices presented in Chapter 3 are not required.

Chapters 4 to 7 contain the formal description of the semantic structure. Chapter 4 describes, in an informal way, the central construct in the semantic structure, which is formally elaborated in Chapter 5. In addition, an example is introduced that is used in the next chapters to illustrate the semantic structure. A description of the dynamics of a compositional system in terms of the semantic structure consists of *traces*, one for each component. A trace is a sequence of *local states* of the component. A local state of a component only includes (descriptions of) the state of that component, and not of any other component. Only local traces that are *compatible* may occur together in the set of traces that describes the behaviour of a compositional system. Local traces of two components that are connected by an information link are compatible if the traces respect information exchange as described for this link: in essence, information that is sent in one trace should be received in the other. (Local traces of two components that are not connected by an information link are compatible by definition.)

Chapter 5 first presents formal definitions for the notions introduced in Chapter 2 and Chapter 3. These notions determine the non-dynamic structure of compositional systems. Three views on the behaviour of a compositional system are formally defined. These three views differ in the extent to which the behaviour of subcomponents is visible: not at all, only for one level, or fully. Chapter 6 elaborates upon the notion of compatibility. The choices described in Chapter 2 with respect to the mechanism of information exchange are presented in this chapter as properties of compatibility relations between sets of local traces. Chapter 7 shows that, starting from the concepts presented in Chapter 5 and Chapter 6 (that do not rely on a notion of global state), it is possible to define a notion of global state, without referring to a notion of global (synchronised) time. The notion of global state is compared to similar notions that can be found in the literature.

Chapter 8 is devoted to modelling control within the semantic structure and within multi-agent systems. Due to the supposed autonomy of agents, modelling

control is subject to constraints. Control is understood to be information transmission from one component, to a component that is to be controlled, with the intention of influencing the behaviour of the component that is to be controlled. In this way, it is possible to describe control in terms of the semantic structure without the need to extend the semantic structure, and without compromising the autonomy of controlled components.

Chapters 9 to 11 present applications of the semantic structure. In these chapters, the DESIRE modelling framework plays a central role. The DESIRE modelling framework provides a partially graphical and partially textual language for the specification of multi-agent systems. The language is sufficiently rich to enable automatic generation of prototypes of the systems that are specified. Chapter 9 presents a detailed description of the semantics of DESIRE specifications in terms of the semantic structure, with emphasis on the dynamics of multi-agent systems specified using DESIRE. In Chapter 10, a model for co-operation to establish mutually exclusive access to a resource is specified using DESIRE. In this model, a small number of agents with a complex structure co-operate. In contrast, in Chapter 11, a DESIRE-specification of altruistic behaviour between a relatively large number of simple agents is presented. A way to analyse both specifications is sketched, based on the semantics in terms of the semantic structure.

The final chapter of this thesis, Chapter 12, contains an evaluation of the semantic structure with respect to the requirements put forward in Chapter 1. Moreover, the approach presented in this thesis is compared to a number of other approaches that can be found in the literature.

Samenvatting

Een compositionele semantische structuur voor de dynamiek van multi-agentsystemen

Sinds enige jaren is er in de Informatica en in de Kunstmatige Intelligentie veel belangstelling voor *multi-agentsystemen*. Hieronder worden software– (of hardware–) systemen verstaan die zijn opgebouwd uit samenwerkende autonome delen, de agenten. Er bestaat, voor analyse en ontwerp, behoefte aan specificatieformalismen die speciaal geschikt zijn voor degelijke systemen. De betekenis (semantiek) van dergelijke formalismen moet precies worden vastgelegd om geautomatiseerde ondersteuning van het analyse- en ontwerpproces mogelijk te maken.

De semantiek van een specificatieformalisme bestaat uit een afbeelding van uitdrukkingen in dat formalisme op een verzameling van concepten en relaties tussen deze concepten die voldoende rijk is om de beoogde semantiek uit te drukken. In dit proefschrift wordt een dergelijke verzameling een *semantische structuur* genoemd. Ten behoeve van bovengenoemde precisie wordt de semantische structuur zelf beschreven met behulp van formele, wiskundige begrippen.

Doel van het in dit proefschrift gepresenteerde onderzoek is het ontwikkelen van een semantische structuur voor de beschrijving van de dynamiek van multi-agentsystemen. Vanwege de specifieke eigenschappen van multi-agentsystemen wordt aan de semantische structuur een viertal eisen gesteld: (i) de semantische structuur moet ondersteuning bieden voor de beschrijving van zowel de interne dynamiek van een agent als van de interactie tussen agenten, (ii) de beschrijving van dynamiek moet compositioneel zijn (dat wil zeggen, gedrag van een systeem wordt beschreven in termen van het gedrag van de onderdelen van het systeem), (iii) dynamiek moet worden uitgedrukt in termen van de toestand en toestandsovergangen van de agenten, en (iv) dynamiek van een bepaald systeemdeel moet beschreven kunnen worden in termen die alleen aan dat systeemdeel (en eventueel onderdelen daarvan) refereren (met andere woorden, er kan niet worden uit-

gegaan van een toestandsbegrip op globaal ofwel systeemniveau). Deze eisen worden nader toegelicht in hoofdstuk 1.

Het startpunt voor de ontwikkeling van de semantische structuur is de aanname dat een multi-agentsysteem gemodelleerd kan worden als een *compositioneel systeem*. Met een compositioneel systeem wordt een systeem bedoeld dat bestaat uit *componenten* en verbindingen tussen deze componenten: de *information links*. Een component vormt een eenheid van gegevens en bewerkingen op die gegevens. Een component biedt diensten aan ten behoeve van andere componenten, en kan zelf van diensten van andere componenten gebruik maken. De hiervoor benodigde uitwisseling van informatie vindt plaats via de information links tussen de componenten. Een component kan zelf bestaan uit andere componenten. Hierdoor ontstaat een recursieve systeemstructuur.

In de hoofdstukken 2 en 3 wordt het startpunt voor de ontwikkeling van de semantische structuur verder uitgewerkt. In hoofdstuk 2 wordt het componentbegrip nader toegelicht en wordt een groot aantal keuzes besproken met betrekking tot het mechanisme van informatie-uitwisseling tussen componenten. Voor de ontwikkeling van de semantische structuur moet een aantal van deze keuzes worden vastgelegd. In hoofdstuk 3 wordt besproken hoe een multi-agentsysteem gemodelleerd kan worden als een compositioneel systeem. Ook hierbij is sprake van een zekere keuzevrijheid, die in hoofdstuk 3 aan de orde komt. Deze keuzes hoeven echter niet vastgelegd te worden voor de verdere ontwikkeling van de semantische structuur.

De hoofdstukken 4 tot en met 7 bevatten de formele beschrijving van de semantische structuur. Hoofdstuk 4 beschrijft op informele wijze de centrale constructie in de semantische structuur, die in hoofdstuk 5 formeel wordt uitgewerkt. Daarnaast wordt een voorbeeld geïntroduceerd dat in de volgende hoofdstukken gebruikt wordt om de semantische structuur te illustreren. Een beschrijving van de dynamiek van een compositioneel systeem in termen van de semantische structuur bestaat uit *traces*, één voor elke component, waarbij een trace een opeenvolging is van *lokale toestanden* van die component. In een lokale toestand komt alleen de (beschrijving van) de toestand van de component in kwestie voor in zijn trace, niet die van andere componenten. Alleen lokale traces die *compatibel* zijn komen tezamen voor in de verzameling van traces die het gedrag van een compositioneel systeem beschrijft. Lokale traces van twee componenten die aan elkaar zijn verbonden met een information link, zijn compatibel als de traces informatie-uitwisseling volgens de beschrijving van deze link respecteren. Hiermee wordt in essentie bedoeld dat informatie die verzonden wordt in de ene trace moet ontvangen worden in de andere. (Lokale traces van twee componenten die niet verbonden zijn met een information link, zijn per definitie compatibel.)

In hoofdstuk 5 worden allereerst formele definities gegeven voor de begrippen die in de hoofdstukken 2 en 3 geïntroduceerd zijn. Hiermee is de niet-dynamische structuur van compositionele systemen vastgelegd. Vervolgens worden drie perspectieven op het gedrag van een compositioneel systeem formeel gedefinieerd.

Deze drie perspectieven verschillen in de mate waarin het gedrag van sub-componenten zichtbaar is in de beschrijving van het gedrag: in het geheel niet, één niveau diep, of volledig. Hoofdstuk 6 gaat nader in op het begrip compatibiliteit. De keuzes die in hoofdstuk 2 beschreven zijn met betrekking tot het mechanisme van informatie-uitwisseling, worden in dit hoofdstuk gerepresenteerd als eisen op de compatibiliteitsrelaties tussen verzamelingen van lokale traces. Hoofdstuk 7 laat zien dat, uitgaande van de concepten gepresenteerd in de hoofdstukken 5 en 6 (waarin geen globaal toestandsbegrip voorkomt) een globaal toestandsbegrip gedefinieerd kan worden, zonder daarbij uit te gaan van een globaal (gesynchroniseerd) tijdsbegrip. Het geïntroduceerde globale toestandsbegrip wordt vergeleken met soortgelijke noties uit de literatuur.

Hoofdstuk 8 is gewijd aan het modelleren van besturing in de semantische structuur en in multi-agentsystemen. Vanwege de veronderstelde autonomie van agenten is de modellering van besturing aan beperkingen onderhevig. Besturing wordt opgevat als informatietransmissie van een component naar een te besturen component met het beoogde doel om invloed uit te oefenen op het gedrag van de te besturen component. Op deze wijze opgevat kan besturing worden beschreven in termen van de semantische structuur zonder dat de semantische structuur uitgebreid hoeft te worden, en wordt de autonomie van bestuurde componenten niet aangetast.

De hoofdstukken 9, 10 en 11 gaan in op toepassingen van de semantische structuur. In deze hoofdstukken staat het DESIRE modelleerraamwerk centraal. Dit modelleerraamwerk biedt een gedeeltelijk grafische en gedeeltelijk tekstuele taal voor de specificatie van multi-agentsystemen. De taal is voldoende rijk om automatische vervaardiging van uitvoerbare prototypes van de gespecificeerde systemen mogelijk te maken. In hoofdstuk 9 wordt een gedetailleerde beschrijving van de betekenis van DESIRE-specificaties in termen van de semantische structuur gegeven, met nadruk op de dynamiek van in DESIRE gespecificeerde multi-agentsystemen. In hoofdstuk 10 wordt een samenwerkingsmodel voor het verkrijgen van exclusieve beschikking over een zeker middel gespecificeerd met behulp van DESIRE. Bij dit model gaat het om samenwerking tussen een gering aantal agenten met een complexe structuur. Het omgekeerde is het geval in hoofdstuk 11, waar een DESIRE-specificatie wordt gepresenteerd van altruïstisch gedrag van eenvoudige agenten. Van beide specificaties wordt geschetst hoe zij geanalyseerd kunnen worden door uit te gaan van hun betekenis in termen van de semantische structuur.

Het laatste hoofdstuk van het proefschrift, hoofdstuk 12, bevat een evaluatie van de semantische structuur met betrekking tot de eisen die in hoofdstuk 1 gesteld zijn. Daarnaast wordt de in dit proefschrift gepresenteerde benadering vergeleken met enkele andere benaderingen uit de literatuur.

SIKS Dissertation Series

98-1 Johan van den Akker (CWI), *DEGAS—An Active, Temporal Database of Autonomous Objects.*

98-2 Floris Wiesman (UM), *Information Retrieval by Graphically Browsing Meta-Information.*

98-3 Ans Steuten (TUD), *A Contribution to the Linguistic Analysis of Business Conversations within the Language/Action Perspective.*

98-4 Dennis Breuker (UM), *Memory versus Search in Games.*

98-5 E. W. Oskamp (RUL), *Computerondersteuning bij Straftoemeting.*

99-1 Mark Sloof (VU), *Physiology of Quality Change Modelling; Automated Modelling of Quality Change of Agricultural Products.*

99-2 Rob Potharst (EUR), *Classification using decision trees and neural nets.*

99-3 Don Beal (Queen Mary and Westfield College), *The Nature of Minimax Search.*

99-4 Jacques Penders (KPN Research), *The practical Art of Moving Physical Objects.*

99-5 Aldo de Moor (KUB), *Empowering Communities: A Method for the Legitimate User-Driven Specification of Network Information Systems.*

99-6 Niek J.E. Wijngaards (VU), *Re-design of compositional systems.*

99-7 David Spelt (UT), *Verification support for object database design.*

99-8 Jacques H.J. Lenting (UM), *Informed Gambling: Conception and Analysis of a Multi-Agent Mechanism for Discrete Reallocation.*

2000-1 Frank Niessink (VU), *Perspectives on Improving Software Maintenance.*

2000-2 Koen Holtman (TUE), *Prototyping of CMS Storage Management.*

2000-3 Carolien M.T. Metselaar (UvA), *Sociaal-organisatorische gevolgen van kennistechnologie; een procesbenadering en actorperspectief.*

2000-4 Geert de Haan (VU), *ETAG, A Formal Model of Competence Knowledge for User Interface Design.*

2000-5 Ruud van der Pol (UM), *Knowledge-based Query Formulation in Information Retrieval.*

2000-6 Rogier van Eijk (UU), *Programming Languages for Agent Communication.*

2000-7 Niels Peek (UU), *Decision-theoretic Planning of Clinical Patient Management.*

2000-8 Veerle Coupé (EUR), *Sensitivity Analyis of Decision-Theoretic Networks.*

2000-9 Florian Waas (CWI), *Principles of Probabilistic Query Optimization.*

2000-10 Niels Nes (CWI), *Image Database Management System Design Considerations, Algorithms and Architecture.*

2000-11 Jonas Karlsson (CWI), *Scalable Distributed Data Structures for Database Management.*

2001-1 Silja Renooij (UU), *Qualitative Approaches to Quantifying Probabilistic Networks.*

2001-2 Koen Hindriks (UU), *Agent Programming Languages: Programming with Mental Models.*

2001-3 Maarten van Someren (UvA), *Learning as problem solving.*

2001-4 Evgueni Smirnov (UM), *Conjunctive and Disjunctive Version Spaces with Instance-Based Boundary Sets.*

2001-5 Jacco van Ossenbruggen (VU), *Processing Structured Hypermedia: A Matter of Style.*

2001-6 Martijn van Welie (VU), *Task-based User Interface Design.*

2001-7 Bastiaan Schönhage (VU), *Diva: Architectural Perspectives on Information Visualization.*

2001-8 Pascal van Eck (VU). *A Compositional Semantic Structure for Multi-Agent Systems Dynamics.*

CWI – Centrum voor Wiskunde en Informatica / National Research Institute for Mathematics and Computer Science, Amsterdam.
EUR – Erasmus Universiteit Rotterdam / Erasmus University Rotterdam.
KUB – Katholieke Universiteit Brabant / Tilburg University
RUL – Universiteit Leiden / Leiden University.
TUD – Technische Universiteit Delft / Delft University of Technology.
TUE – Technische Universiteit Eindhoven.
UM – Universiteit Maastricht.
UT – Universiteit Twente / University of Twente.
UU – Universiteit Utrecht / Utrecht University.
UvA – Universiteit van Amsterdam.
VU – Vrije Universiteit Amsterdam.