Università degli Studi di Pisa
Dipartimento di Informatica

Dottorato di Ricerca in Informatica

Ph.D. Thesis: TD-1/00

Modelling Concurrent Computations: from Contextual Petri Nets to Graph Grammars

Paolo Baldan

March 2000

Addr: Corso Italia 40, 56125 Pisa, Italy
Tel: +39-050-887268 — Fax: +39-050-887226 — E-mail: baldan@di.unipi.it
http://www.di.unipi.it/~ baldan

Thesis Supervisors:
Dott. Andrea Corradini and Prof. Ugo Montanari

Abstract

Graph grammars (or *graph transformation systems*), originally introduced as a generalization of string grammars, can be seen as a powerful formalism for the specification of concurrent and distributed systems, which properly extends Petri nets. The idea is that the state of a distributed system can be naturally represented (at a suitable level of abstraction) as a graph and local state transformations can be expressed as production applications.

With the aim of consolidating the foundations of the concurrency theory for graph transformation systems, the thesis extends to this more general setting some fundamental approaches to the semantics coming from Petri net theory. More specifically, focusing on the so-called *double pushout* (DPO) *algebraic approach* to graph rewriting, the thesis provides graph transformation systems with truly concurrent semantics based on (concatenable) processes and on a Winskel's style unfolding construction, as well as with more abstract semantics based on event structures and domains.

The first part of the thesis studies two generalizations of Petri nets, already known in the literature, which reveal a close relationship with graph transformation systems, namely *contextual nets* (also called nets with read, activator or test arcs) and *inhibitor nets* (or nets with inhibitor arcs). Extending Winskel's seminal work on safe nets, the truly concurrent semantics of contextual nets is given via a chain of coreflections leading from the category of contextual nets to the category of finitary coherent prime algebraic domains. A basic role is played by *asymmetric event structures*, which generalize prime event structures by allowing a non-symmetric conflict relation. The work is then generalized to inhibitor nets, where, due to the non-monotonicity of the enabling, the causal structure of computations is far more complex, and a new, very general, notion of event structure, called *inhibitor event structure*, is needed to faithfully describe them.

The second part of the thesis, relying on the conceptual basis drawn in the first part, focuses on graph grammars. Inhibitor event structures turn out to be expressive enough to model graph grammar computations, and the theory developed for contextual and inhibitor nets, comprising the unfolding and the (concatenable) process semantics, can be lifted to graph grammars. The developed semantics is shown to to be consistent also with the classical theory of concurrency for DPO graph grammars relying on *shift-equivalence*.

To Alessandra.

Acknowledgments

Not surprisingly, the people I am most indebted to are my advisors, Ugo Montanari and Andrea Corradini, who followed and supported me with high competence during my PhD studies. They have been a constant source of ideas and suggestions which have played a fundamental role in the development of the material in this thesis.

I surely cannot forget to thank Furio Honsell and Fabio Alessi, the advisors of my master thesis in Udine. There, I moved my first steps in the world of research, learning the importance of aesthetic criteria, like elegance and beauty, in the evaluation of scientific productions, and I understood that it is possible (and necessary) to enjoy working.

Many thanks are due to my external reviewers, Philippe Darondeau and Hartmut Ehrig, for their careful reading of a preliminary version of this thesis, and for their useful comments and suggestions. I would like to express my gratitude also to Giorgio Ghelli and Andrea Maggiolo Schettini, who followed the annual advancement of my thesis work during the last three years.

I took full advantage from my participation to the GETGRATS project, which offered me the opportunity of meeting and discussing with several people. In particular, I would like to remember Martin Große-Rhode, Reiko Heckel, Manuel Koch, Marta Simeoni and Francesco Parisi Presicce. I would like to acknowledge also Nadia Busi and Michele Pinna who provided me some useful hints to improve the material in Chapter 3.

I want to thank also my colleagues and friends at the Department of Computer Science in Pisa, who contributed to create a nice and stimulating (not only from the scientific viewpoint) working environment. When I moved to Pisa to start my PhD course, from the very beginning I found many kind people who received me in the best way. In particular I would like to mention Simone Martini, Andrea Masini and Laura Semini who helped me with their friendship and with many useful advices, and Giorgio Ghelli and Maria Simi who kindly hosted me in their office. I would like to thank Giorgio also for the pleasure I had working with him.

Then, many many people with whom I had courses, meals and discussions have influenced positively the quality of my staying at the department. Roberto Bruni, Gianluigi Ferrari, Fabio Gadducci and Mercè Llabrés Segura (a guest who has immediately become part of the "family") besides being very good friends (or perhaps just for this), read a preliminary version of the thesis helping me to greatly improve

the presentation. A special thank goes to Roberto for the never-ending discussions he has always accepted to engage on disparate subjects. The long working days (often continuing through nights) have been less hard for the presence of my roommates, from the former ones, Vladimiro Sassone, Marco Pistore and Paola Quaglia to the recent ones, Stefano Bistarelli, Diego Sona and Emilio Tuosto. I also want to thank Chiara Bodei and Roberta Gori, with whom I shared the pain of the thesis deadline, the "polemic" Andrea Bracciali, the "comrade" Simone Contiero, Paolo Volpe and Francesca Scozzari.

Vorrei ringraziare i miei genitori per essermi sempre stati vicini, e ricordare mia nonna Cea che è stata per me una seconda madre. Un grazie anche a Leo e Nico, e un benvenuto ad Edoardo.

Infine, un ringraziamento davvero speciale va ad Alessandra, mia moglie e compagna di vita. Anche nella preparazione di questa tesi il suo aiuto è stato prezioso e insostituibile ... vorrei dire altro, ma cadrei presto in luoghi comuni, così come spesso accade a chi, non essendo un poeta, si avventuri a parlare di cose per cui le parole non bastano.

Contents

List of Figures

Chapter 1

Introduction

Over the last thirty years there has been a steadily growing interest in concurrent and distributed systems. An unbroken thread leads from the more classical ideas related to the sharing of many computing resources among various users (multiuser systems, databases, etc.) and the use of many distinct computing resources to obtain a greater computational power (multiprocessor computing), to the current widespread diffusion of Internet and network applications. Day by day real systems become more complex and sophisticated, but at the same time more difficult to test and verify, and thus possibly more unreliable. Because of this, new formal models adequate to ease the specification, development and verification of concurrent systems are called for.

Generally speaking, a formal model must give a representation of a system which is abstract enough to disregard unnecessary details, and, at the same time, is sufficiently rich to allow one to represent properties and aspects of the system which may be relevant for the design and verification activities. Besides representing the state and the architectural aspects of a system, a model typically comes equipped with an operational semantics which formally explains how the system behaves. On top of the concrete operational description, depending on which observations one wants to take into account, more abstract semantics can be introduced. At this level one can define techniques for checking the equivalence of systems with respect to the selected observations, for verifying if a systems satisfies a given property, for synthesizing in an efficient way a system satisfying a given property, etc.

For sequential systems it is often sufficient to consider an input/output semantics and thus the appropriate semantic domain is usually a suitable class of functions from the input to the output domains. When concurrent or distributed features are involved, instead, typically more information about the actual computation of the system has to be recorded in the semantic domain. For instance, one may want to know which steps of computation are independent (concurrent), which steps are causally related and which steps represent the (nondeterministic) choice points. This information is necessary, for example, if one wants to have a compositional semantics, allowing to reduce the complexity of the analysis of concurrent systems built from

smaller parts, or if one wants to allocate a computation on a distributed architecture. Roughly speaking, *nondeterminism* can be represented either by collecting all the possible different computations in a set, or by merging the different computations in a unique *branching* structure where the choice points are explicitly represented. On the other hand, *concurrent* aspects can be represented by using a *truly concurrent* approach, where the causal dependencies among events are described directly in the semantics using a partially ordered structure. There is some agreement in considering this choice more appropriate for the analysis of concurrent and distributed systems than the *interleaving* approach, where concurrency is confused with nondeterminism, in the sense that the concurrent execution of events is represented as the nondeterministic choice among the possible interleavings of such events.

Petri nets are one of the the most widely used models of concurrency, which has attracted, since its introduction, the interest of both theoreticians and practitioners. Along the years Petri nets have been equipped with satisfactory semantics, making justice of their intrinsically concurrent nature and which have served as basis for the development of a variety of modelling and verification techniques. However, the simplicity of Petri nets, which is one of the reasons of their success, represents also a limit in their expressiveness. If one is interested in giving a more structured description of the state, or if the kind of dependencies between steps of computation cannot be reduced simply to causality and conflict, Petri nets are likely to be inadequate.

This thesis is part of a project aimed at proposing graph transformation systems as an alternative model of concurrency, extending Petri nets. The basic intuition underlying the use of graph transformation systems for formal specifications is to represent the states of a system as graphs (possibly attributed with data-values) and state transformations by means of rule-based graph transformations. Since a rule has only a local effect on the state, it is natural to allow for the parallel application of rules acting on different parts of the state, a fact that makes graph transformation systems suited for the representation of concurrency.

Needless to say, the idea of representing system states by means of graphs is pervasive in computer science. Whenever one is interested in giving an explicit representation of the interconnections, or more generally of the relationships among the various components of a system, a natural solutions is to use (possibly hierarchical and attributed) graphs. The possibility of giving a suggestive pictorial representation of graphical states makes them adequate for the description of the meaning of a system specification, even to a non-technical audience. A popular example of graph-based specification language is given by the Unified Modeling Language (UML). In UML the conceptual models of system states, the collection of admissible states, and, on an higher level, the distribution of systems are represented by means of diagrams, which are of course graphs, sometimes attributed with textual information. Recall also the more classical Entity/Relationship (ER) approach, where graphs are used to specify the conceptual organization of the data, or Statecharts, a specification language suited for reactive systems, where states are organized in a hierarchical tree-like structure. Furthermore, graphs provides a privileged representation of sys-

tems consisting of a set of processes communicating through ports. When one is interested in modelling the dynamic aspects of systems whose states have a graphical nature, graph transformation systems are clearly one of the most natural choices.

With the aim of consolidating the foundations of the concurrency theory of graph transformation systems, this thesis extends to this more general setting some fundamental approaches to the semantics of Petri nets. More specifically, inspired by the close relationships existing between nets and graph transformation systems, we provide graph transformation systems with truly concurrent semantics based on deterministic processes and on a Winskel-like unfolding construction, as well as with more abstract semantics based on event structures and domains.

As an intermediate step, we study two generalizations of Petri nets proposed in the literature, which reveal a close relationship with graph transformation systems, namely contextual nets (also called nets with read, activator or test arcs) and nets with inhibitor arcs. Due to their relatively wide diffusion, we believe that the work on these extended kinds of nets may be understood as an additional outcome of the thesis, independently from its usefulness in carrying out our program on graph transformation systems.

The rest of this introductory chapter is aimed at presenting the general framework in which this thesis has been developed, the main motivations and concepts from Petri net theory, and an overview of the results. First, in Sections 1.1 and 1.2, we give a description of the basic models, namely ordinary Petri nets, contextual and inhibitor nets and finally graph transformation systems, organized in an ideal chain where each model generalizes its predecessor. Then Section 1.3 outlines the approach to the truly concurrent semantics of ordinary Petri nets which we propose as a paradigm. Section 1.4 gives an overview of the thesis, by explaining how the semantical framework of ordinary nets has been lifted along the chain of models, first to contextual and inhibitor nets and then to graph grammars. We give a flavour of the main problems which we encountered and we outline the main achievements of the thesis. Finally, Section 1.5 describes how the thesis is organized and explains the origin of the chapters.

1.1 Petri nets

Petri nets, introduced by Carl Adam Petri in the early Sixties [Pet62] (see also [Rei85]), are one of the most widely accepted models of concurrent and distributed systems, and one of the rare meeting point of theoreticians and practitioners. The success of Petri nets in the last thirty years can be measured by looking not only at the uncountably many applications of nets, but also at the development of the theoretical aspects, which range from a complete analysis of the various phenomena arising in simple models to the definition of more expressive (and complex) classes of nets.

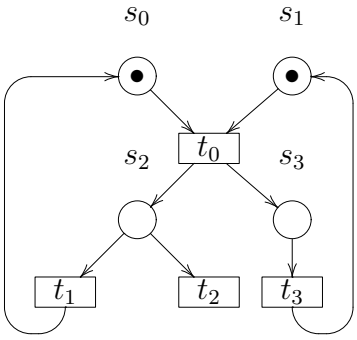


Figure 1.1: A simple Petri net.

One of the reasons of the popularity of Petri nets is probably the simplicity of the basic model, whose behaviour, at a fundamental level, can be understood simply in terms of the so-called *token game*. The state of a net is a set of *tokens* distributed among a set of *places*. A *transition* is enabled in a state if enough tokens are present in the places in its pre-set. The firing of the transition *removes* some tokens and *produces* new tokens in its postconditions.

Petri nets admit a very pleasant and simple graphical representation where places are drawn as circles, transitions as rectangular boxes and arrows are used to represent the *flow relation*, specifying which resources are consumed and produced by each transition. To visualize the state of the net, usually called *marking*, the tokens are depicted as bullets inside the corresponding places. For instance, in the net of Figure 1.1, the transition t_0 consumes two tokens, one from s_0 and one from s_1, and thus it is enabled in the represented marking. Its firing produces the new marking where s_0 and s_1 are empty and a new token is produced both in s_2 and s_3.

The above informal description is probably sufficient to suggest the appropriateness of nets as models of concurrency. The state of a net has an intrinsic distributed nature, and a transition modifies a local part of the state, making natural to allow the concurrent firing of transitions when they consume mutually disjoint sets of tokens (e.g., t_2 and t_3 in the state produced after the firing of t_0). A situation of mutual exclusion is naturally represented by two transitions competing for a single token, like t_1 and t_2 in the figure. The nondeterministic behaviour of a net is intimately connected with such kind of situations.

A limit in the expressiveness of Petri nets is represented by the fact that a transition can only consume and produce tokens, and thus a net cannot express in a natural way activities which involves non-destructive testing operations on resources. In the following we review contextual nets and inhibitor nets, two generalizations of classical nets aimed at overcoming this limit.

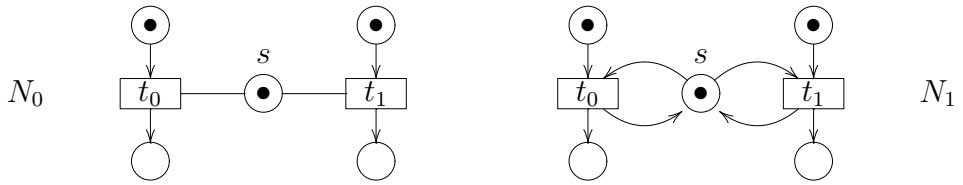


Figure 1.2: Traditional nets do not allow for concurrent read-only operations.

Contextual nets.

Contextual nets [MR95], also called nets with test arcs in [CH93], activator arcs in [JK95] or read arcs in [Vog96], extend classical nets with the possibility of checking for the presence of tokens which are not consumed. Concretely, besides the usual preconditions and postconditions, a transition of a contextual net has also some *context* conditions, which specify that the presence of some tokens in certain places is necessary to enable the transition, but such tokens are not affected by the firing of the transition. In other words, a context can be thought of as a resource which is *read but not consumed* by the transition, in the same way as preconditions can be considered being read and consumed and postconditions being simply written. Coherently with this view, the same token can be used as context by many transitions at the same time and with multiplicity greater than one by the same transition. For instance, the situation of two agents, which access a shared resource in a read-only manner can be modelled directly by the contextual net N_0 of Figure 1.2, where the transitions t_0 and t_1 use the place s as context. According to the informal description of the semantics of contextual nets, in N_0 the transitions t_0 and t_1 can fire concurrently. Notice that in the pictorial representation of a contextual net, directed arcs represent, as usual, preconditions and postconditions, while, following [MR95], non-directed (usually horizontal) arcs are used to represent context conditions.

It is worth remarking that the naïve technique of representing the reading of a token via a consume/produce cycle may cause a loss in concurrency, even if the same markings are reachable. For instance, in the net N_1 of Figure 1.2, the two transitions t_0 and t_1 cannot read the shared resource s concurrently, but their accesses must be serialized.

The ability of faithfully representing the "reading of resources" allows contextual nets to model many concrete situations more naturally than classical nets. In recent years they have been used to model the concurrent access to shared data (e.g., reading in a database) [Ris94, DFMR94], to provide concurrent semantics to concurrent constraint (CC) programs [MR94, BHMR98] where several agents may access a common store, to model priorities [JK91] and to compare temporal efficiency in asynchronous systems [Vog97a].

As a concrete example we hint how contextual nets allow to face the problem of serializability of concurrent transactions in the field of databases [Ris94]. Roughly

speaking a *transaction* implements an "activity" on the database by means of a sequence of read and write operations, which, starting from a consistent state of the database, produces a new consistent state. When several agents access a database it is natural to allow transactions to be executed concurrently. However in this way the operations of distinct transactions may interleave in an arbitrary way and this may cause interferences which lead the database to an inconsistent state. To ensure that the consistency of the state is maintained we must show that the considered interleaving, called a *scheduling* of the transactions, is indeed equivalent to a serial scheduling of the same transactions, where no interleaving is admitted.

A transaction $T = t_1; \ldots; t_n$ where each t_i is a read or write operation can be represented by means of a contextual net process (formally defined later, in Chapter 2). For instance a transaction $T' = t_1; t_2; t_3$, given by

$t_1 : \mathsf{read}(x, y);$
$t_2 : \mathsf{write}(z);$
$t_3 : \mathsf{read}(z)$

is represented by the process in Figure 1.3. The places labelled by s', in the left part of the figure, represent the internal state of the agent performing the transaction, which evolves step by step. The places labelled by x, y and z, in the top part of the figure, represent the initial values of the variables. A $\mathsf{read}(x_1, \ldots, x_n)$ operation is represented by a single transition (e.g., t_1 in the figure) which reads the places corresponding to the current values of $x_1, \ldots, x_n$. A $\mathsf{write}(x)$ operation is instead represented by two transitions (e.g., t_{21} and t_{22} in the figure), one consuming the previous value of x and the other producing a new value for x. The two transitions are distinct since the write operation does not depend on the previous value of x but only destroys it. Observe that both read and write operations consume the place corresponding to the previous internal state of the agent and produce the new state.

Given two transactions T' and T'' and a scheduling consisting of a possible interleaving of their actions, we can construct, in the same way, a corresponding contextual net process. In [Ris94] it is shown that two such schedulings lead to the same final state for all the possible interpretation of the transitions if and only if the corresponding processes are isomorphic. Hence a scheduling of T' and T'' is serializable if and only if the corresponding process is isomorphic either to the process for $T'; T''$ or to the process for $T''; T'$.

For example, let T' be the transaction defined above and let T'' be the transaction consisting of the only operation

$t_4 : \mathsf{read}(z)$

Figure 1.4 reports the processes corresponding to the schedulings (a) $t_1; t_4; t_2; t_3$ and and (b) $t_1; t_2; t_4; t_3$, of T' and T''. The two processes are not isomorphic, witnessing that the two schedulings are not equivalent. Indeed observe that in the first scheduling t_4 read the initial value of the variable z, while in the second scheduling t_4 read the value written in z by t_2.

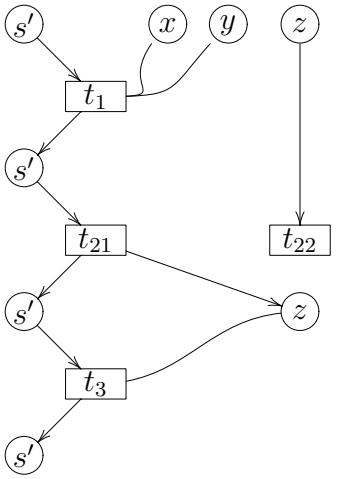


Figure 1.3: The contextual net process corresponding to a transaction.

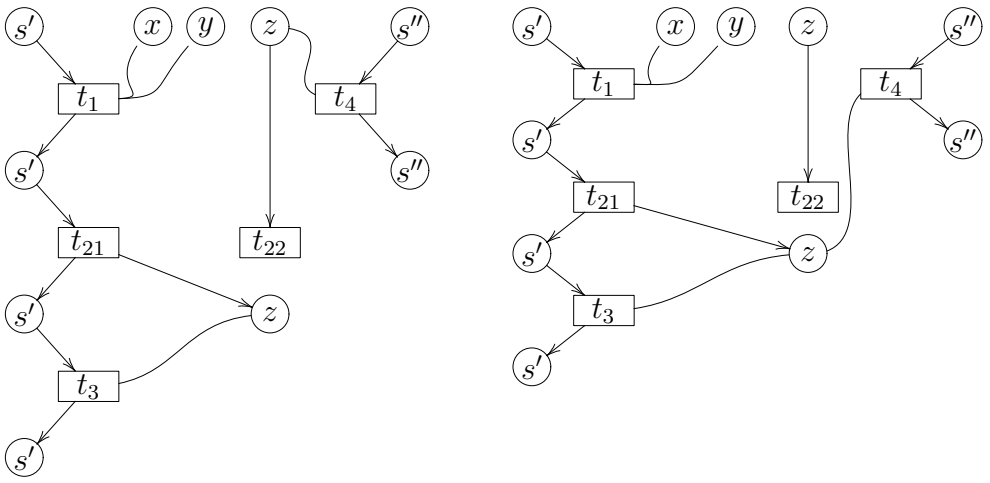


Figure 1.4: The contextual net processes for two possible schedulings of the transactions T' and T''.

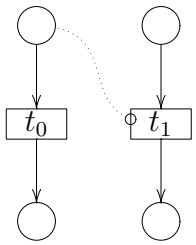


Figure 1.5: Representing priorities via inhibitor arcs.

Instead, it is not difficult to realize that the serial schedulings $T'; T''$ and $T''; T'$ have associated exactly the processes in Figure 1.4 (a) and (b), respectively. Therefore both the considered schedulings $t_1; t_4; t_2; t_3$ and $t_1; t_2; t_4; t_3$ of T' and T'' are serializable.

Inhibitor nets

Inhibitor nets (or nets with *inhibitor arcs*) [AF73] further generalize contextual nets with the possibility of checking not only for the presence, but also for the *absence* of tokens in a place. For each transition an *inhibitor set* is defined and the transition is enabled only if no token is present in the places of its inhibitor set. When a place s is in the inhibitor set of a transition t we say that s *inhibits* (the firing of) t.

While, at a first glance, this could seem a minor extension, it definitely increases the expressive power of the model. In fact, many other extensions of ordinary nets can be simulated in a direct way by using nets with inhibitor arcs (see, e.g., [Pet81]). For instance, this is the case for prioritized nets [Hac76, DGV93], where each transition is assigned a priority and whenever two transitions are enabled at a given marking, then the transition with the highest priority fires. To have an informal idea of how the encoding may proceed, two transitions t_0 and t_1 of a prioritized net, with t_0 having the highest priority, may be represented by simply forgetting the priorities and adding an inhibitor arc, as depicted in Figure 1.5. Observe that the fact that a place s inhibits a transition t is graphically represented by drawing a dotted line from s to t, ending with an empty circle.

Indeed the crucial observation is that traditional nets can easily simulate all the operation of RAM machines, with the exception of the *zero-testing*. Enriching nets with inhibitor arcs is the simplest extension which allows to overcome this limit, thus giving the model the computational power of Turing machines [Age74, Kel72, Kos73].

It is worth stressing that while from the point of view of expressiveness Turing completeness is surely a desirable property, when analysis is concerned it may become a problem since many properties which were known to be decidable for Petri nets become undecidable. The thesis [Bus98] shows that this problem can be partially overcome by singling out restricted, but still interesting, subclasses of inhibitor

nets for which algorithms and techniques of classical nets may be extended.

1.2 Graph Grammars

The *theory of graph grammars* (or of *graph transformation systems*) studies a variety of formalisms which extend the theory of formal languages in order to deal with structures more general than strings, like graphs and maps. A graph grammar allows one to describe finitely a (possibly infinite) collection of graphs, i.e., those graphs which can be obtained from a start graph through repeated applications of graph productions. A *graph production* is a rule of the kind $p : L \rightsquigarrow R$, that specifies that, under certain conditions, once an occurrence (a *match*) of the left-hand side L in a graph G has been detected, it can be replaced by the right-hand side R. The form of graph productions, the notion of match and in general the mechanisms stating how a production can be applied to a graph and what the resulting graph is, depend on the specific graph rewriting formalism.

Graph grammars have been deeply studied following the classical lines of the theory of formal languages, namely focusing on the properties of the generated graph languages and on their decidability; briefly, on the *results* of the generation process. However, quite early, graph grammars have been recognized as a powerful tool for the specification of concurrent and distributed systems. The basic idea is that the state of many distributed systems can be naturally represented (at a suitable level of abstraction) as a graph, and (local) transformations of the state can be expressed as production applications. Thus a stream of research, mainly dealing with the algebraic approaches, has grown, concentrating on the *rewriting process itself*, seen as a representation of systems computations, studying properties of derivation sequences, their transformations and equivalences.

Double-pushout algebraic approach

Here we follow the so-called *double-pushout* (DPO) *algebraic approach* (DPO *approach*, for short) [Ehr87], where the basic notions of production and direct derivation are defined in terms of constructions and diagrams in a suitable category. Consequently, the resulting theory is very general and flexible, easily adaptable to a very wide range of structures, simply by changing the underlying category. In this thesis we will concentrate on directed (typed) graphs, but it is easy to realize that the results immediately extends also to hypergraphs. The generalization to more general structures and to abstract categories (e.g., to high level replacement systems [EHKPP91]) is instead less trivial and left as a matter of further investigation.

In the DPO approach a graph production consists of a left-hand side graph L, a right-hand side graph R and a (common) interface graph K embedded both in R and in L, as depicted in the top part of Figure 1.6. Informally, to apply such a rule to a graph G we must find a *match*, namely an occurrence of its left-hand side L in G.

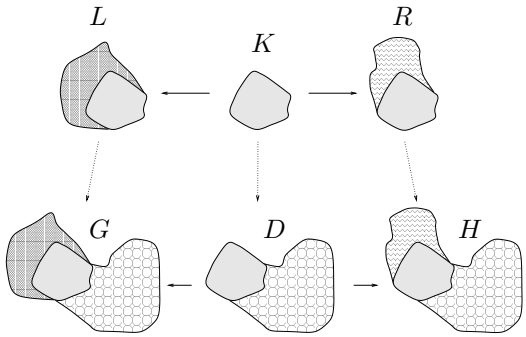


Figure 1.6: A (double-pushout) graph rewriting step.

The rewriting mechanism first removes the part of the left-hand side L which is not in the interface K producing the graph D, and then adds the part of the right-hand side R which is not in the interface K, thus obtaining the graph H. Formally, this is obtained by requiring the two squares in Figure 1.6 to be pushouts in the category of graphs, hence the name of the approach. The interface graph K is "preserved": it is necessary to perform the rewriting step, but it is not affected by the step itself, and as such it corresponds to the context of a transition in contextual nets. Notice that the interface K plays a fundamental role in specifying how the right-hand side has to be glued with the graph D. Working without contexts, which for graph grammars would mean working with productions having an empty interface graph K, the expressive power would drastically decrease: only disconnected subgraphs could be added.

A basic observation belonging to the folklore (see, e.g., [Cor96]) regards the close relationship existing between graph grammars and Petri nets. Basically a Petri net can be viewed as a graph transformation system that acts on a restricted kind of graphs, namely discrete, labelled graphs (that can be considered as sets of tokens labelled by places), the productions being the transitions of the net. In this view, general graph transformation systems are a *proper* extension of ordinary Petri nets in two dimensions:

1. they allow for the specification of *context-dependent rewritings*, where part of the state is required, but not affected by the rewriting step;

2. they allow for a *more structured description of the state*, that is an arbitrary, possibly non-discrete, graph.

The relevance of the first capability in the representation of concurrent accesses to shared resources has been already stressed when we have presented contextual nets.

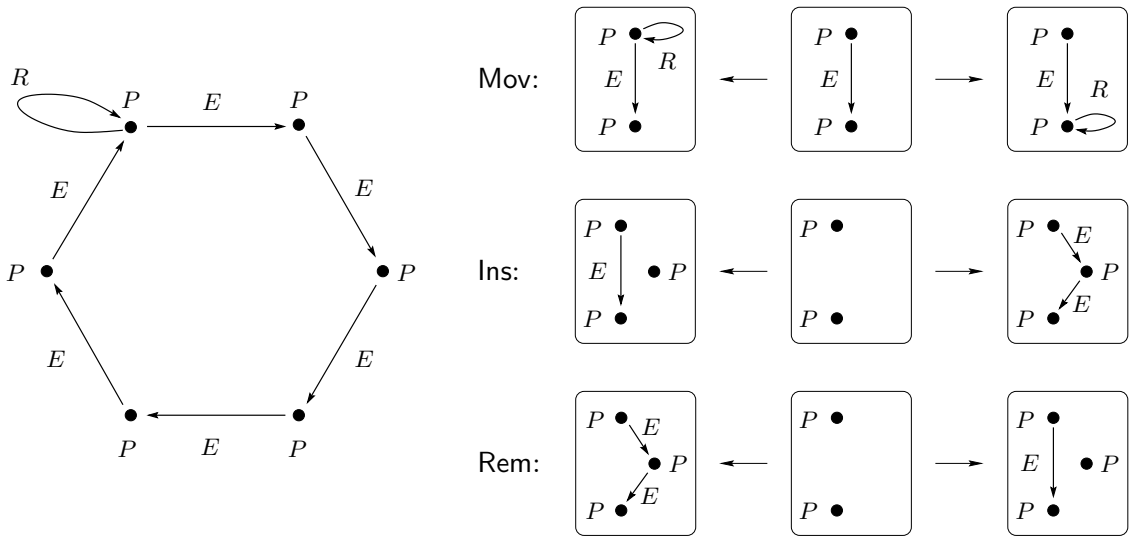


Figure 1.7: A graph grammar representation of system RING.

As for the second capability, even if multisets may be sufficient in many situations, it is easy to believe that graphs are more appropriate when one is interested in giving an explicit representation of the interconnections among the various components of the systems, e.g. if one wants to describe the topology of a distributed system and the way it evolves.

Furthermore, in a graph transformation system each rewriting step is required to preserve the consistency of the graphical structure of the state, namely each step must produce a well-defined graph. In the DPO approach this is expressed by the so-called *application condition* for productions, which, technically, ensures the existence of the left-pushout of Figure 1.6 for the given match and thus the applicability of the rule. The restrictions to the behaviour which are imposed by such requirement have often a very natural interpretation in the modelled system.

Let us present an example, which even if very simple, may help the reader to get convinced of the gain in expressiveness determined by the use of graphs in place of multisets. Consider a system RING consisting of a set of processes using a single common resource which is accessed in mutual exclusion. The processes are connected to form a ring and the right to access the resource passes from one process to its successor in the ring.

This situation can be modelled by the graph grammar consisting of the start graph depicted in the left part of Figure 1.7 and the production Mov. The state of the system is naturally represented as a graph where processes are nodes labelled by P and the connections are established by means of edges, labelled by E. The fact that a process has the right to access the resource is represented by adding a loop R to the corresponding node. The behaviour of the system is represented via production Mov, which moves the loop from the node currently holding the resource its successor in the ring, preserving their connection.

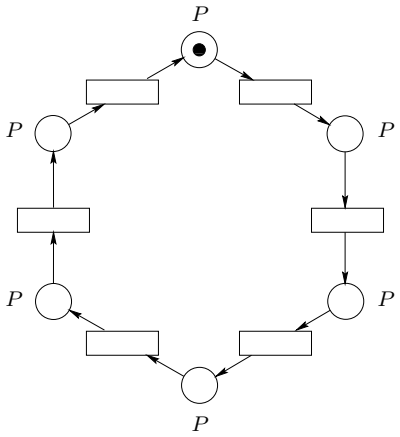


Figure 1.8: A Petri net representation of system RING.

One can observe that the same system can be represented also by the Petri net of Figure 1.8, where processors are modelled as places and the right to access the resource is given by the token moving around the ring. This representation is surely reasonable and simple, but conceptually it seems more natural to represent the topology of the system as part of the state and to model the moving of the resource from a processor the its successor via a single rule (ideally applied by some supervisor of the system).

At a more concrete level one can notice that the net representation is not very flexible. In fact, consider a slightly different system, where processors are not always part of the ring. The topology of the system can change dynamically, since a processor can (ask to) be inserted and removed from the ring, with the constraint that the processor holding the resource cannot be removed form the ring. While the net of Figure 1.8 cannot be easily extended to deal with the new situation, the graph grammar of Figure 1.7 can be adapted by just including the two new rules Add and Rem for adding and removing a processor from the ring, respectively.

It is worth noting that, since the consistency of the graphical structure of the state must be preserved, the rule Rem cannot be applied to the processor holding the resource because after its removal the loop R would remain *dangling*. Therefore the satisfaction of the constraint imposing that the processor holding the resource cannot be removed is entailed by the basic properties of the rewriting mechanism.

The above observation leads us to another crucial remark. As shown by the example, to ensure that after each rewriting step the new state is a well defined graph, before applying a production q which removes a node n we must be sure that any edge with source or target in n is removed by q as well, otherwise it would remain dangling in the new state. This requirement, a part of the application condition called *dangling condition* in the DPO approach, can be interpreted by thinking that the edges with source or target in the node n, not removed by q, *inhibits* the application

of q. This establishes a deep connection between inhibitor nets and graph grammars which will be exploited throughout the thesis. Among other things this suggests the appropriateness of graph grammars to model phenomena which can be expressed by using inhibitor nets.

1.3 Truly concurrent semantics of Petri nets

In this section we describe the approaches to the semantics of Petri nets which represent the starting point of the results presented in this thesis. Then we hint at the possible applications of such semantics and we comment on the role of category theory in its development.

1.3.1 Semantics of Petri nets

Along the years Petri nets have been equipped with several semantics, aimed at describing appropriately, at the right degree of abstraction, the truly concurrent nature of net computations. The approach that we propose as a paradigm, comprises the semantics based on *deterministic processes*, whose origin dates back to an early proposal by Petri himself [Pet77] and the semantics based on the *nondeterministic unfolding* introduced in a seminal paper by Nielsen, Plotkin and Winskel [NPW81], and shows how the two may be reconciled in a very satisfactory framework.

Deterministic process semantics

The notion of *deterministic process* naturally arises when trying to give a truly concurrent description of net computations, taking explicitly into account the *causal relationships* which rule the occurrences of events in *single* computations.

Apart from Best-Devillers processes [BD87] which do not account for causality, the prototypical example of process for Petri nets is given by the *Goltz-Reisig processes* [GR83]. A Goltz-Reisig process of a net N is a (deterministic) occurrence net O, i.e. a (safe) finite net satisfying suitable acyclicity and conflict freeness properties, plus a mapping to the original net $\varphi : O \to N$. The flow relation induces a partial order on the elements of the net O, which can be naturally interpreted as causality. The mapping essentially labels the places and transitions of O with places and transitions of N, in such a way that places in O can be thought of as tokens in a computation of N and transitions of O as occurrences of transition firings in such computation.

A limitation of Goltz-Reisig processes resides in the fact that they cannot be endowed with an operation of sequential composition, meaningful with respect to causality. The naïve attempt of concatenating a process φ_1 with target u with a second one φ_2, with source u, by merging the source and the target immediately fails. In fact, there are in general, many ways of putting in one-to-one correspondence the

maximal places of φ_1 with the minimal places of φ_2, respecting the labelling, and they lead to different resulting processes of the net. The problem is that the places of a process represent *tokens* produced in a computations, and tokens in the same place should not be confused since they may have different (causal) histories.

Concatenable processes are defined in [DMM96] as Goltz-Reisig processes in which minimal and maximal places carrying the same label are linearly ordered. Such an ordering allows one to disambiguate token identities and thus an operation of concatenation can be safely defined. This brings us to the definition of a category $\mathbf{CP}[N]$ of concatenable processes, in which objects are markings (states of the net), arrows are processes (computations) and arrow composition models the sequential composition of computations. It turns out that such category is a *symmetric monoidal category*, in which the tensor product represents faithfully the parallel composition of processes.

Unfolding semantics

A deterministic process specifies only the meaning of a single, deterministic computation of a net. Nondeterminism is captured only implicitly by the existence of several different "non confluent" processes having the same source. Instead the accurate description of the fine interplay between concurrency and nondeterminism is one of the most interesting features of Petri nets.

An alternative classical approach to the semantics of Petri nets is based on an *unfolding construction*, which maps each net into a single denotational structure, representing, in an unambiguous way, all the possible events that can occur in all the possible computations of the net and the relations existing between them. This structure expresses not only the causal ordering between the events, but also gives an explicit representation of the branching (choice) points of the computations.

In the seminal work of Nielsen, Plotkin and Winskel [NPW81], the denotation of a *safe net* is defined as a *coherent finitary prime algebraic Scott domain* [Sco70], or dI-domain [Ber78] (briefly *domain*), via a construction which first unfolds the net into a (nondeterministic) occurrence net which is then abstracted to a prime event structure. Building on such result, Winskel [Win87a] proves the existence of a chain of coreflections (a particularly nice kind of adjunction), leading from the category **S-N** of safe (marked) P/T nets to the category **Dom** of finitary prime algebraic domains, through the categories **O-N** of occurrence nets and **PES** of prime event structures.

$$\mathbf{S\text{-}N} \underset{\mathscr{U}}{\overset{\mathscr{I}_{\mathbf{Occ}}}{\underset{\perp}{\leftarrow}}} \mathbf{O\text{-}N} \underset{\mathscr{E}}{\overset{\mathscr{N}}{\underset{\perp}{\leftarrow}}} \mathbf{PES} \underset{\mathscr{L}}{\overset{\mathscr{P}}{\underset{\sim}{\leftarrow}}} \mathbf{Dom}$$

The first step unfolds a safe net N into a *nondeterministic occurrence* net $\mathscr{U}(N)$. Such a net can be understood as a nondeterministic process of the net N where each transition represents a precise firing of a transition in N, and places represent

occurrences of tokens in the places of N. Differently from deterministic processes, the unfolding can be infinite and can contain (forward) conflicts. In this way it can take advantage of its branching (tree-like) structure to represent *all* the possible computations of the original net N.

The subsequent step abstracts the occurrence net obtained via the unfolding construction to a *prime event structure*. Recall that *prime event structures* (PES) are a simple event based model of (concurrent) computations in which events are considered as atomic, indivisible and instantaneous steps, which can appear only once in a computation. An event can occur only after some other events (its causes) have taken place and the execution of an event can inhibit the execution of other events. This is formalized via two binary relations: *causality*, modelled by a partial order relation $\leq$ and *conflict*, modelled by a symmetric and irreflexive relation $\#$, hereditary with respect to causality. The PES semantics is obtained from the unfolding simply by forgetting the places, but remembering the basic relations of causality and conflict between transitions that they induce.

The last step of Winskel's construction shows that the category of prime event structures is equivalent to the category of domains. An element of the domain corresponding to a PES is a set of events (configuration) which can be understood as a possible computation in the PES. The order (which is simply set inclusion) represents a computational order: if $C \sqsubseteq C'$, then C can evolve and become C'.

In [MMS92, MMS97] it has been shown that essentially the same construction applies to the wider category of *semi-weighted* nets, i.e., P/T nets in which the initial marking is a set and transitions can generate at most one token in each post-condition. It is worth noting that, besides being more general than safe nets, semi-weighted nets present the advantage of being characterized by a "static condition", not involving the behaviour but just the structure of the net. Figure 1.9 shows an example of semi-weighted P/T net, which is not safe. The generalization of Winskel's construction to the whole category of P/T nets requires some original technical machinery and allows one to obtain a proper adjunction rather than a coreflection [MMS92].

Reconciling deterministic processes and unfolding

Since the unfolding of a net is essentially a nondeterministic process that completely describes the behaviour of the net, one would expect that a relation exists between the unfolding and the deterministic process semantics. Indeed, as shown in [MMS96], the domain associated to a net N through the unfolding construction can be equivalently characterized as the set of deterministic processes of the net starting from the initial marking, endowed with a kind of prefix ordering. This result is stated in an elegant categorical way. The comma category $\langle m \downarrow \mathbf{CP}[N] \rangle$, where m is the initial marking of the net, is shown to be a preorder (more precisely, this is true for semi-weighted nets, while for general nets a slight variation of the notion of process has to be considered, called *decorated process*). Intuitively, the elements of such preorder

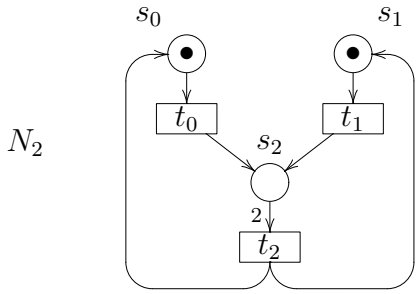
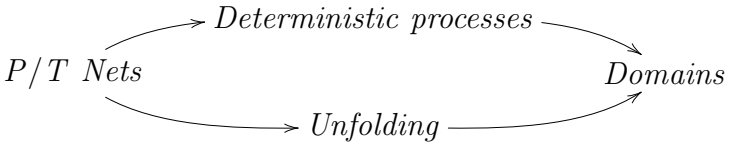


Figure 1.9: A semi-weighted P/T net, which is not safe.

are computations starting from the initial state, and if φ_1 and φ_2 are elements of the preorder, we have $\varphi_1 \preceq \varphi_2$ if φ_1 can be extended to φ_2 by performing appropriate steps of computation. Finally, the ideal completion of such preorder, which can be seen as a representation of the (finite and infinite) computations of the net, is shown to be isomorphic to the domain associated to the unfolding.



Although not central in this thesis, we recall that there exists a third approach to Petri net semantics which fits nicely in the above picture, called *algebraic semantics*. Roughly speaking, the algebraic approaches to Petri net semantics, originated from [MM90], characterize net computations as an equational term algebra, freely generated starting from the basic elements of the net and having (suitable kinds of) monoidal categories as models. For instance, the category of concatenable processes can be given a purely algebraic and completely abstract characterization as the *free symmetric strict monoidal category* generated by the net N, modulo some suitable axioms [Sas96]. In particular the distributivity of tensor product and arrow composition in monoidal categories is shown to capture the basic facts about net computations.

1.3.2 A view on applications

The semantical framework for Petri nets illustrated before, besides being elegant and satisfactory from a theoretical point of view, represents a basis on which the bridge towards applications can be settled. The discussed constructions provide a description of the behaviour of a net which is, in general, "infinite" and thus apparently difficult to deal with. However, on top of them one can build more abstract

"approximated" descriptions of the behaviour which turn out to be useful in the verification of several properties of the modelled systems.

The deterministic process and event structure semantics represent the basis for the definition of *history preserving bisimulation* [RT88, BDKP91], a bisimulation-like observational equivalence suited to deal with concurrent systems. For instance it is a congruence with respect to some kind of refinement operations and it preserves the amount of autoconcurrency. History preserving bisimulation is known to be decidable for n-safe nets, where the number of tokens in each place is bounded by n. Algorithms for checking the bisimilarity of two nets and to get a minimal realization in a class of bisimilar systems have been proposed in the literature [Vog91, MP98].

Furthermore, although the unfolding is infinite for non trivial nets, as observed by McMillan in his PhD thesis [McM93], limiting attention to (n-)safe nets it is possible to construct a finite initial part of the unfolding which contains as much information as the unfolding itself, the so-called *finite complete prefix*. The advantage of complete prefixes is that they can be much smaller of the state space of the system, when they are generated via "clever" algorithms, and, as such, they represent a useful technique to attack the well-known state explosion problem of model-checking techniques. Moreover, the information about causality, concurrency and distribution contained in the unfolding may be used to verify properties expressed in local logics, which allow to reason on the knowledge that each component has of the global state of the system. The unfolding technique has been applied to the verification of circuits, telecommunication systems, distributed algorithms, etc.

1.3.3 The role of category theory

Category theory originated as an abstract theory of mathematical structures and of the relationships between structures, aimed at giving a unified view of "similar" results from disparate areas of mathematics.

The categorical language, with its elegance and abstractness, has been exploited in Computer Science as a tool to give alternative systematic formulations of existing theories, making clear their real essence and disregarding unnecessary details, and as a guidance for the development and justification of new concepts, properties and results. The advantages of the use of category theory in Computer Science are well summarized in [Gog91]. Here we try to point out some aspects which are particularly relevant to our approach.

Considering categories of systems, one is lead to introduce an appropriate notion of morphism between systems, typically formalizing the idea of "simulation". Then expressing the semantics via a *functor* means to define the semantical transformation consistently with such notion: a morphism between two systems must yield a morphism between their models.

Moreover, the notion of *universal construction* (e.g., *adjunction*, *reflection*, *coreflection*) provides a formal way to justify the naturality of the semantics, by expressing its "optimality". It is often the case that an obvious functor maps models back

into the category of systems (e.g., this happens for Petri nets, where occurrence nets are particular nets and thus such a functor is simply the inclusion). Consequently the semantics can be defined naturally as the functor in the opposite direction, forming an adjunction, which (if it exists) is unique up to natural isomorphism. In other words, once one has decided the notion of simulation, there is a unique way to define the semantics consistently with such notion.

Finally, several composition operations can be naturally expressed at categorical level as limit/colimit constructions (*products*, *sums*, *pushouts*, *pullbacks*, just to cite a few). For instance, a pushout construction can be used to compose two nets, merging some part of them, obtaining a kind of generalized nondeterministic composition, while synchronization of nets can be modelled as a product (see [Win87a, MMS97]).

Remarkably, since left/right adjoint functors preserve colimits/limits, a semantics defined via an adjunction turns out to be compositional with respect to such operations.

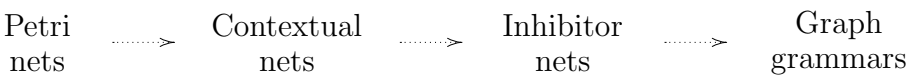
1.4 From Petri nets to graph grammars: an overview of the thesis

Inspired by the close relationship between graph grammars and Petri nets, in order to present graph grammars as a formalism for the specification of concurrent/distributed systems alternative to Petri nets, the thesis explores the possibility of developing a theory of concurrency for graph transformation systems recasting in this more general framework notions, constructions and results from Petri nets theory. More precisely, the thesis investigates the possibility of generalizing to graph grammars the nice semantical framework described for Petri nets in the previous section, by endowing them with deterministic process and unfolding semantics, as well as with more abstract semantics based on (suitable of extensions of) event structures and domains.

Remarkably, the reason for which graph grammars represent an appealing generalization of Petri nets, namely the fact that they extend nets with some non-trivial features, makes non-trivial also such generalization. In fact, the main complications which arise in the treatment of graph grammars are related on the one hand to the possibility of expressing contextual rewritings, and on the other hand to the necessity of preserving the consistency of the graphical structure of the state, a constraint which leads to the described "inhibiting effects" between productions applications.

We already observed that contextual nets, where a transition can test for the presence of a token without consuming it, share with graph grammars the ability of specifying a "context-sensitive" firing of events. Furthermore inhibitor nets, where the presence of a token in a place can disable a transition, allow one to model a kind of dependency between events analogous to the one which arises in DPO graph grammars due to the requirement of preserving the consistency of the state.

Informally, we can organize the considered formalisms in an ideal chain leading from Petri nets to graph transformation systems as follows



Motivated by the idea that contextual nets and nets with inhibitor arcs can serve as a bridge for transferring notions and results from classical nets to graph grammars, the first part of the thesis concentrates on these intermediate models, while the second part is devoted to the study of graph grammars.

Differently from what happens for ordinary nets, we define an unfolding semantics (essentially based on nondeterministic processes) before developing a theory of deterministic processes. To understand why we proceed in this way observe that for traditional nets the only source of nondeterminism is the the presence of pairs of different transitions with a common precondition, and therefore there is an obvious notion of "deterministic net". When considering contextual nets, inhibitor nets or graph grammars the situation becomes much more involved: the dependencies between event occurrences cannot be described only in terms of causality and conflict, and the deterministic systems cannot be given a purely syntactical characterization. Consequently, a clear understanding of the structure of nondeterministic computations becomes essential to be able to single out which are the good representatives of deterministic computations.

The core of the theory developed for each one of the considered models is the formalization of the kind of dependencies among events in their computations and the definition of an appropriate notion of event structure for faithfully modelling such dependencies.

Contextual nets and asymmetric conflicts

When dealing with contextual nets, the crucial point is the fact that the presence of context conditions leads to *asymmetric conflicts* or *weak dependencies* between events. Consider, for instance, the net N_3 of Figure 1.10, with two transitions t_0 and t_1 such that the same place s is a context for t_0 and a precondition for t_1. The possible firing sequences are given by the firing of t_0, the firing of t_1 and the firing of t_0 followed by t_1, denoted $t_0; t_1$, while $t_1; t_0$ is not allowed. This represents a new situation not arising within ordinary net theory: t_0 and t_1 are neither in conflict nor concurrent nor causal dependent. Simply, as for a traditional conflict, the firing of t_1 prevents t_0 to be executed, so that t_0 can never follow t_1 in a computation. But the converse is not true, since t_1 *can* fire after t_0. This situation can be interpreted naturally as an *asymmetric conflict* between the two transitions. Equivalently, since t_0 precedes t_1 in any computation where both transitions are executed, in such computations t_0 acts as a cause of t_1. However, differently from a true cause, t_0 is not necessary for t_1 to be fired. Therefore we can also think of the relation between the two transitions as a *weak* form of *causal dependency*.

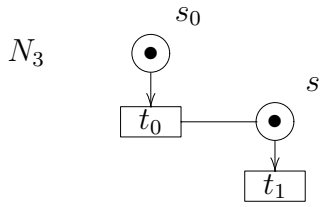


Figure 1.10: Asymmetric conflict in contextual nets.

Prime event structures and in general Winskel's event structures result inadequate to give a direct representation of situations of asymmetric conflict. To give a faithful representation of the dependencies between events arising in contextual nets we introduce *asymmetric event structures* (AES's), a generalization of PES's where symmetric conflict is replaced by a relation $\nearrow$ modelling *asymmetric conflict*. An AES allows us to specify the new kind of dependency described above for transitions t_0 and t_1 of the net in Figure 1.10 simply as $t_0 \nearrow t_1$.

The notion of asymmetric conflict plays an essential role both in the ordering of the configurations of an AES, which is different from set-inclusion, and in the definition of (deterministic) occurrence contextual nets, which are introduced as the net-theoretical counterpart of (deterministic) AES's. Then the entire Winskel's construction naturally lifts to contextual nets.

Inhibitor nets and the disabling-enabling relation

When considering inhibitor nets, the nonmonotonic features related to the presence of inhibitor arcs (negative conditions) make the situation far more complicated. First if a place s is in the post-set of a transition t', in the inhibitor set of t and in the pre-set of t_0 (see the net N_4 in Figure 1.11), then the execution of t' inhibits the firing of t, which can be enabled again by the firing of t_0. Thus t can fire before or after the "sequence" $t'; t_0$, but not in between the two transitions. Roughly speaking there is a sort of atomicity of the sequence $t'; t_0$ with respect to t.

The situation can be more involved since many transitions $t_0, \ldots, t_n$ may have the place s in their pre-set (see the net N_5 in Figure 1.11). Therefore, after t' has been fired, t can be re-enabled by any of the conflicting transitions $t_0, \ldots, t_n$. This leads to a sort of *or*-causality. With a logical terminology we can say that t causally depends on the implication $t' \Rightarrow t_0 \vee t_1 \vee \ldots \vee t_n$.

To face these additional complications we introduce *inhibitor event structures* (IES's), which enrich asymmetric event structures with a ternary relation, called *DE-relation (disabling-enabling relation)*, denoted by $\mapsto(\cdot,\cdot,\cdot)$. Such a relation is used to model the previously described situation as $\mapsto(\{t'\}, t, \{t_0, \ldots, t_n\})$. The *DE-relation* is sufficient to represent both causality and asymmetric conflict and thus concretely it is the only relation of a IES.

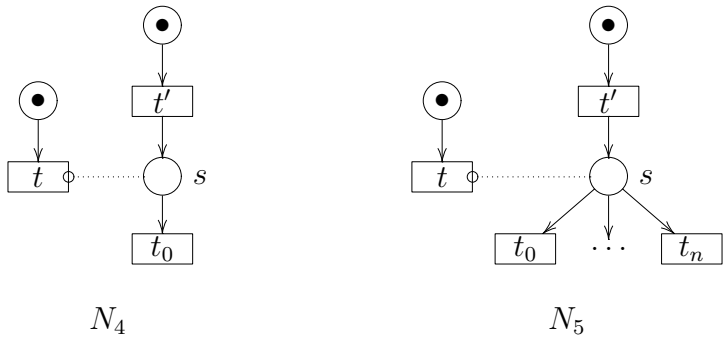


Figure 1.11: Two basic nets with inhibitor arc.

Remarkably, computations of an inhibitor net (and thus configurations of an IES) involving the same events may be different from the point of view of causality. For instance, in the basic net N_4 of Figure 1.11 there are two possible orders of execution of transitions t, t' and t_0, namely $t; t'; t_0$ and $t'; t_0; t$, and while in the first case it is natural to think of t as a cause of t', in the second case we can imagine instead that t_0 (and thus t') causes t. To take into account correctly this further information, both configurations of IES's and processes of inhibitor nets are enriched with a so-called *choice relation* specifying which of the possible computations we are referring to.

The unfolding construction for inhibitor nets makes an essential use of the construction already developed for contextual nets. The main problem emerges in the passage from occurrence inhibitor net to IES's where the backward steps is impossible, basically because of complications due to the complex kind of causality expressible in IES's. More technically, the construction associating an inhibitor event structure to an occurrence net is functorial, but does not give rise to a categorical coreflection.

Lifting the results to graph grammars

When we finally turn our attention to graph grammars we are rewarded of the effort spent in the first part, since basically nothing new has to be invented. Inhibitor event structures are expressive enough to model the structure of graph grammar computations and the theory developed for inhibitor nets smoothly lifts, at the price of some technical complications, to grammars. Furthermore, not only the process and the unfolding semantics proposed for a graph grammars are shown to agree, but the theory developed in this thesis is shown to be consistent also with the classical theory of concurrency for DPO grammar in the literature, basically relying on *shift-equivalence* [Kre77, CMR$^+$97, CEL$^+$96b]. We hope that this can be considered a valuable contribution to the understanding of the theory of concurrency for DPO graph transformation.

1.4.1 The general approach

For each one of the mentioned formalisms, namely contextual nets, inhibitor nets and graph grammars we develop a similar theory by following a common schema which can be described as follows:

1. We define a *category of systems* **Sys**. The morphisms, which basically origins from an algebraic view of the systems, can be interpreted as simulations.

2. We develop an *unfolding semantics*, expressed as a coreflection between **Sys** and a subcategory **O-Sys**, where objects are suitable systems exhibiting an acyclic behaviour.

   From the unfolding we extract an (appropriate kind of) *event structure*, the transformation being expressed as a functor from **O-Sys**, to the considered category of event structures **ES**.

   Finally, a connection is established with domains and traditional PES by showing that the category **ES** of generalized event structures coreflects into the category **Dom** of domains.

   Summing up, we obtain the following chain of functors, leading from systems to event structures and domains

   $$\mathbf{Sys} \underset{\perp}{\overset{\longleftarrow}{\longrightarrow}} \mathbf{O\text{-}Sys} \longrightarrow \mathbf{ES} \underset{\perp}{\overset{\longleftarrow}{\longrightarrow}} \mathbf{Dom} \underset{\sim}{\overset{\longleftarrow}{\longrightarrow}} \mathbf{PES}$$

   The last step in the chain is the equivalence between the categories **Dom** of domains and **PES** of prime event structures, due to Winskel.

   In the case of contextual nets, the step leading from **O-Sys** to **ES** is not only a functor, but a true coreflection.

3. We define a notion of *deterministic process* for systems in **Sys**. Relying on the work developed in the previous point, a general (possibly nondeterministic) *process* of a system $\mathscr{S}$ is introduced as "occurrence system" in **O-Sys**, plus a (suitable kind) of morphism back to the original system $\mathscr{S}$ (the prototypical example of nondeterministic process being the unfolding).

   Then, roughly speaking, a process is *deterministic* if it contains no conflict, or, in other words, if it uniquely identifies a single configuration of the event structure associated to the system, in such a way that it can be seen as the representative of a single deterministic (concurrent) computation of $\mathscr{S}$.

   The deterministic processes of a system $\mathscr{S}$ are turned into a category $\mathbf{CP}[\mathscr{S}]$, by endowing them with a notion of concatenation, modelling the sequential composition of computations.

4. We show that the deterministic process and the unfolding semantics can be reconciled by proving that, as for traditional nets, the comma category $\langle \mathit{Initial\ State} \downarrow \mathbf{CP}[\mathscr{S}] \rangle$, is a preorder whose ideal completion is isomorphic to the domain obtained from the unfolding, as defined at point (2).

1.4.2 Summary of the results

The main achievement of the thesis is the development of a systematic theory of concurrency for graph grammars which contribute to close the gap existing between graph transformation systems and Petri nets.

1. We define a Winskel's style semantics for graph grammars. An unfolding construction is presented, which associates to each graph grammar a nondeterministic occurrence grammar describing its behaviour. Such a construction establishes a coreflection between suitable categories of grammars and the category of occurrence grammars. The unfolding is then abstracted to an inhibitor event structure and finally to a prime algebraic domain (or equivalently to a prime event structure).

2. We introduce a notion of *nondeterministic graph process* generalizing the deterministic processes of [CMR96]. The notion fits nicely in our theory since a graph process of a grammar $\mathscr{G}$ can be defined simply as a (special kind of) grammar morphism from an occurrence grammar into $\mathscr{G}$ (while in [CMR96] an ad hoc mapping was used).

3. We define *concatenable graph processes*, as a variation of (deterministic finite) processes endowed with an operation of concatenation, consistent with the flow of causality, which models sequential composition of computations.

   The appropriateness of this notion is confirmed by the fact that the category $\mathbf{CP}[\mathscr{G}]$ of concatenable processes of a grammar $\mathscr{G}$ turns out to be isomorphic to the classical truly concurrent model of computation of a grammar based on traces of [CMR$^+$97, BCE$^+$99].

4. The event structure obtained via the unfolding is shown to coincide both with the one defined by Corradini et al. [CEL$^+$96b] via a comma category construction on the category of concatenable derivation traces, and with the one proposed by Schied [Sch94], based on a deterministic variant of the DPO approach. These results, besides confirming the appropriateness of the proposed unfolding construction, give an unified view of the various event structure semantics for the DPO approach to graph transformation.

A second achievement is the development of an analogous unifying theory for two widely diffused generalizations of Petri nets, namely *contextual nets* and *inhibitor nets*. While a theory of deterministic processes for these kind of nets was already

available in the literature, the Winskel-style semantics, comprising the unfolding construction, its abstraction to a prime algebraic semantics, as well as its relation with the deterministic process semantics are original.

Finally, we like to mention as a result also the development of a categorical theory of two kind of generalized event structures, namely *asymmetric event structures* and *inhibitor event structures* and of their relation with Winskel's event structures. In this thesis they are presented as a mean for the treatment of (extended nets and) grammars, but the generality of the phenomena they allow to model and their connections with other extensions of event structures in the literature makes us convinced that their applicability goes beyond the considered examples.

1.5 Structure of the thesis

The thesis is divided into two parts. The FIRST PART is devoted to the study of contextual and inhibitor nets, while the SECOND PART, exploiting also some notions and results from the FIRST PART, concentrates on the theory of concurrency of DPO graph transformation systems.

The FIRST PART consists of three chapters. In CHAPTER 2 we introduce some background material. After fixing the basic mathematical notation, we present the notions of contextual and inhibitor net and we review some concurrent semantics proposed in the literature for these generalized kinds of nets. Then we present the work of Winskel on prime event structures, their equivalence with prime algebraic domains and we present some generalizations of the basic notion of event structure proposed in the literature.

CHAPTERS 3 and 4 contain the original contributions of the FIRST PART. CHAPTER 3 proposes a truly concurrent semantics for (semi-weighted) contextual nets by following the general approach described in the INTRODUCTION. First, we define *asymmetric event structures* (AES's) as a new event based model capable to represent the dependencies between events in contextual net computations and we study their relationship with PES's and domains. Then, exploiting an unfolding construction, the semantics of semi-weighted contextual nets is given via a chain of coreflections leading from the category of semi-weighted contextual nets to the category of prime algebraic domains. Finally, we prove that the unfolding and the (deterministic) process semantics of contextual nets can be reconciled by showing that *concatenable processes* allow to give an alternative characterization of the domain obtained from the unfolding.

CHAPTER 4 generalizes (part of) the work in the previous chapter to the case of *inhibitor nets*. This chapter has the same structure as the previous one. First, in order to deal with the greater complexity of inhibitor net computations we introduce a generalization of asymmetric event structures, called *inhibitor event structures* (IES's) and we study their relationship with PES's and domains. Then we give a truly

concurrent semantics for (semi-weighted) inhibitor nets via functorial construction which first associates to each inhibitor net an occurrence inhibitor net. The unfolding is mapped to an inhibitor event structure and then to a prime algebraic domain. Finally we discuss the notion of deterministic process for inhibitor nets arising from our theory, we define their concatenable version and we show that, as for contextual nets, concatenable processes allow us to recover the domain semantics of an inhibitor net as obtained via the unfolding construction.

The second part is divided into four chapters. Chapter 5 provides an introduction to the *algebraic approach* to graph transformation based on the *double-pushout* (dpo) construction, by presenting some (partly revisited) background notions. We first give the basic definitions of dpo graph grammar, rewriting step and derivation. Then we introduce the fundamental notions of the concurrency theory of the dpo approach by presenting the *trace semantics* based on the *shift equivalence*. The chapter is closed by the definition of the *category of graph grammars* considered in this thesis.

The original contributions of the Second Part are presented in Chapters 6-8. Chapter 6 defines an *unfolding semantics* for dpo graph transformation systems. As for inhibitor nets, the unfolding construction is characterized as a categorical coreflection. Then we define a functorial construction which maps the unfolding to an inhibitor event structure, and finally, by the results in Chapter 4, to a prime algebraic domain. This chapter uses many notions and intuitions from Chapter 4 on inhibitor nets (in particular it resorts to inhibitor event structures and to their relationship with domains).

Chapter 7 presents the notion of *concatenable process* for dpo graph transformation systems, and proves that the category of concatenable processes provides a semantics for a graph grammar which is equivalent to the classical abstract truly model of computation, discussed in Chapter 5. We also study the relationship between the process and the unfolding semantics of a graph grammar, by showing, that, as for nets, concatenable graph processes allow to recover the same domain semantics defined in Chapter 6.

Chapter 8 reviews two other event structure semantics which have been proposed in the literature for dpo graph transformation systems in [CEL+96b, Sch94]. By exploiting the results in Chapters 6 and 7, these two alternative event structures are shown to coincide with the one obtained from the unfolding, which thus can be claimed to be "the" event structure semantics of dpo graph transformation.

Both the First Part and the Second Part end with a summary of the presented results and some final remarks. The Conclusions contains a general discussion on the thesis and sketches possible future directions of research.

Finally, the Appendix collects some basic notions of category theory which are used in the thesis.

Origins of the chapters

The truly concurrent semantics for contextual nets presented in CHAPTER 3 first appeared as [BCM98b], while a complete version, including full proofs and the relation with the process semantics, appeared in [BCM99a].

The notion of nondeterministic process and the unfolding construction for graph transformation systems described in CHAPTER 6 can be found in [BCM99b]. The characterization of the unfolding as a universal construction, in the case of semi-weighted grammars, will appear as [BCM99c].

Concatenable graph processes and their relation with the derivation trace semantics, presented in CHAPTER 7, appeared in [BCM98a, BCE+99]. The relation between the process and the unfolding semantics, with a different proof, has been presented in [BCM99b].

Finally, CHAPTER 8 relating our event structure semantics with other proposal in the literature has been extracted from [BCM99b].

Part I

Contextual and inhibitor nets

Chapter 2

Background

This chapter presents some background material which will be used in the First Part. We first fix the basic mathematical notation. Then we present contextual and inhibitor nets and we review some concurrent semantics proposed in the literature for these generalized kinds of nets. In the last part of the chapter we review the work by Winskel on prime event structures, their equivalence with prime algebraic domains and we present some generalizations of the basic notion of event structure proposed in the literature.

2.1 Basic notation

This section introduces the basic mathematical concepts and notation which are used in the rest of the thesis. It is mainly intended to provide a list of symbols and keywords with the aim of fixing the notation for sets, relations, functions and multirelations.

Sets

Without referring to a formal set-theory, we resort to the intuitive notion of *set* as unordered collection of elements and we use the common operations on sets. For instance, given two sets X and X' we can consider their *union* $X \cup X'$, *intersection* $X \cap X'$, *difference* $X - X'$, *cartesian product* $X \times X'$, etc. The *cardinality* of a set X is denoted by $|X|$.

The *powerset* of a set X is denoted by $\mathbf{2}^X$, while $\mathbf{2}^X_{fin}$ denotes the set of finite subsets of X and $\mathbf{2}^X_1$ the set of subsets of X of cardinality at most one (singletons and the empty set $\emptyset$). When $Y \in \mathbf{2}^X_{fin}$ we will write $Y \subseteq_{fin} X$. Given a set of sets $Y \subseteq \mathbf{2}^X$, the *big union* of Y is defined as

$$\bigcup Y = \{y \mid \exists Z \in Y.\ y \in Z\}$$

and similarly the *big intersection* of Y is the set $\bigcap Y = \{y \mid \forall Z \in Y.\ y \in Z\}$. If the

set Y is indexed, i.e., $Y = \{Z_i : i \in I\}$, its big union and intersection are written as $\bigcup_{i\in I} Z_i$ and $\bigcap_{i\in I} Z_i$, respectively.

Relations

Let X and Y be sets. A (binary) *relation* r from X to Y is a subset of the cartesian product of X and Y, namely $r \subseteq X \times Y$. We often use the "infix" and "prefix" notation for relations by writing $x\, r\, y$ and $r(x, y)$, respectively, in place of $(x, y) \in r$. We denote by r^{-1} the *inverse relation* of r, namely $\{(y, x) \mid (x, y) \in r\}$. The *domain* $dom(r)$ and *codomain* $cod(r)$ of the relation r are defined by

$$dom(r) = \{x \in X \mid \exists y \in Y.(x, y) \in r\} \qquad cod(r) = \{y \in Y \mid \exists x \in X.(x, y) \in r\}.$$

Given $X' \subseteq X$ we sometimes write $r(X')$ for the set $\{y \mid \exists x \in X.\ (x, y) \in r\}$, called the *image of* X' *through* r.

The *composition* of two relations $r_1 \subseteq X \times Y$ and $r_2 \subseteq Y \times Z$ is the relation $r_2 \circ r_1 \subseteq X \times Z$, sometimes written $r_1; r_2$, defined as follows

$$r_2 \circ r_1 = \{(x, z) \mid \exists y \in Y.\ (x, y) \in r_1 \ \wedge\ (y, z) \in r_2\}$$

Let us turn our attention to relations $r \subseteq X \times X$ (which we call binary relations over X). By r^+ we denote the *transitive closure* of r, and by r^* the *reflexive and transitive closure* of r. Given $X' \subseteq X$ the symbol $r_{X'}$ indicates the restriction of r to $X' \times X'$, i.e., $r \cap (X' \times X')$.

We say that r is *acyclic* if it has no "cycles" $e_0\, r\, e_1\, r\, \ldots\, r\, e_n\, r\, e_0$, with $e_i \in X$, or, in other words, if r^+ is irreflexive. We call r *well-founded* if it has no infinite descending chains, i.e., $\{e_i\}_{i\in\mathbb{N}} \in X$ such that $e_{i+1}\, r\, e_i$, $e_i \neq e_{i+1}$, for all $i \in \mathbb{N}$. In particular, if r is well-founded it has no (non-trivial) cycles.

The relation r is called a *preorder* if it is reflexive and transitive; it is a *partial order* if it is also antisymmetric. The relation r is an *equivalence* if it is reflexive, transitive and symmetric.

Functions

A *partial function* $f : X \to Y$ is a special relation $f \subseteq X \times Y$ with the property that for any $x \in X$ there is at most one $y \in Y$ such that $(x, y) \in f$, namely $|f(\{x\})| \leq 1$. In this case we write $f(x) = y$ instead of $f(\{x\}) = \{y\}$. Furthermore we write $f(x) = \perp$ and we say that f is *undefined* on x when $x \notin dom(f)$, or equivalently if $f(\{x\}) = \emptyset$. A function $f : X \to Y$ is called *total* if $dom(f) = X$.

Multisets and Multirelations

Let A be a set. A *multiset* of A is a function $M : A \to \mathbb{N}$. Such a multiset is denoted sometimes as a formal sum $M = \sum_{a\in A} n_a \cdot a$, where $n_a = M(a)$. The coefficients 1 are

usually implicit and terms with a zero coefficient are omitted. The set of multisets of A is denoted by μA.

The usual operations and relations on multisets are used. For instance, multiset *union* is denoted by $+$ and defined as $(M + M')(a) = M(a) + M'(a)$; multiset *difference* $(M - M')$ is defined as $(M - M')(a) = M(a) - M'(a)$ if $M(a) \geq M'(a)$ and $(M - M')(a) = 0$ otherwise. We write $M \leq M'$ if $M(a) \leq M'(a)$ for all $a \in A$. If M is a multiset of A, we denote by $[\![M]\!]$ the *flattening* of M, namely the multiset $\sum_{\{a \in A | M(a) > 0\}} 1 \cdot a$, obtained by changing all non-zero coefficients of M to 1. Sometimes we confuse the multiset $[\![M]\!]$ with the corresponding subset $\{a \in A \mid M(a) > 0\}$ of A, and use on them the usual set operations and relations.

A *multirelation* $f : A \to B$ is a multiset of $A \times B$. It induces in an obvious way a function $\mu f : \mu A \to \mu B$, defined as

$$\mu f(\sum_{a \in A} n_a \cdot a) = \sum_{b \in B} \sum_{a \in A} (n_a \cdot f(a, b)) \cdot b.$$

If the multirelation f satisfies $f(a, b) \leq 1$ for all $a \in A$ and $b \in B$ then we sometimes confuse it with the corresponding set-relation and write $a\, f\, b$ or $f(a, b)$ for $f(a, b) = 1$. We call *inverse* of f the multirelation $f^{-1} : B \to A$, defined by $f^{-1}(b, a) = f(a, b)$ for all $a \in A$, $b \in B$.

A multirelation $f : A \to B$ is called *finitary* if for any a in A the set $\{b \in B \mid f(a, b) > 0\}$ is finite. In the following we will consider only finitary multirelations, and thus the qualification "finitary" will be often omitted.

The *composition* of two multirelations $f : A \to B$ and $g : B \to C$ is the multirelation $g \circ f : A \to C$ defined as

$$(g \circ f)(a, c) = \sum_{b \in B} f(a, b) \cdot g(b, c).$$

Observe that the above definition is well given only if the involved multirelations are finitary (since "infinite" coefficients are not allowed). The category having sets as objects and (finitary) multirelations as arrows is denoted by **MSet**.

We conclude by recalling some trivial properties of multisets which will be used in the sequel.

Proposition 2.1
Let $h : A \to B$ be a multirelation and let $M, M' \in \mu A$. Then

1. $\mu h(M + M') = \mu h(M) + \mu h(M')$;
2. *if $M \leq M'$ then $\mu h(M) \leq \mu h(M')$;*
3. $[\![\mu h(M)]\!] \leq \mu h([\![M]\!])$, *for all $M \in \mu A$.*

2.2 Contextual and inhibitor nets

In this section we review the definition of contextual and inhibitor nets, the two generalizations of ordinary Petri nets which are investigated in the FIRST PART. The adopted notions of concurrent enabling and step sequence (token game) are compared with other proposals in the literature.

2.2.1 Contextual nets

Contextual nets extend ordinary Petri nets with the possibility of handling contexts: in a contextual net transitions can have not only preconditions and postconditions, but also *context* conditions. A transition can fire if enough tokens are present in its preconditions and context conditions. In the firing, preconditions are consumed, context conditions remain *unchanged* and new tokens are generated in the postconditions. We next introduce *(marked) contextual P/T nets* [Ris94, Bus98] (or *c-nets* for short), that following the lines suggested in [MR95] for C/E systems, add contexts to P/T Petri nets.

DEFINITION 2.2 (C-NET)
A (marked) contextual Petri net (c-net) *is a tuple* $N = \langle S, T, F, C, m \rangle$*, where*

- S *is a set of* places*;*
- T *is a set of* transitions*;*
- $F = \langle F_{pre}, F_{post} \rangle$ *is a pair of multirelations, from* T *to* S*.*
- C *is a multirelation from* T *to* S*, called the* context relation*;*
- m *is a multiset of* S*, called the* initial marking*.*

We assume, without loss of generality, that $S \cap T = \emptyset$. Moreover, we require that for each transition $t \in T$, there exists at least a place $s \in S$ such that $F_{pre}(t, s) > 0$.[1] This is a common assumption when one is interested in having a causal semantics for nets. In fact given a transition with empty pre-set, an unbounded number of indistinguishable occurrences of such transition can be fired in parallel. Consequently, although a theory of deterministic processes could be still developed, very serious problems arise when trying to define an unfolding and an event structure semantics, where each event is intended to represent a uniquely determined occurrence of a transition.

[1] This is a weak version of the condition of *T-restrictedness* that requires also $F_{post}(t, s) > 0$, for some $s \in S$.

DEFINITION 2.3 (PRE-SET, POST-SET, AND CONTEXT)
Let N be a c-net. As usual, the functions from μT to μS induced by the multirelations F_{pre} and F_{post} are denoted by ${}^\bullet(\ )$ and $(\ )^\bullet$, respectively. If $A \in \mu T$ is a multiset of transitions, ${}^\bullet A$ is called its pre-set*, while $A^\bullet$ is called its* post-set*. Moreover, by $\underline{A}$ we denote the* context *of A, defined as $\underline{A} = \mu C(A)$.*

An analogous notation is used to denote the functions from S to $\mathbf{2}^T$ defined as, for $s \in S$, ${}^\bullet s = \{t \in T \mid F_{post}(t,s) > 0\}$, $s^\bullet = \{t \in T \mid F_{pre}(t,s) > 0\}$, $\underline{s} = \{t \in T \mid C(t,s) > 0\}$.

For a multiset of transitions A to be enabled by a marking M it is sufficient that M contains the pre-set of A and at least one *additional* token in each place in the context of A. This corresponds to the intuition that a token in a place can be used as context by many transitions at the same time and with multiplicity greater than one by the same transition.

DEFINITION 2.4 (TOKEN GAME)
Let N be a c-net and let M be a marking *of N, that is a multiset $M \in \mu S$. Given a finite multiset of transitions $A \in \mu T$, we say that A is* enabled *by M if ${}^\bullet A + [\![\underline{A}]\!] \leq M$. The* transition relation *between markings is defined as*

$$M\,[A\rangle\,M' \qquad \textit{iff} \qquad A \textit{ is enabled by } M \textit{ and } M' = M - {}^\bullet A + A^\bullet.$$

We call $M\,[A\rangle\,M'$ a *step*. A *simple step* (also called a *firing*) is a step involving just one transition, i.e., $M\,[t\rangle\,M'$.

DEFINITION 2.5 (REACHABLE MARKING)
Let N be a c-net. A marking M is called reachable *if there exists a finite* step sequence

$$m\,[A_0\rangle\,M_1\,[A_1\rangle\,M_2 \ldots\,[A_n\rangle\,M$$

starting from the initial marking and leading to M.

2.2.2 Inhibitor nets

Inhibitor nets (or nets with *inhibitor arcs*) further generalize contextual nets with the possibility of checking not only for the presence, but also for the *absence* of tokens in a place. Concretely, inhibitor nets are contextual nets enriched with a new relation that specifies, for each transition, which are the places that inhibit its firing.

DEFINITION 2.6 (I-NET)
A (marked) inhibitor Petri net (i-net) *is a tuple $N = \langle S, T, F, C, I, m\rangle$, where $\langle S, T, F, C, m\rangle$ is a contextual net and $I \subseteq T \times S$ is a relation, called the* inhibitor relation.

Let N be an i-net. The pre-set, post-set and context of transitions and places are defined as for c-nets. Furthermore, for a multiset of transitions $A \in \mu T$ we define the *inhibitor set* of A, denoted by ${}^{\circledcirc}A$, as the set of places which inhibit the transitions in A, namely ${}^{\circledcirc}A = I(\llbracket A \rrbracket)$. Similarly, for a place s we define ${}^{\circledcirc}s = \{t \in T \mid I(t,s)\}$.

The notion of enabling changes in order to take into account also the effect of the inhibitor arcs. As for c-nets, a finite multiset of transitions A is enabled by a marking M, if M contains the pre-set of A and covers the context of A. In addition no token must be present nor produced by the step in the places of the inhibitor set of A.

DEFINITION 2.7 (TOKEN GAME)
Let N be an i-net and let $M \in \mu S$ be a marking *of N. Given a finite multiset of transitions $A \in \mu T$, we say that A is* enabled *by M if ${}^\bullet A + \llbracket \underline{A} \rrbracket \leq M$ and $\llbracket M + A^\bullet \rrbracket \cap {}^{\circledcirc}A = \emptyset$. The* transition relation *between markings is defined as*

$$M\,[A\rangle\,M' \qquad \textit{iff} \qquad A \textit{ is enabled by } M \textit{ and } M' = M - {}^\bullet A + A^\bullet.$$

Steps, firings and reachable markings are defined in the obvious way.

2.2.3 Alternative approaches

In the literature there is not a complete agreement on the notion of enabling for transitions and steps of contextual and inhibitor nets. The alternative proposals, for instance in [Vog97b] and in [JK95], allow for the execution of steps where the same token is used both as context and as precondition, or dually of steps where a token is generated in the inhibitor set.

The basic assumption which motivates our choice is that concurrent transitions should be allowed to fire also in any order. In other words we require that given two multisets of transitions $A_1, A_2 \in \mu T$, then

$$M\,[A_1 + A_2\rangle\,M' \qquad \Rightarrow \qquad M\,[A_1\rangle\,M''\,[A_2\rangle\,M' \text{ for some } M''.$$

Consequently, any marking reachable via step sequences is also reachable by firing sequences where each step consists of a single transition.

Without getting into technical details, let us compare the different approaches by means of simple examples. Consider the c-nets N_0 and N_1 in Figure 2.1. According to our definition, the step consisting of the concurrent execution of t_0 and t_1 is not legal for both nets.

The definition of [Vog97b] allows for the concurrent execution of t_0 and t_1 in N_0 but not in N_1, the idea being that the firings are not instantaneous and two actions should be considered concurrent if their executions can overlap in time. This is the case for t_0 and t_1 in N_0, where the only constraint is that the firing of t_0 must start after the firing of t_1 has begun. Instead, in N_1 when the firing of a transition starts the other transition is no more executable. The property that *any* firing sequence

Figure 2.1: Different notions of enabling in the literature.

serializing a concurrent step must be admissible is weakened to the *existence* of a possible serialization. Consequently it is still true that the markings reachable via step sequences coincide with the markings reachable via firing sequences. According to [JK95], instead, the concurrent execution of t_0 and t_1 is permitted also in N_1. Observe that the firing of one of t_0 or t_1 inhibits the other transition. Hence also the weaker property of coincidence of the markings reachable via step and firing sequences is lost.

As already pointed out in [MR95, JK95] the more liberal definitions of enabling are suited to deal with timed systems, where action have non-zero durations, while our approach is more appropriate when the firings are considered as instantaneous.

We conclude by observing that minor differences exist also with respect to [Bus98], where a single transition is allowed to produce a token in its inhibitor set and the context is a set (rather than a multiset). As for the first point, since in both approaches a transition cannot use the same token as precondition and context, we judge more coherent to forbid the firing also in the dual situation. Moreover, in this way a multiset of transitions can be safely thought of as a single "composed" transition, making the treatment more uniform. Instead, the choice of viewing contexts as multisets, allowing the firing when at least one token is present in each context place, is taken by analogy with graph grammars: a graph production may specify a context with multiple occurrences of the same resource, but it can be applied with a match which is non-injective on the context. Anyway both choices are mainly a matter of taste, and they have little influence on the developed theory

2.3 Process semantics of generalized Petri nets

The problem of providing a (deterministic) process semantics for contextual and inhibitor nets has been faced by various authors [Vog97b, Ris94, GM98, Win98, Bus98, JK95]. This section is aimed at giving some pointers to the related literature and an overview of the approaches in [Ris94, GM98, Win98, Bus98] which will be "rediscovered" in the next chapters as special cases of the developed theory. Rather

than giving a formal introduction, the section is intended to explain the framework in which our results have been developed.

2.3.1 Contextual nets

A theory of non-sequential processes for contextual nets has been first proposed in [MR95], where two kind of processes for C/E contextual nets are defined. A *c-process* consists of an ordinary occurrence net with a suitable mapping to the given contextual net. A simpler notion, called *cc-process* is obtained by allowing for the presence of contexts in the occurrence net underlying the process. Contextual P/T nets and a corresponding notion of deterministic process can be found in the PhD thesis [Ris94] and in [GM98, Win98].

The *causal dependency* relation for a contextual net N is obtained as the transitive and reflexive closure of the relation $\prec_N$, defined as $t \prec_N t'$ iff

$$t^\bullet \cap ({}^\bullet t' \cup \underline{t'}) \neq \emptyset \quad \vee \quad \underline{t} \cap {}^\bullet t' \neq \emptyset. \qquad (\dagger)$$

A (deterministic) *occurrence contextual net* is defined as a c-net O where $\prec_N$ is irreflexive, the causal dependency relation is a partial order and each place is in the pre- and post-set of at most one transition. Then, a (*deterministic*) *process* of a c-net N is a deterministic occurrence c-net O, with a total mapping $\varphi : O \to N$ preserving the pre-set, post-set and the context of transitions.

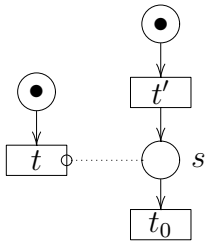
Observe that since an occurrence net is intended to represent a deterministic computation, the problem described in the INTRODUCTION, related to the presence of asymmetric conflicts in c-nets does not arise here. Simply, as expressed by (†) above, if $\underline{t} \cap {}^\bullet t' \neq \emptyset$ then transition t *must* precede t' in the computation represented by the net, exactly as it happens for causality, and thus the kind of dependency between t and t' can be safely assimilated to causality.

The papers [GM98, Win98] extend the theory of concatenable processes of ordinary nets [DMM89] to c-nets, by showing that the concatenable processes of a c-net N form the arrows of a symmetric monoidal category $\mathbf{CP}[N]$, where the objects are the elements of the free commutative monoid over the set of places (multisets of places). In particular, in [GM98] a purely algebraic characterization of such category is given.

A different approach to the process semantics of contextual nets is considered in [Vog97b]. As explained before, there the author assumes a different notion of enabling, allowing for the concurrent firing of two transitions also if one consumes a token read by the other. Therefore "syntactically" the processes of [Vog97b] coincide with those in [MR95], but they are intended to represent different step sequences. The paper argues that a partial order is not sufficient to express the dependencies between transitions occurrences even inside a deterministic computation. Thus a process is associated with a relational structure consisting of two relations modelling causality and start precedence (*spc-structure*) and it is shown that they allow to recover exactly the concurrent computations represented by the process.

2.3.2 Inhibitor nets

The deterministic processes of [MR95, Ris94] are generalized to nets with read and inhibitor arcs in [BP96, Bus98, BP99]. An occurrence i-net is naturally defined as an acyclic i-net where all transitions may fire. The main problem in this case is that the informations contained in an occurrence i-net may not be sufficient to single out a unique deterministic computation. Consider the simple net below.



As already discussed, there are two possible different executions of the net and they cannot be reasonably identified from the point of view of causality. To get rid of this ambiguity the occurrence i-net underlying a process is *enriched* by partitioning the inhibitor arcs in two disjoint sets: the "before" arcs I_b and the "after" arcs I_a. Intuitively, if $(t, s) \in I_b$ is a "before" arc then t must be executed before the place s is filled, while if $(t, s) \in I_a$ is an "after" arc then t must be executed after the place s has been emptied.

Processes of i-nets are shown to be appropriate, in the sense that they uniquely determine the causal dependencies among their events, independent transitions can fire concurrently and concurrent sets of places correspond to reachable markings.

A notion of process for elementary net systems (a kind of C/E nets) extended with inhibitor arcs is proposed also in [JK95, JK93, JK91]. This work relies on the more liberal notion of enabling discussed in the previous section, which permits the concurrent firing of non serializable transitions. As it happens for c-nets in [Vog97b], this choice makes partial orders insufficient to express the dependencies between events also in a single deterministic computation. To overcome the problem the authors introduce *stratified order structures*, where two different relations are employed to represent causality and weak causality, and weak dependent events may also happen simultaneously.

2.4 Event structures and domains

Event structures [NPW81, Win87a] are a widely used model of concurrent computations which arose from the attempt of developing a theory of concurrency incorporating both the insights of C.A. Petri and D.S. Scott. In this section we introduce the fundamental notions of prime event structure and of prime algebraic domain, and we review the close relationship between these mathematical models explaining how they can be seen as equivalent presentations of the same fundamental idea.

Finally we review some generalizations of prime event structures which have been proposed in the literature to overcome some limitations of expressiveness of prime event structures.

2.4.1 Prime event structures

Prime event structures (PES) [NPW81] are a simple event-based model of concurrent computations in which events are considered as atomic and instantaneous steps, which can appear only once in a computation. An event can occur only after some other events (its causes) have taken place and the execution of an event can inhibit the execution of other events. This is formalized via two binary relations: *causality*, modelled by a partial order relation $\leq$ and *conflict*, modelled by a symmetric and irreflexive relation $\#$, hereditary with respect to causality.

DEFINITION 2.8 (PRIME EVENT STRUCTURES)
A prime event structure (PES) *is a tuple* $P = \langle E, \leq, \# \rangle$*, where* E *is a set of* events *and* $\leq$*,* $\#$ *are binary relations on* E *called* causality relation *and* conflict relation *respectively, such that:*

1. *the relation* $\leq$ *is a partial order and* $\lfloor e \rfloor = \{e' \in E : e' \leq e\}$ *is finite for all* $e \in E$*;*

2. *the relation* $\#$ *is irreflexive, symmetric and hereditary with respect to* $\leq$*, i.e.,* $e \# e' \leq e''$ *implies* $e \# e''$ *for all* $e, e', e'' \in E$*;*

A configuration of a PES is a set of events representing a possible computation of the system modelled by the event structure.

DEFINITION 2.9 (CONFIGURATION)
A configuration *of a* PES $P = \langle E, \leq, \# \rangle$ *is a subset of events* $C \subseteq E$ *such that for all* $e, e' \in C$

1. $\neg(e \# e')$ (conflict-freeness)

2. $\lfloor e \rfloor \subseteq C$ (left-closedness)

Given two configurations $C_1 \subseteq C_2$ if $e_0, \ldots, e_n$ is any linearization of the events in $C_2 - C_1$, compatible with causality, then

$$C_1 \subseteq C_1 \cup \{e_0\} \subseteq C_1 \cup \{e_0, e_1\} \subseteq \ldots \subseteq C_2$$

is a sequence of well-defined configurations. Therefore subset inclusion can be safely thought of as a computational ordering on configurations.

DEFINITION 2.10 (POSET OF CONFIGURATIONS)
We denote by $Conf(P)$ *the set of configurations of a prime event structure* P*, ordered by subset inclusion.*

The class of PES is turned into a category by introducing a notion of morphism.

DEFINITION 2.11 (CATEGORY **PES**)
Let $P_0 = \langle E_0, \leq_0, \#_0 \rangle$ *and* $P_1 = \langle E_1, \leq_1, \#_1 \rangle$ *be two* PES*'s. A* PES-morphism $f : P_0 \to P_1$ *is a partial function* $f : E_0 \to E_1$ *such that:*

1. *for all* $e_0 \in E_0$*, if* $f(e_0) \neq \bot$ *then* $\lfloor f(e_0) \rfloor \subseteq f(\lfloor e_0 \rfloor)$*;*
2. *for all* $e_0, e_0' \in E_0$*, if* $f(e_0) \neq \bot \neq f(e_0')$ *then*
   (a) $(f(e_0) = f(e_0')) \wedge (e_0 \neq e_0') \quad \Rightarrow \quad e_0 \#_0 e_0'$*;*
   (b) $f(e_0) \#_1 f(e_0') \quad \Rightarrow \quad e_0 \#_0 e_0'$*;*

The category of prime event structures and PES*-morphisms is denoted by* **PES**.

It is possible to verify that PES morphisms "preserve computations", in the sense that the image through a PES morphism of a configuration is a configuration. Therefore a PES morphism naturally induces a monotone mapping between the corresponding posets of configurations.

2.4.2 Prime algebraic domains

This subsection reviews the definition of the category **Dom** of finitary prime algebraic domains as introduced in [Win87a]. The intuition behind their computational interpretation helps in understanding the close relationship existing between domains and event structures, which can be formalized, at categorical level, as an equivalence of categories.

First we need some basic notions and notations for partial orders. A preordered or partially ordered set $\langle D, \sqsubseteq \rangle$ will be often denoted simply as D, by omitting the (pre)order relation. Given an element $x \in D$, we write $\downarrow x$ to denote the set $\{y \in D \mid y \sqsubseteq x\}$. A subset $X \subseteq D$ is *compatible*, written $\uparrow X$, if there exists an upper bound $d \in D$ for X (i.e., $x \sqsubseteq d$ for all $x \in X$). It is *pairwise compatible* if $\uparrow \{x, y\}$ (often written $x \uparrow y$) for all $x, y \in X$. A subset $X \subseteq D$ is called *directed* if for any $x, y \in X$ there exists $z \in X$ such that $x \sqsubseteq z$ and $y \sqsubseteq z$.

DEFINITION 2.12 ((FINITARY) (ALGEBRAIC) COMPLETE PARTIAL ORDER)
A partial order D *is* (directed) complete (CPO) *if for any directed subset* $X \subseteq D$ *there exists the least upper bound* $\bigsqcup X$ *in* D*. An element* $e \in D$ *is* compact *if for any directed set* $X \subseteq D$*,* $e \sqsubseteq \bigsqcup X$ *implies* $e \sqsubseteq x$ *for some* $x \in X$*. The set of* compact *elements of* D *is denoted by* $\mathsf{K}(D)$*.*

A CPO D *is called* algebraic *if for any* $x \in D$*,* $x = \bigsqcup(\downarrow x \cap \mathsf{K}(D))$*. We say that* D *is* finitary *if for each compact element* $e \in D$ *the set* $\downarrow e$ *is finite.*

Given a finitary algebraic CPO D we can think of its elements as "pieces of information" expressing the states of evolution of a process. Finite elements represent states

which are reached after a finite number of steps. Thus algebraicity essentially says that each infinite computation can be approximated with arbitrary precision by the finite ones.

Winskel's domains satisfy stronger completeness properties, which are formalized by the following definition.

DEFINITION 2.13 ((PRIME ALGEBRAIC) COHERENT POSET)
A partial order D is called coherent *(*pairwise complete*) if for all pairwise compatible $X \subseteq D$, there exists the least upper bound $\bigsqcup X$ of X in D.*

A complete prime *of D is an element $p \in D$ such that, for any compatible $X \subseteq D$, if $p \sqsubseteq \bigsqcup X$ then $p \sqsubseteq x$ for some $x \in X$. The set of complete primes of D is denoted by $Pr(D)$. The partial order D is called* prime algebraic *if for any element $d \in D$ we have $d = (\bigsqcup \downarrow d \cap Pr(D))$. The set $\downarrow d \cap Pr(D)$ of complete primes of D below d will be denoted $Pr(d)$.*

Being not expressible as the least upper bound of other elements, the complete primes of D can be seen as elementary indivisible pieces of information (events). Thus prime algebraicity expresses the fact that all the possible computations of the system at hand can be obtained by composing these elementary blocks of information.

Notice that directed sets are pairwise compatible, and thus each coherent partial order is a CPO. For the same reason each complete prime is a compact element, namely $Pr(D) \subseteq \mathsf{K}(D)$ and thus prime algebraicity implies algebraicity. Moreover if D is coherent then for each non empty $X \subseteq D$ there exists the greatest lower bound $\bigsqcap X$, which can be expressed as $\bigsqcup\{y \in D \mid \forall x \in X.\ y \sqsubseteq x\}$.

DEFINITION 2.14 (DOMAINS)
The partial orders we shall work with are coherent, prime algebraic, finitary partial orders, hereinafter simply referred to as (Winskel's) domains.[2]

The definition of morphism between domains is based on the notion of immediate precedence. Given a domain D and two distinct elements $d \neq d' \in D$ we say that d is an *immediate predecessor* of d', written $d \prec d'$ if

$$d \sqsubseteq d' \ \wedge \forall d'' \in D.\ (d \sqsubseteq d'' \sqsubseteq d' \ \Rightarrow \ d'' = d \ \vee \ d'' = d')$$

Moreover we write $d \preceq d'$ if $d \prec d'$ or $d = d'$. According to the informal interpretation of domain elements sketched above, $d \preceq d'$ intuitively means that d' is obtained from d by adding a quantum of information. Domain morphisms are required to preserve such relation.

[2]The use of this kind of structures in semantics have been first investigated by Berry [Ber78], where they are called *dI-domains*. The relation between Winskel domains and dI-domains, which are finitary distributive consistent-complete algebraic CPO's is established by the fact that for a finitary algebraic consistent-complete (or coherent) CPO, prime algebraicity is equivalent to distributivity.

DEFINITION 2.15 (CATEGORY **Dom**)
Let D_0 and D_1 be domains. A domain morphism *$f : D_0 \to D_1$ is a function, such that:*

- $\forall x, y \in D_0$, *if* $x \preceq y$ *then* $f(x) \preceq f(y)$. ($\preceq$-preserving)
- $\forall X \subseteq D_0$, X *pairwise compatible,* $f(\bigsqcup X) = \bigsqcup f(X)$; (Additive)
- $\forall X \subseteq D_0$, $X \neq \emptyset$ *and compatible,* $f(\bigsqcap X) = \bigsqcap f(X)$; (Stable)

We denote by **Dom** *the category having domains as objects and domain morphisms as arrows.*

Relating prime event structures and domains

Both event structures and domains can be seen as models of systems where computations are built out from atomic pieces. Formalizing this intuition, in [Win87a] the category **Dom** is shown to be equivalent to the category **PES**, the equivalence being established by two functors $\mathscr{L} : \mathbf{PES} \to \mathbf{Dom}$ and $\mathscr{P} : \mathbf{Dom} \to \mathbf{PES}$

$$\mathbf{PES} \underset{\mathscr{L}}{\overset{\mathscr{P}}{\underset{\sim}{\rightleftarrows}}} \mathbf{Dom}$$

The functor $\mathscr{L}$ associates to each PES the poset $Conf(P)$ of its configurations which can be shown to be a domain. The image via $\mathscr{L}$ of a PES-morphism $f : P_0 \to P_1$ is the obvious extension of f to sets of events.

The definition of the functor $\mathscr{P}$, mapping domains back to PES's requires the introduction of the notion of prime interval.

DEFINITION 2.16 (PRIME INTERVAL)
Let $\langle D, \sqsubseteq \rangle$ be a domain. A prime interval *is a pair $[d, d']$ of elements of D such that $d \prec d'$. Let us define*

$$[c, c'] \leq [d, d'] \quad \text{if } (c = c' \sqcap d) \ \wedge \ (c' \sqcup d = d'),$$

and let $\sim$ be the equivalence obtained as the transitive and symmetric closure of (the preorder) $\leq$.

The intuition that a prime interval represents a pair of elements differing only for a "quantum" of information is confirmed by the fact that there exists a bijective correspondence between $\sim$-classes of prime intervals and complete primes of a domain D (see [NPW81]). More precisely, the map

$$[d, d']_{\sim} \mapsto p,$$

where p is the unique element in $Pr(d') - Pr(d)$, is an isomorphism between the $\sim$-classes of prime intervals of D and the complete primes $Pr(D)$ of D, whose inverse is the function:

$$p \mapsto [\bigsqcup\{c \in D \mid c \sqsubset p\}, p]_{\sim}.$$

The above machinery allows us to give the definition of the functor $\mathscr{P}$ "extracting" an event structure from a domain.

DEFINITION 2.17 (FROM DOMAINS TO PES'S)
The functor $\mathscr{P} : \mathbf{Dom} \to \mathbf{PES}$ *is defined as follows:*

- *given a domain* D, $\mathscr{P}(D) = \langle Pr(D), \leq, \# \rangle$ *where*

$$p \leq p' \quad \textit{iff} \quad p \sqsubseteq p' \qquad \textit{and} \qquad p \# p' \quad \textit{iff} \quad \neg(p \uparrow p');$$

- *given a domain morphism* $f : D_0 \to D_1$, *the morphism* $\mathscr{P}(f) : \mathscr{P}(D_0) \to \mathscr{P}(D_1)$ *is the function:*

$$\mathscr{P}(f)(p_0) = \begin{cases} p_1 & \text{if } p_0 \mapsto [d_0, d_0']_{\sim},\ f(d_0) \prec f(d_0') \\ & \text{and } [f(d_0), f(d_0')]_{\sim} \mapsto p_1; \\ \bot & \text{otherwise, i.e., if } f(d_0) = f(d_0'). \end{cases}$$

2.4.3 Generalized event structure models

To represent in a direct way the behaviour of languages or models of computations where the dependencies between events are not adequately described in terms of causality and conflict, several extensions of prime event structures have been considered in the literature. Some of these extensions can be seen as special subclasses of the more general Winskel's event structures, while others represents orthogonal generalizations of Winskel's model.

Flow and bundle event structures

In a prime event structure, and more generally in a stable event structure, each event has a uniquely determined history, namely given an event we can always identify a unique subset of events which are necessary for the execution of e, the set of its *causes*.

While in certain situations the mentioned property is appreciable since it gives to the events a very simple operational meaning, in other cases it may be an undesirable restriction. Consider for instance a process algebra with nondeterministic choice "+"and sequential composition ";" operators. To give a PES semantics of a term $(a + b); c$ we are forced to use two different events to represent the execution of c, one for the execution of c after a and the other for the execution of c after b.

To model nondeterministic choices, or equivalently the possibility of having multiple disjunctive and mutually exclusive causes for an event, Boudol and Castellani [BC88] introduce the notion of *flow event structure*, where the causality relation is replaced by an irreflexive (in general non transitive) *flow relation*, representing essentially immediate causal dependency, and conflict is no more hereditary.

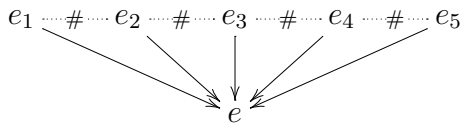


Figure 2.2: A flow event structure F such that there are no bundle event structures with the same configurations.

Definition 2.18 (flow event structure)
A flow event structure *is a triple* $\langle E, \prec, \# \rangle$*, where* E *is a denumerable set of events,* $\prec \subseteq E \times E$ *is an irreflexive relation called the* flow relation*, and* $\# \subseteq E \times E$ *is the* symmetric conflict *relation.*

A configuration is then defined as a conflict free set of events, where the flow relation is acyclic, and given any event e in the configuration, if $e' \prec e$ then the configuration must contain e' or another event e'' in conflict with e' such that $e'' \prec e$ (thus one of the disjunctive causes must be present in the configuration).

To face a similar problem in the semantics of LOTOS, Langerak [Lan92a, Lan92b] defines *bundle event structures*, where a set of multiple disjunctive and mutually exclusive causes for an event is called a *bundle set* for the event, and comes into play as a primitive notion.

Definition 2.19 (bundle event structure)
A bundle event structure *is a triple* $\langle E, \mapsto, \# \rangle$*, where* E *is the denumerable set of events,* $\# \subseteq E \times E$ *is the (irreflexive)* symmetric conflict *relation and* $\mapsto \subseteq \mathbf{2}^E_{fin} \times E$ *is the* bundle relation*. Distinct events in the same bundle are required to be in conflict.*

The explicit representation of the bundles makes bundle event structures strictly less expressive than flow event structures. For instance, consider the flow event structure F in Figure 2.2, where conflict is represented by dotted lines labelled by #, while the flow relation is represented by arrows. The configurations of F are $\{e_1, e_3, e_5\}$, $\{e_1, e_4\}$ and $\{e_2, e_4\}$. Observing that the only possible bundles are pairs of conflictual events it is not difficult to see that there are no bundle event structures having the same set of configurations (see [Lan92b] for a wider discussion). On the other hand, bundle event structures offer the advantage of having a simpler theory. For instance, differently from what happens for flow event structures, non-executable events can be removed without affecting the behaviour of the event structure.

Event automata and asymmetric conflicts

The papers [PP92, PP95] introduce the "operational" notion of *event automaton*, which can be seen as a generalization of the set of configurations of an event structure. An event automaton is a triple $\mathscr{E} = \langle E, St, \rightarrowtriangle \rangle$, where E is a set of events,

$St \subseteq \mathbf{2}^E_{fin}$ is a set of states and $\rightarrowtriangle \subseteq St \times St$ is a relation such that if $X \rightarrowtriangle Y$ then the state Y is obtained from X by adjoining a single event.

A language is introduced which allows one to specify in logical terms the behaviour of an event automata, and an event structure can then be thought of as a specification expressed in the language. In particular, including negation in the language and allowing for prescriptions of the kind "$\neg e_1 \vdash e_0$", meaning that the event e_0 is enabled in a state S if $e_1 \notin S$, one is naturally lead to the notion of possible event and asymmetric conflict. In fact, if "$\neg e_1 \vdash e_0$" holds then if both events are in the same state, necessarily e_0 must have been executed first. Hence, as already discussed, we can think that e_0 is a possible cause of e_1, or that an asymmetric conflict exists between the two events.

Inspired by these considerations, the paper [PP95] extends prime and flow event structures with *possible events/flow*. A *prime event structure with possible events* is a tuple $\langle E, E_p, \leq, \# \rangle$, where $\langle E, \leq, \# \rangle$ is a PES and $E_p \subseteq E$ is a set of *possible events*. A configuration is then required to contain all the causes of an event in $E - E_p$, but some possible causes may be absent.

Similarly, a *flow event structure with possible flow* is a flow event structure enriched with a new relation $<_p$, called the *possible flow relation*. The flow and possible flow relations must be acyclic on a configuration, but the causal closure is requested only with respect to the flow relation. Again the precedences imposed by the possible flow relation can be thought of as possible precedences.

To conclude, it is worth remarking that similar ideas are developed, under a different perspective by Degano, Vigna and Gorrieri, in [DGV93, Bod98], where *prioritized event structures* are introduced as PES enriched with a partial order relation modelling priorities between events. Furthermore also bundle event structures have been extended by Langerak in [Lan92b] to take into account asymmetric conflicts.

Chapter 3

Semantics of Contextual Nets

This chapter presents an event structure semantics for *contextual nets*, an extension of P/T Petri nets where transitions can check for the presence of tokens without consuming them (read-only operations). A basic role is played by *asymmetric event structures*, a generalization of Winskel's prime event structures where symmetric conflict is replaced by a relation modelling *asymmetric conflict* or *weak causality*, used to represent the new kind of dependency between events arising in contextual nets. Extending Winskel's seminal work on safe nets, the truly concurrent event based semantics of contextual nets is given at categorical level via a chain of coreflections leading from the category **SW-CN** of semi-weighted contextual nets to the category **Dom** of finitary prime algebraic domains:

$$\mathbf{SW\text{-}CN} \underset{\mathscr{U}_a}{\overset{\mathscr{I}_O}{\underset{\perp}{\hookleftarrow}}} \mathbf{O\text{-}CN} \underset{\mathscr{E}_a}{\overset{\mathscr{N}_a}{\underset{\perp}{\leftarrow}}} \mathbf{AES} \underset{\mathscr{L}_a}{\overset{\mathscr{P}_a}{\underset{\perp}{\leftarrow}}} \mathbf{Dom} \underset{\mathscr{P}}{\overset{\mathscr{L}}{\underset{\sim}{\leftarrow}}} \mathbf{PES}$$

First an unfolding construction generates from a contextual net N a corresponding *occurrence contextual net* $\mathscr{U}_a(N)$. The unfolding describes the behaviour of N in a static way by making explicit the possible events in the computations of the net and the dependency relations between them. The construction can be extended to a functor from **SW-CN** to the category **O-CN** of occurrence contextual nets, that is right adjoint to the inclusion functor. The transitions of an occurrence contextual net are related by causal dependency and asymmetric conflict, and thus, the semantics of semi-weighted contextual nets given in terms of occurrence contextual nets can be naturally abstracted to an AES semantics: given an occurrence contextual net we obtain an AES simply forgetting the places, but remembering the dependency relations that they induce between transitions. Again, this transformation is expressed, at categorical level, as a coreflection between the category **AES** of asymmetric event structures and **O-CN**. Finally, the configurations of the asymmetric event structure, endowed with a suitable order, are shown to form a finitary prime algebraic domain. This last step generalizes Winskel's equivalence between **PES** and **Dom** to the existence of a coreflection between **AES** and **Dom**. Such a coreflection

allows for an elegant translation of the AES semantics into a domain, and thus (via Winskel's equivalence) into a traditional PES semantics.

We also investigate the relation between the proposed unfolding semantics and several deterministic process semantics for contextual nets in the literature. First we show that the notion of *deterministic process* naturally arising from our theory coincides with other common proposals in the literature. Then a tight relationship is established between the unfolding and the deterministic process semantics, by showing that the domain obtained via the unfolding can be characterized as the collection of the deterministic processes of the net endowed with a kind of prefix ordering.

The rest of the chapter is organized as follows. Section 3.1 introduces the category **AES** of asymmetric event structures and describes some properties of such structures. Section 3.2 defines the coreflection between **AES** and the category **Dom** of finitary prime algebraic domains. Section 3.3 presents the category of contextual nets and focuses on the subcategory **SW-CN** of (semi-weighted) contextual nets which we shall work with. Section 3.4 is devoted to the definition and analysis of the category **O-CN** of occurrence contextual nets. Section 3.5 describes the unfolding construction for semi-weighted contextual nets and shows how such a construction gives rise, at categorical level, to a coreflection between **SW-CN** and **O-CN**. Section 3.6 completes the chain of coreflections from **SW-CN** to **Dom**, by presenting a coreflection between **O-CN** and **AES**. Section 3.7 discusses the relation between the unfolding and the deterministic process semantics of contextual nets.

3.1 Asymmetric conflicts and asymmetric event structures

We stressed in the INTRODUCTION that PES's (and in general Winskel's event structures) are not expressive enough to model in a direct way the behaviour of models of computation, such as string, term, graph rewriting and contextual nets, where a rule may preserve a part of the state, in the sense that part of the state is necessary for the application of the rule, but it is not affected by such application.

The critical situation when dealing with contextual nets is represented by the net in Figure 3.1.(a), where the firing of t_1 inhibits t_0, but not vice versa. The dependency between the two transitions can be seen either as a kind of asymmetric conflict or as a weak form of causality, and cannot be modelled directly via a PES.

A reasonable way to encode this situation in a PES is to represent the firing of t_0 with an event e_0 and the firing of t_1 with two distinct mutually exclusive events: e_1', representing the execution of t_1 that prevents t_0, thus mutually exclusive with e_0; and e_1'', representing the execution of t_1 after t_0 (thus caused by e_0). Such PES is depicted in Figure 3.1.(b), where causal dependency is represented by a plain arrow and conflict is represented by a dotted line, labelled by #. However, this solution is

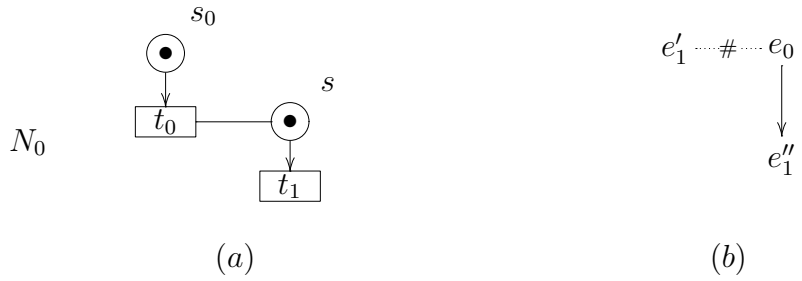


Figure 3.1: A simple contextual net and a prime event structure representing its behaviour.

not completely satisfactory with respect to the interpretation of contexts as "read-only resources": since t_0 just reads the token in s without changing it, one would expect the firing of t_1, preceded or not by t_0, to be represented by a unique event. This encoding may lead to an explosion of the size of the PES, since whenever an event is "duplicated" also all its consequences must be duplicated. In addition it should be noted that the information on the new kind of dependency determined by read-only operations is completely lost because it is "confused" with causality or symmetric conflict.

It is worth noting that the inability of representing the asymmetric conflict between events without resorting to duplications is not specific to prime event structures, but it is basically related to the axiom of general Winskel's event structures (see [Win87a], Definition 1.1.1) stating that the enabling relation $\vdash$ is "monotone" with respect to set inclusion:

$$A \vdash e \ \wedge \ A \subseteq B \ \wedge \ B \ \textit{consistent} \quad \Rightarrow \quad B \vdash e.$$

A consequence of this axiom is that the computational order between configurations is set inclusion, the idea being that if A and B are finite configurations such as $A \subseteq B$, then starting from A we can reach B by performing the events in $B - A$, whenever they become enabled. Obviously, this axiom does not hold in the presence of asymmetric conflict.

As mentioned in the previous chapter (Section 2.4), the problem of representing asymmetric conflicts in event structures has been already faced in the literature [PP92, Lan92b]. However none of the proposed models is suited for our aims. On the one hand, PES's with possible events [PP92]are not sufficiently expressive since they resort to a "global" notion of possible event. For example, the net of Figure 3.2 cannot be modelled by a PES with possible events since transition t_0 should be a "possible" cause for t_1, but a "strong" cause for t_2. On the other hand flow event structures with possible flow and bundle event structures with asymmetric conflict are expressive enough, but unnecessarily complicate for our aims due to their possibility of expressing multiple disjunctive causes. Technically, as it will become clear

in the next chapter, this greater generality would have prevented us from realizing the step from occurrence c-nets to event structures as a coreflection. Furthermore no categorical treatment of such models was available, and their relation with domains and PES's was not fully investigated.

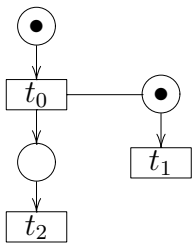


Figure 3.2: A contextual net for which PES's with possible events are not adequate.

3.1.1 Asymmetric event structures

In order to provide a more direct, event based representation of the behaviour of contextual nets this section introduces a new kind of event structure, called *asymmetric event structure* (AES). An AES, besides of the usual causal relation $\leq$ of a prime event structure, has a relation $\nearrow$ that allows us to specify the new kind of dependency described above for transitions t_0 and t_1 of the net in Figure 3.1 simply as $t_0 \nearrow t_1$. As already remarked, the same relation has two natural interpretations: it can be thought of either as an asymmetric version of conflict or as a weak form of causality. We decided to call it *asymmetric conflict*, but the reader should keep in mind both views, since in some situations it will be preferable to refer to the *weak causality* interpretation. Informally, in an AES each event has a set of "strong" causes (given by the causal dependency relation) and a set of weak causes (due to the presence of the asymmetric conflict relation). To be fired, each event must be preceded by all strong causes and by a (suitable) subset of the weak causes. Therefore, differently from PES's, the firing of an event can have more than one history. However observe that AES's still satisfy a property of stability, since a least history always exists, coinciding with the set of strong causes.

It comes of no surprise that in this setting the symmetric binary conflict is not anymore needed as a primitive relation, but it can be represented via "cycles" of asymmetric conflict. For instance, if $e \nearrow e'$ and $e' \nearrow e$ then clearly e and e' can never occur in the same computation. As a consequence, PES's can be identified with a special subclass of asymmetric event structures, namely those where all conflicts are actually symmetric.

The basic ideas for the treatment of asymmetric conflict in our approach are similar to those suggested by Pinna and Poigné in [PP92, PP95]. Apart from a different presentation, asymmetric event structures can be seen as a generalization of PES with possible events. Using their terminology, when $e_0 \nearrow e_1$ we can say that e_0 is a possible cause of e_1. However, differently from what happens for event structures with possible events, where a distinct set of possible events is singled out, our notion of possible cause is local, being induced by the asymmetric conflict relation. The extended bundle event structures of Langerak [Lan92b] share with our approach also the intuition that when asymmetric conflict is available, the symmetric conflict becomes useless.

For technical reasons we first introduce pre-asymmetric event structures. Then asymmetric event structures will be defined as special pre-asymmetric event structures satisfying a suitable condition of "saturation".

Recall from Section 2.1 that given a relation $r \subseteq X \times X$ and a subset $Y \subseteq X$, we denote by r_Y the restriction of r to Y, namely $r \cap (Y \times Y)$.

DEFINITION 3.1 (PRE-ASYMMETRIC EVENT STRUCTURE)
A pre-asymmetric event structure (pre-AES) *is a tuple* $G = \langle E, \leq, \nearrow \rangle$*, where* E *is a set of* events *and* $\leq$, $\nearrow$ *are binary relations on* E *called* causality relation *and* asymmetric conflict *respectively, such that:*

1. *the relation* $\leq$ *is a partial order and* $\lfloor e \rfloor = \{e' \in E \mid e' \leq e\}$ *is finite for all* $e \in E$*;*

2. *the relation* $\nearrow$ *satisfies, for all* $e, e' \in E$*:*

   (a) $e < e' \quad \Rightarrow \quad e \nearrow e'$;[1]

   (b) $\nearrow_{\lfloor e \rfloor}$ *is acyclic.*[2]

If $e \nearrow e'$*, according to the double interpretation of* $\nearrow$*, we say that* e is prevented *by* e' *or* e weakly causes e'*. Moreover we say that* e is strictly prevented *by* e' *(or that* e strictly weakly causes e'*), written* $e \rightsquigarrow e'$*, if* $e \nearrow e'$ *and* $\neg(e < e')$*.*

The definition can be easily understood by giving a more precise account of the ideas presented in the INTRODUCTION. Let $Fired_C(e)$ denote the fact that the event e has been fired in a computation C, later formalized by the notion of configuration, and let $prec_C(e, e')$ denote that e precedes e' in the computation. Then

$$e < e' \qquad \text{means that} \qquad \forall C.\ Fired_C(e') \ \Rightarrow\ Fired_C(e) \ \wedge\ prec_C(e, e')$$
$$e \nearrow e' \qquad \text{means that} \qquad \forall C.\ Fired_C(e) \wedge Fired_C(e') \ \Rightarrow\ prec_C(e, e').$$

[1] With $e < e'$ we mean $e \leq e$ and $e \neq e'$.

[2] Equivalently, we can require $(\nearrow_{\lfloor e \rfloor})^+$ irreflexive. This implies, in particular, that the relation $\nearrow$ is irreflexive.

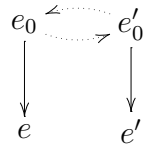


Figure 3.3: A pre-AES with two conflictual events e and e', not related by asymmetric conflict.

Therefore $<$ represents a global order of execution, while $\nearrow$ determines an order of execution only locally to each computation. Thus it is natural to impose $\nearrow$ to be an extension of $<$. Moreover notice that if a set of events form a cycle of asymmetric conflict then such events cannot appear in the same computation, otherwise the execution of each event should precede the execution of the event itself. This explains why we require the transitive closure of $\nearrow$, restricted to the causes $\lfloor e \rfloor$ of an event e, to be acyclic (and thus well-founded, being $\lfloor e \rfloor$ finite). Otherwise not all causes of e could be executed in the same computation and thus e itself could not be executed. The informal interpretation makes also clear that $\nearrow$ is *not* in general transitive. If $e \nearrow e' \nearrow e''$ it is not true that e must precede e'' when both fire. This holds only in a computation where also e' fires.

The fact that a set of events in a cycle of asymmetric conflict can never occur in the same computation can be naturally interpreted as a kind of conflict. More formally, it is useful to associate to each pre-AES an explicit conflict relation (on sets of events) defined in the following way:

DEFINITION 3.2 (INDUCED CONFLICT RELATION)
*Let $G = \langle E, \leq, \nearrow \rangle$ be a pre-*AES*. The* conflict *relation $\#^a \subseteq \mathbf{2}^E_{fin}$ associated to G is defined as:*

$$\frac{e_0 \nearrow e_1 \nearrow \ldots \nearrow e_n \nearrow e_0}{\#^a\{e_0, e_1, \ldots, e_n\}} \qquad \frac{\#^a(A \cup \{e\}) \quad e \leq e'}{\#^a(A \cup \{e'\})}$$

where A denotes a generic finite subset of E. The superscript "a" in $\#^a$ reminds that this relation is induced by asymmetric conflict. Sometimes we use the infix notation for the "binary version" of the conflict, i.e., we write $e\#^a e'$ for $\#^a\{e, e'\}$.

Notice that if $\#^a A$ then $\lfloor A \rfloor$ contains a cycle of asymmetric conflict, and, vice versa, if $\lfloor A \rfloor$ contains a cycle $e_0 \nearrow e_1 \ldots e_n \nearrow e_0$ then there exists a subset $A' \subseteq A$ such that $\#^a A'$ (for instance, by choosing an event $a_i \in A$ such that $e_i \leq a_i$ for each $i \in \{0, \ldots, n\}$ the set A' can be $\{a_i \mid i \in \{0, \ldots, n\}\}$).

Clearly, by the rules above, if $e \nearrow e'$ and $e' \nearrow e$ then $\#^a\{e, e'\}$. The converse, instead, does not hold, namely in general we can have $e\#^a e'$ and $\neg(e \nearrow e')$, as in the AES Figure 3.3, because $\#^a$ is inherited along $<$, while $\nearrow$ is not. An asymmetric

event structure is a pre-AES where each binary conflict is induced directly by an asymmetric conflict in both directions.

DEFINITION 3.3
An asymmetric event structure *(*AES*) is a pre-*AES $G = \langle E, \leq, \nearrow \rangle$ *such that for any* $e, e' \in E$, *if* $e \#^a e'$ *then* $e \nearrow e'$.

Observe that given any pre-AES $G = \langle E, \leq, \nearrow \rangle$, we can always "saturate" G in order to obtain an AES $\overline{G} = \langle E, \leq, \nearrow' \rangle$, by defining $\nearrow'$ as $e \nearrow' e'$ if and only if $(e \nearrow e') \vee (e \#^a e')$. Furthermore it is easy to verify that the conflict relations of G and of G' coincide.

3.1.2 Morphisms of asymmetric event structures

The notion of AES-morphism is a quite natural extension of that of PES-morphism. Intuitively, it is a (possibly partial) mapping of events that "preserves computations".

DEFINITION 3.4 (AES-MORPHISM)
Let $G_0 = \langle E_0, \leq_0, \nearrow_0 \rangle$ *and* $G_1 = \langle E_1, \leq_1, \nearrow_1 \rangle$ *be two* AES*'s. An* AES-morphism $f : G_0 \to G_1$ *is a partial function* $f : E_0 \to E_1$ *such that:*

1. *for all* $e_0 \in E_0$, *if* $f(e_0) \neq \perp$ *then* $\lfloor f(e_0) \rfloor \subseteq f(\lfloor e_0 \rfloor)$;

2. *for all* $e_0, e_0' \in E_0$, *if* $f(e_0) \neq \perp \neq f(e_0')$ *then*

   (a) $(f(e_0) = f(e_0')) \wedge (e_0 \neq e_0') \quad \Rightarrow \quad e_0 \#_0^a e_0'$.

   (b) $f(e_0) \nearrow_1 f(e_0') \quad \Rightarrow \quad e_0 \nearrow_0 e_0'$;

It is easy to show that AES-morphisms are closed under composition. In fact, let $f_0 : G_0 \to G_1$ and $f_1 : G_1 \to G_2$ be AES-morphisms. The fact that $f_1 \circ f_0$ satisfies conditions (1) and (2.a) of Definition 3.4 is proved as for ordinary PES's. The validity of condition (2.b) is straightforward.

DEFINITION 3.5 (CATEGORY **AES**)
We denote by **AES** *the category having asymmetric event structures as objects and* AES*-morphisms as arrows.*

NOTATION 3.6
In the following when considering a PES P and an AES G, we implicitly assume that $P = \langle E, \leq, \# \rangle$ and $G = \langle E, \leq, \nearrow \rangle$. Moreover superscripts and subscripts on the structure name carry over the names of the involved sets, functions and relations (e.g., $G_i = \langle E_i, \leq_i, \nearrow_i \rangle$).

The binary conflict in an AES is represented by asymmetric conflict in both directions, and thus, analogously to what happens for PES's, it is reflected by AES-morphisms. The next lemma shows that AES-morphisms reflect also the general (induced) conflict relation.

LEMMA 3.7 (AES-MORPHISMS REFLECT CONFLICTS)
Let G_0 and G_1 be two AES*'s and let $f : G_0 \to G_1$ be an* AES*-morphism. Given a set of events $A \subseteq_{fin} E_0$, if $\#_1^a f(A)$ then $\#_0^a A'$ for some $A' \subseteq A$.*

PROOF. Let $A \subseteq_{fin} E_0$ and let $\#_1^a f(A)$. By definition of conflict there is a $\nearrow_1$-cycle $e'_0 \nearrow_1 e'_1 \nearrow_1 \ldots \nearrow_1 e'_n \nearrow_1 e'_0$ in $\lfloor f(A) \rfloor$. By definition of AES-morphism, we have that $\lfloor f(A) \rfloor \subseteq f(\lfloor A \rfloor)$ and thus we can find $e_0, \ldots, e_n \in \lfloor A \rfloor$ such that $e'_i = f(e_i)$ for all $i \in \{0, \ldots, n\}$. Consider $A' = \{a_0, \ldots, a_n\} \subseteq A$ such that $e_i \leq_0 a_i$ for $i \in \{0, \ldots, n\}$. By definition of AES-morphism, $e_0 \nearrow_0 e_1 \nearrow_0 \ldots \nearrow_0 e_0$, and thus $\#_0^a A'$. □

3.1.3 Relating asymmetric and prime event structures

We conclude this section by formalizing the relation between AES's and PES's. We show that AES's are a proper extension of PES's, in the sense that, as one would expect, PES's can be identified with the subclass of AES's where the strict asymmetric conflict relation is actually symmetric. The corresponding full embedding functor is right adjoint to the forgetful functor from **AES** to **PES**.

LEMMA 3.8
Let $P = \langle E, \leq, \# \rangle$ be a PES*. Then $G = \langle E, \leq, < \cup \# \rangle$ is an* AES*, where the asymmetric conflict relation is defined as the union of the "strict" causality and conflict relations.*

Moreover, if $f : P_0 \to P_1$ is a PES*-morphism then f is an* AES*-morphism between the corresponding* AES*'s G_0 and G_1, and if $g : G_0 \to G_1$ is an* AES*-morphism then it is also a* PES*-morphism between the original* PES*'s.*

PROOF. Let $P = \langle E, \leq, \# \rangle$ be a PES. The fact that $G = \langle E, \leq, < \cup \# \rangle$ is an AES is a trivial consequence of the definitions. In particular, the asymmetric conflict relation of G is acyclic on the causes of each event since $\#$ is hereditary with respect to $\leq$ and irreflexive, and $<$ is a strict partial order (namely an irreflexive and transitive relation) in P.

Now, let $f : P_0 \to P_1$ be a PES-morphism. To prove that f is also an AES-morphism between the corresponding AES's G_0 and G_1, first observe that, according the definition of $\leq_{G_i}$ and $\nearrow_{G_i}$, the validity of the conditions (1) and (2.a) of Definition 3.4 follows immediately from the corresponding conditions in the definition of PES-morphism (Definition 2.11). As for condition (2.b), if $f(e_0) \nearrow_{G_1} f(e_1)$, then, by construction, $f(e_0) <_{P_1} f(e_1)$ or $f(e_0) \#_{P_1} f(e_1)$ and thus, by properties of PES's (easily derivable from Definition 2.11), in the first case $e_0 <_{P_0} e_1$ or $e_0 \#_{P_0} e_1$ whilst, in the second case, $e_0 \#_{P_0} e_1$. Hence, in both cases, $e_0 \nearrow_{G_0} e_1$.

Similar considerations allow us to conclude that, if $g : G_0 \to G_1$ is an AES-morphism, then it is also a PES-morphism between the original PES's. □

By the previous lemma, there is a full embedding functor from **PES** into **AES** that transforms each PES by replacing the symmetric conflict with an asymmetric conflict relation given by the union of "strict" causality and conflict relation, and that is identity on arrows.

Proposition 3.9 (from pes's to aes's)
The functor $\mathscr{J} : \mathbf{PES} \to \mathbf{AES}$ *defined by*

- $\mathscr{J}(\langle E, \leq, \# \rangle) = \langle E, \leq, < \cup \# \rangle$;
- $\mathscr{J}(f : P_0 \to P_1) = f$

is a full embedding of **PES** *into* **AES**.

Observe that given any AES $G = \langle E, \leq, \nearrow \rangle$ the induced binary conflict, which can be expressed as $\nearrow \cap \nearrow^{-1}$, satisfies all the properties of the conflict relation of PES's, i.e., it is irreflexive (since $\nearrow$ is irreflexive), symmetric and hereditary with respect to $\leq$. Therefore $\langle E, \leq, \# \rangle$, where $\# = \nearrow \cap \nearrow^{-1}$, is a PES, in the following referred to as the PES underlying the AES G.

Definition 3.10 (pes underlying an aes)
We denote by $\mathscr{F}_{ap}$: **AES** $\to$ **PES** *the forgetful functor defined on objects as* $\mathscr{F}_{ap}(G) = \langle E, \leq, \nearrow \cap \nearrow^{-1} \rangle$ *and which is the identity on arrows.*

The functor is well-defined, as it can be verified by showing that each AES-morphism is a PES-morphism between the underlying PES's. Furthermore for any PES P it is easy to see that $\mathscr{F}_{ap}(\mathscr{J}(P)) = P$. The functor $\mathscr{F}_{ap}$ is left adjoint to $\mathscr{J}$ and they establish a reflection between **AES** and **PES**, the component of the counit at a PES P being the identity id_P.

Proposition 3.11 (relating aes's and pes's)
$\mathscr{F}_{ap} \dashv \mathscr{J}$.

3.2 From asymmetric event structures to domains

As shown in Section 2.4, the category **PES** of prime event structures is equivalent to the category **Dom** of (finitary coherent) prime algebraic domains. For asymmetric event structures this result generalizes to the existence of a coreflection between **AES** and **Dom**. Such a coreflection allows for an elegant translation of an AES semantics into a domain, and thus into a classical PES semantics. The PES semantics obtained in this way represents asymmetric conflicts via symmetric conflict and causality with a duplication of events, as described in Section 3.1 (see Figure 3.1).

The domain corresponding to an AES G is obtained by considering the configurations of G, suitably ordered using the asymmetric conflict relation. Vice versa, given a domain D we obtain the corresponding AES by applying first the functor

$\mathscr{P} : \mathbf{Dom} \to \mathbf{PES}$ (see Section 2.4) and then the embedding $\mathscr{J} : \mathbf{PES} \to \mathbf{AES}$, defined in Proposition 3.9.

3.2.1 The domain of configurations of an AES

As already explained, a configuration of an event structure is a set of events representing a possible computation of the system modelled by the event structure. The presence of the asymmetric conflict relation makes such definition slightly more involved than the traditional one.

DEFINITION 3.12 (CONFIGURATION)
Let $G = \langle E, \leq, \nearrow \rangle$ be an AES. *A* configuration *of G is a set of events $C \subseteq E$ such that*

1. *$\nearrow_C$ is well-founded;*
2. *$\{e' \in C \mid e' \nearrow e\}$ is finite for all $e \in C$;*
3. *C is left-closed with respect to $\leq$, i.e., for all $e \in C$, $e' \in E$, $e' \leq e$ implies $e' \in C$.*

The set of all configurations of G is denoted by $Conf(G)$.

Condition (1) first ensures that in C there are no $\nearrow$-cycles, and thus excludes the possibility of having in C a subset of events in conflict (formally, for any $A \subseteq_{fin} C$, we have $\neg(\#^a A)$). Moreover it guarantees that $\nearrow$ has no infinite descending chains in C, that, together with condition (2), implies that the set $\{e' \in C \mid e'(\nearrow_C)^+ e\}$ is finite for each event e in C; thus each event has to be preceded only by finitely many other events of the configuration. Finally condition (3) requires that all the causes of each event are present.

If a set of events A satisfies only the first two properties of Definition 3.12 it is called *consistent* and we write $co(A)$. Notice that, unlike for traditional event structures, consistency is not a finitary property.[3] For instance, let $A = \{e_i \mid i \in \mathbb{N}\} \subseteq E$ be a set of events such that all e_i's are distinct and $e_{i+1} \nearrow e_i$ for all $i \in \mathbb{N}$. Then A is not consistent, but each finite subset of A is.

Let us now define an order $\sqsubseteq$ on the configurations of an AES, aimed at formalizing the idea of "computational extension", namely such that $C_1 \sqsubseteq C_2$ if the configuration C_1 can evolve into C_2. A remarkable difference with respect to Winskel's event structures (see, e.g., Definition 2.10) is that the order on configurations is not simply set-inclusion, since a configuration C cannot be extended with an event inhibited by some of the events already present in C.

[3] A property Q on the subsets of a set X is *finitary* if given any $Y \subseteq X$, from $Q(Z)$ for all finite subsets $Z \subseteq Y$ it follows $Q(Y)$.

Definition 3.13 (extension)
Let $G = \langle E, \leq, \nearrow \rangle$ be an aes *and let $A, A' \subseteq E$ be sets of events. We say that A'* extends *A and we write $A \sqsubseteq A'$, if*

1. $A \subseteq A'$;
2. $\neg(e' \nearrow e) \quad$ *for all* $e \in A,\ e' \in A' - A$.

Often in the sequel it will be preferable to use the following condition, equivalent to (2):

$$\forall e \in A.\ \forall e' \in A'.\ e' \nearrow e \ \Rightarrow\ e' \in A.$$

The extension relation is a partial order on the set $Conf(G)$ of configurations of an aes. Our aim is now to prove that $\langle Conf(G), \sqsubseteq \rangle$ is a finitary prime algebraic domain. This means that like prime event structures [Win87a], flow event structure [Bou90], and prioritized event structures [DGV93], also asymmetric event structures provide a concrete presentation of prime algebraic domains.

Given an aes G, in the following we will denote by $Conf(G)$ both the set of configurations of G and the corresponding partial order. The following proposition presents a simple but useful property of the partial order of configurations of an aes, strictly connected with coherence. Recall from Section 2.4 that, given a partial order $(D, \sqsubseteq)$, two elements $d, d' \in D$ are called compatible, written $d \uparrow d'$, if there exists $d'' \in D$ such that $d \sqsubseteq d''$ and $d' \sqsubseteq d''$. Furthermore, a subset $X \subseteq D$ is called pairwise compatible $d \uparrow d'$ for any $d, d' \in X$.

Lemma 3.14
Let G be an aes *and let $A \subseteq Conf(E)$ be a pairwise compatible set of configurations. Then for all $C \in A$ and $e \in C$*

$$e' \in \bigcup A \ \wedge\ e' \nearrow e \quad \Rightarrow \quad e' \in C;$$

Proof. Let $e' \in \bigcup A$ be an event such that $e' \nearrow e$. Then there is a configuration $C' \in A$ such that $e' \in C'$. Being C and C' compatible, there is $C'' \in Conf(G)$ such that $C, C' \sqsubseteq C''$. Thus $e' \in C''$ and, since $C \sqsubseteq C''$, by definition of $\sqsubseteq$ we conclude that $e' \in C$. □

The next lemma proves that for pairwise compatible sets of configurations the least upper bound and the greatest lower bound are simply given by union and intersection.

Lemma 3.15 ($\bigsqcup$ and $\bigsqcap$ for sets of configurations)
Let G be an aes. *Then*

1. *if $A \subseteq Conf(E)$ is pairwise compatible then $\bigsqcup A = \bigcup A$;*
2. *if $C_0 \uparrow C_1$ then $C_0 \sqcap C_1 = C_0 \cap C_1$.*

PROOF.

1. Let $A \subseteq \mathit{Conf}(E)$ be a pairwise compatible set of configurations. First notice that $\bigcup A$ is a configuration. In fact:

   - $\nearrow_{\bigcup A}$ is well-founded.
     Let us suppose that there is in $\bigcup A$ an infinite descending chain:
     $$\ldots e_{i+1} \nearrow e_i \nearrow e_{i-1} \nearrow \ldots \nearrow e_0.$$
     Let $C \in A$ such that $e_0 \in C$. Lemma 3.14, together with an inductive reasoning, ensure that this infinite chain is entirely contained in C. But this contradicts $C \in \mathit{Conf}(G)$.
   - $\{e' \in \bigcup A \mid e' \nearrow e\}$ is finite for all $e \in \bigcup A$.
     Let $e \in \bigcup A$, then there exists $C \in A$ such that $e \in C$. By Lemma 3.14, the set $\{e' \in \bigcup A \mid e' \nearrow e\} = \{e' \in C \mid e' \nearrow e\}$, and thus it is finite.
   - $\bigcup A$ is left-closed, since each $C \in A$ is left-closed.

   The configuration $\bigcup A$ is an upper bound for A. In fact, for any $C \in A$, clearly $C \subseteq \bigcup A$ and for all $e \in C$, $e' \in \bigcup A$, if $e' \nearrow e$ then, by Lemma 3.14, $e' \in C$. Thus $C \sqsubseteq \bigcup A$. Moreover, if C_0 is another upper bound for A, namely a configuration such that $C \sqsubseteq C_0$ for all $C \in A$, then $\bigcup A \subseteq C_0$. Furthermore, for any $e \in \bigcup A$, $e' \in C_0$ with $e' \nearrow e$, since $e \in C$ for some $C \in A$ we conclude that $e' \in C \subseteq \bigcup A$. Thus $\bigcup A \sqsubseteq C_0$ and this shows that $\bigcup A$ is the least upper bound of A.

2. Let C_0 and C_1 be two compatible configurations and let $C = C_0 \cap C_1$. Then it is easily seen that C is a configuration. Moreover $C \sqsubseteq C_0$. In fact $C \subseteq C_0$ and for all $e \in C$, $e' \in C_0$, if $e' \nearrow e$ then, since $e \in C_1$ and $C_0 \uparrow C_1$, by Lemma 3.14, $e' \in C_1$ and thus $e' \in C$. In the same way $C \sqsubseteq C_1$, and thus C is a lower bound for C_0 and C_1. To show that C is the greatest lower bound observe that if $C' \sqsubseteq C_0, C_1$ is another lower bound then, clearly $C' \subseteq C$. Furthermore, if $e \in C'$, $e' \in C$ with $e' \nearrow e$, since, in particular, $e' \in C_0$, we conclude $e' \in C'$. Hence $C' \sqsubseteq C$. □

For prime event structures an event e uniquely determines its history, that is the set $\lfloor e \rfloor$ of its causes, independently of the configuration at hand. In the case of asymmetric event structures, instead, an event e may have different histories, in the sense that the set of events that must precede e in a configuration C depends on C. Essentially, the possible histories of e are obtained inserting or not in a configuration the weak causes of e, which thus can be seen as "possible causes".

DEFINITION 3.16 (POSSIBLE HISTORY)
Let G be an AES *and let $e \in E$. Given a configuration $C \in \mathit{Conf}(G)$ such that $e \in C$, the* history of e in C *is defined as $C[\![e]\!] = \{e' \in C \mid e' \nearrow_C^* e\}$. The set of* (possible) histories *of e, denoted by $\mathit{Hist}(e)$, is then defined as*

$$\mathit{Hist}(e) = \{C[\![e]\!] \mid C \in \mathit{Conf}(E) \ \wedge \ e \in C\}.$$

We denote by $\mathit{Hist}(G)$ the set of possible histories of all events in G, namely

$$\mathit{Hist}(G) = \bigcup\{\mathit{Hist}(e) \mid e \in E\}.$$

Notice that each history is a *finite* set of events since, by conditions (1) and (2) in the definition of configuration, $(\nearrow_C)^*$ is a finitary partial order.

Let us now give some properties of the set of histories. Point (1) shows that each history of an event e in a configuration C, is itself a configuration which is extended by C. Point (2) essentially states that although an event e has in general more than one history, as one would expect, the history cannot change after the event has occurred. Point (3) asserts that different histories of the same event are incompatible.

Lemma 3.17 (history properties)
Let G be an aes. *Then in $\langle Conf(G), \sqsubseteq\rangle$ we have that:*

1. *if $C \in Conf(G)$ and $e \in C$, then $C[\![e]\!] \in Conf(G)$. Moreover $C[\![e]\!] \sqsubseteq C$;*
2. *if $C, C' \in Conf(G)$, $C \uparrow C'$ and $e \in C \cap C'$ then $C[\![e]\!] = C'[\![e]\!]$; in particular this holds for $C \sqsubseteq C'$;*
3. *if $e \in E$, $C_0, C_1 \in Hist(e)$ and $C_0 \uparrow C_1$ then $C_0 = C_1$.*

Proof.

1. Obviously, $C[\![e]\!] \in Conf(G)$. In fact, the requirements (1) and (2) of the definition of configuration are trivially satisfied, while (3) follows by recalling that $\nearrow \supseteq <$. Moreover $C[\![e]\!] \subseteq C$ and if $e' \in C[\![e]\!]$, $e'' \in C$ and $e'' \nearrow e'$, then $e'' \nearrow e'(\nearrow_C)^* e$, thus $e'' \in C[\![e]\!]$. Therefore $C[\![e]\!] \sqsubseteq C$.

2. By Lemma 3.14, since $C \uparrow C'$ and $e \in C$, an inductive reasoning ensures that if $e_0 \nearrow e_1 \nearrow \ldots \nearrow e_n \nearrow e$, with $e_i \in C \cup C'$, then each e_i is in C. Therefore $C[\![e]\!] = (C \cup C')[\![e]\!] = C'[\![e]\!]$.

3. Since $C_0 \uparrow C_1$ and $e \in C_0 \cap C_1$, by (2), we have that $C_0 = C_0[\![e]\!] = C_1[\![e]\!] = C_1$. □

We are now able to show that the complete primes of $Conf(G)$ are exactly the possible histories of events in G.

Lemma 3.18 (primes)
Let G be an aes. *Then*

1. *for all configurations $C \in Conf(G)$*

$$C = \bigsqcup\{C' \in Hist(G) \mid C' \sqsubseteq C\} = \bigsqcup\{C[\![e]\!] \mid e \in C\}.$$

2. *$Pr(Conf(G)) = Hist(G)$ and $Pr(C) = \{C[\![e]\!] \mid e \in C\}$.*

PROOF.

1. Let $C \in \mathit{Conf}(G)$ and let $C_0 = \bigsqcup\{C' \in \mathit{Hist}(G) \mid C' \sqsubseteq C\}$. Then clearly $C_0 \sqsubseteq C$. Moreover for all $e \in C$, by Lemma 3.17.(1), the history $C[\![e]\!] \sqsubseteq C$ and thus $e \in C[\![e]\!] \subseteq C_0$. This gives the converse inclusion and allows us to conclude $C = C_0$.

2. Let $C[\![e]\!] \in \mathit{Hist}(e)$, for some $e \in E$, be a history and let $A \subseteq \mathit{Conf}(G)$ be a pairwise compatible set of configurations. If $C[\![e]\!] \sqsubseteq \bigsqcup A$, then $e \in \bigcup A$. Thus there exists $C_e \in A$ such that $e \in C_e$. Therefore:

$$\begin{array}{ll} C[\![e]\!] = (\bigsqcup A)[\![e]\!] & \text{[by Lemma 3.17.(2), since } C[\![e]\!] \sqsubseteq \bigsqcup A\text{]} \\ \quad = C_e[\![e]\!] & \text{[by Lemma 3.17.(2), since } C_e \sqsubseteq \bigsqcup A\text{]} \\ \quad \sqsubseteq C_e & \text{[by Lemma 3.17.(1)]} \end{array}$$

   Therefore $C[\![e]\!]$ is a complete prime in $\mathit{Conf}(G)$.

   Conversely, let $C \in \mathit{Pr}(\mathit{Conf}(G))$. Then, by point (1),

$$C = \bigsqcup\{C' \in \mathit{Hist}(G) \mid C' \sqsubseteq C\}.$$

   Being C a complete prime, there must exist $C' \in \mathit{Hist}(G)$, $C' \sqsubseteq C$ such that $C \sqsubseteq C'$ and thus $C = C' \in \mathit{Hist}(G)$. □

It is now immediate to prove that the configurations of an AES ordered by the extension relation form a finitary prime algebraic domain.

THEOREM 3.19 (CONFIGURATIONS FORM A DOMAIN)
Let G be an AES. *Then $\langle \mathit{Conf}(G), \sqsubseteq \rangle$ is a (coherent finitary prime algebraic) domain.*

PROOF. By Lemma 3.15.(1), $\mathit{Conf}(G)$ is a coherent partial order. By Lemma 3.18, for any configuration $C \in \mathit{Conf}(G)$

$$\mathit{Pr}(C) = \{C[\![e]\!] \mid e \in C\}$$

and $C = \bigsqcup C[\![e]\!]$. Therefore $\mathit{Conf}(G)$ is prime algebraic.

Finally, $\mathit{Conf}(G)$ is finitary, as it immediately follows from the fact that compact elements in $\mathit{Conf}(G)$ are exactly the finite configurations. To see this, let $C \in \mathit{Conf}(G)$ be finite and let us consider a directed $A \subseteq \mathit{Conf}(G)$ such that $C \sqsubseteq \bigsqcup A$. Then we can choose, for all $e \in C$, $C_e \in A$ such that $e \in C_e$. Being A directed and C finite, the set $\{C_e \mid e \in C\}$ has an upper bound $C' \in A$. Then $C = \bigsqcup_{e \in C} C[\![e]\!] = \bigsqcup_{e \in C} C_e[\![e]\!] \sqsubseteq C'$ follows immediately from Lemma 3.17. Thus C is compact. For the converse, let $C \in \mathit{Conf}(G)$ be a compact element. Since each possible history is finite, $\{\bigcup_{e \in Z} C[\![e]\!] \mid Z \subseteq_{\mathit{fin}} C\}$ is a directed set of *finite* configurations, having C as least upper bound. Since C is compact, we conclude that there exists $Z \subseteq_{\mathit{fin}} C$ such that $C \sqsubseteq \bigcup_{e \in Z} C[\![e]\!]$. Thus $C = \bigcup_{e \in Z} C[\![e]\!]$ is finite. □

An example of AES with the corresponding domain can be found in Figure 3.7, (a) and (b), at the end of Section 3.6 (for the moment, the reader should disregard how the AES is obtained from the c-net N). In particular notice how asymmetric conflict influences the order on configurations, which is different from set-inclusion. For instance, $\{t_0, t_4\} \subseteq \{t_0, t_1', t_4\}$, but $\{t_0, t_4\} \not\sqsubseteq \{t_0, t_1', t_4\}$ since $t_1' \nearrow t_4$.

3.2.2 A coreflection between AES and Dom

To prove that the construction which associates to an AES the domain of its configurations is functorial we first give a characterization of the immediate predecessors of a configuration. As one could expect, we pass from a configuration to one of its immediate successors by executing a single event.

Lemma 3.20 (immediate precedence)
Let G be an AES *and let $C \sqsubseteq C'$ be configurations in $Conf(G)$. Then*

$$C \prec C' \quad \textit{iff} \quad |\, C' - C \,| = 1.$$

where $\prec$ denotes the immediate precedence relation on configurations.

Proof. ($\Rightarrow$) Let $C \prec C'$ and let $e', e'' \in C' - C$. We have $C \sqsubset C \sqcup (C'[\![e']\!]) \sqsubseteq C'$ and thus, by definition of immediate precedence, $C' = C \cup (C'[\![e']\!])$. In the same way $C' = C \cup C'[\![e'']\!]$. Hence, by definition of history, we have $e'(\nearrow_{C'})^* e''(\nearrow_{C'})^* e'$ and thus $e' = e''$ (otherwise $\nearrow_{C'}$ would not be acyclic, contradicting the definition of configuration).
($\Leftarrow$) Obvious. □

The following lemma leads to the definition of a functor from **AES** to **Dom**. First we prove that AES-morphisms preserve configurations and then we show that the function naturally induced by an AES-morphism between the corresponding domains of configurations is a domain morphism.

Lemma 3.21 (AES-morphisms preserve configurations)
Let G_0, G_1 be two AES*'s and let $f : G_0 \to G_1$ be an* AES*-morphism. Then for each $C_0 \in Conf(G_0)$ the morphism f is injective on C_0 and the f-image of C_0 is a configuration of G_1, i.e.,*

$$f^*(C_0) = \{f(e) \mid e \in C_0\} \in Conf(G_1).$$

Moreover $f^* : Conf(G_0) \to Conf(G_1)$ is a domain morphism.

Proof. Let $C_0 \in Conf(G_0)$ be a configuration. Since $\nearrow_{C_0}$ is well founded and thus $\neg(e \#^a e')$ for all $e, e' \in C_0$, the conditions of the definition of AES-morphism (Definition 3.4) imply that for all e, e' in C_0 such that $f(e) \neq \bot \neq f(e')$:

$$\begin{array}{l} \lfloor f(e) \rfloor \subseteq f(\lfloor e \rfloor); \\ f(e) = f(e') \quad \Rightarrow \quad e = e'; \\ f(e) \nearrow_1 f(e') \quad \Rightarrow \quad e \nearrow_0 e'. \end{array}$$

Therefore f is injective on C_0 (as expressed by the second condition) and we immediately conclude that $f^*(C_0)$ is a configuration in G_1.

Let us now prove that $f^* : Conf(G_0) \to Conf(G_1)$ is a domain morphism. Additivity and stability follow from Lemma 3.15. In particular for stability one should also observe that if C_0 and C_1 are compatible then f is injective on $C_1 \cup C_2$ and thus $f(C_1 \cap C_2) = f(C_1) \cap f(C_2)$. Finally, the fact that f^* preserves immediate precedence can be straightforwardly derived from Lemma 3.20.
□

Theorem 3.19 and Lemma 3.21 suggest how to define a functor from the category **AES** of asymmetric event structures to the category **Dom** of domains. Instead, the functor going back from **Dom** to **AES** first transforms a domain into a PES via $\mathscr{P} : \mathbf{Dom} \to \mathbf{PES}$, introduced in Definition 2.17, and then embeds such PES into **AES** via $\mathscr{J} : \mathbf{PES} \to \mathbf{AES}$, defined in Proposition 3.9.

DEFINITION 3.22 (FROM AES'S TO DOMAINS AND BACKWARDS)
The functor $\mathscr{L}_a : \mathbf{AES} \to \mathbf{Dom}$ *is defined as:*

- *for any* **AES***-object* G,

$$\mathscr{L}_a(G) = \langle Conf(G), \sqsubseteq \rangle;$$

- *for any* **AES***-morphism* $f : G_0 \to G_1$,

$$\mathscr{L}_a(f) = f^* : \mathscr{L}_a(G_0) \to \mathscr{L}_a(G_1).$$

The functor $\mathscr{P}_a : \mathbf{Dom} \to \mathbf{AES}$ *is defined as* $\mathscr{J} \circ \mathscr{P}$.

It is worth recalling that, concretely, given a domain $\langle D, \sqsubseteq \rangle$, the PES $\mathscr{P}(D)$ is defined as $\langle Pr(D), \sqsubseteq, \# \rangle$, where $\#$ is the incompatibility relation (i.e., $p \# p'$ iff p and p' do not have a common upper bound). Then $\mathscr{P}_a(D) = \mathscr{J}(\mathscr{P}(D))$ is the corresponding AES, namely $\langle Pr(D), \sqsubseteq, \sqsubset \cup \# \rangle$.

The functor $\mathscr{P}_a$ is left adjoint to $\mathscr{L}_a$ and they establish a coreflection between **AES** and **Dom**. The counit of the adjunction maps each history of an event e into the event e itself. The next technical lemma shows that the function defined in this way is indeed an AES-morphism.

LEMMA 3.23
Let G *be an* AES. *Then* $\epsilon_G : \mathscr{P}_a(\mathscr{L}_a(G)) \to G$ *defined as:*

$$\epsilon_G(C) = e \qquad \text{if } C \in Hist(e),$$

is an AES*-morphism.*

PROOF. Let us verify that ϵ_G satisfies the three conditions imposed on AES-morphisms: for all $C, C' \in Hist(G)$, with $C \in Hist(e)$, $C' \in Hist(e')$:

- $\lfloor \epsilon_G(C) \rfloor \subseteq \epsilon_G(\lfloor C \rfloor)$.
  We have:

$$\begin{aligned}
&\epsilon_G(\lfloor C \rfloor) = \\
&\quad = \epsilon_G(Pr(C)) \\
&\quad = \epsilon_G(\{C[\![e']\!] \mid e' \in C\}) && \text{[by Lemma 3.18]} \\
&\quad = C \\
&\quad \supseteq \lfloor e \rfloor && \text{[since } C \text{ is left-closed]} \\
&\quad = \lfloor \epsilon_G(C) \rfloor
\end{aligned}$$

- $(\epsilon_G(C) = \epsilon_G(C')) \ \wedge \ C \neq C' \quad \Rightarrow \quad C \#^a C'$.
  Let $\epsilon_G(C) = e = e' = \epsilon_G(C')$ and $C \neq C'$. Since $C, C' \in \mathit{Hist}(e)$, by Lemma 3.17, we have $\neg(C \uparrow C')$ and thus $C \# C'$ in $\mathscr{P}(\mathscr{L}_a(G))$ and therefore, by definition of $\mathscr{J}$, $C \#^a C'$ in $\mathscr{P}_a(\mathscr{L}_a(G))$.

- $\epsilon_G(C) \nearrow \epsilon_G(C') \quad \Rightarrow \quad C \nearrow C'$.
  Let $\epsilon_G(C) = e \nearrow e' = \epsilon_G(C')$. Since the relation $\nearrow$ is irreflexive, surely $e \neq e'$ and thus $C \neq C'$. Now, if $e \notin C'$ then, by Lemma 3.14, surely $\neg(C \uparrow C')$, thus $C \# C'$ in $\mathscr{P}(\mathscr{L}_a(G))$ and therefore, by definition of $\mathscr{J}$, $C \nearrow C'$ in $\mathscr{P}_a(\mathscr{L}_a(G))$. Otherwise, if $e \in C'$ we distinguish two cases:

  - $C = C[\![e]\!] = C'[\![e]\!]$.
    In this case, by Lemma 3.17.(1), we have that $C \sqsubseteq C'$, and the relation is strict, since $C \neq C'$. Thus, by definition of $\mathscr{P}_a$, $C \nearrow C'$ in $\mathscr{P}_a(\mathscr{L}_a(G))$.
  - $C = C[\![e]\!] \neq C'[\![e]\!]$.
    In this case, by Lemma 3.17.(2), we conclude that C and C' are not compatible, namely $\neg(C \uparrow C')$. Hence $C \# C'$ in $\mathscr{P}(\mathscr{L}_a(G))$ and therefore $C \nearrow C'$ in $\mathscr{P}_a(\mathscr{L}_a(G))$.

□

The next technical lemma characterizes the behaviour of the functor $\mathscr{P}_a$ on morphisms having a domain of configurations as codomain.

Lemma 3.24
Let G be an aes*, D a domain and let $g : D \to \mathscr{L}_a(G)$ be a domain morphism. Then for all $p \in Pr(D)$, $\mid g(p) - \bigcup g(Pr(p) - \{p\}) \mid \leq 1$ and*

$$\mathscr{P}_a(g)(p) = \begin{cases} \bot & \text{if } g(p) - \bigcup g(Pr(p) - \{p\}) = \emptyset \\ g(p)[\![e]\!] & \text{if } g(p) - \bigcup g(Pr(p) - \{p\}) = \{e\} \end{cases}$$

Proof. Let $p \in Pr(D)$ and let us consider the corresponding prime interval

$$[\bigsqcup(Pr(p) - \{p\}), p]\,,$$

then also

$$\Big[g(\bigsqcup(Pr(p) - \{p\})), g(p)\Big]\,, \tag{3.1}$$

is a prime interval in $\mathscr{L}_a(G)$, and, by definition of the functor $\mathscr{E}_a$ (Definition 3.22)

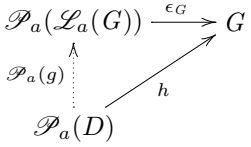
$$\mathscr{P}_a(g)(p) = \begin{cases} \bot & \text{if } g(p) = g(\bigsqcup(Pr(p) - \{p\})) \\ C & \text{if } Pr(g(p)) - Pr(g(\bigsqcup(Pr(p) - \{p\}))) = \{C\} \end{cases}$$

Now, by additivity of g and Lemma 3.15.(1), $g(\bigsqcup(Pr(p) - \{p\})) = \bigsqcup g(Pr(p) - \{p\}) = \bigcup g(Pr(p) - \{p\})$, and, since (3.1) is a prime interval, by Lemma 3.20, $g(p) - \bigcup g(Pr(p) - \{p\})$ has at most one element. If $g(p) = \bigcup g(Pr(p) - \{p\})$ then $\mathscr{P}_a(g)(p) = \bot$. Otherwise, if $g(p) - \bigcup g(Pr(p) - \{p\}) = \{e\}$, then, by Lemma 3.18.(2), we have that $Pr(g(p)) - Pr(\bigcup g(Pr(p) - \{p\})) = \{g(p)[\![e]\!]\}$ and thus we conclude. □

Finally we can prove the main result of this section, namely, that $\mathscr{P}_a$ is left adjoint to $\mathscr{L}_a$ and they establish a coreflection between **AES** and **Dom**. Given an aes G, the component at G of the counit of the adjunction is $\epsilon_G : \mathscr{P}_a \circ \mathscr{L}_a(G) \to G$.

THEOREM 3.25 (COREFLECTION BETWEEN **AES** AND **Dom**)
$\mathscr{P}_a \dashv \mathscr{L}_a$.

PROOF. Let G be an AES and let $\epsilon_G : \mathscr{P}_a(\mathscr{L}_a(G)) \to G$ be the morphism defined as in Lemma 3.23. We have to show that given any domain D and AES-morphism $h : \mathscr{P}_a(D) \to G$, there is a unique domain morphism $g : D \to \mathscr{L}_a(G)$ such that the following diagram commutes:



Existence
Let $g : D \to \mathscr{L}_a(G)$ be defined as:

$$g(d) = h^*(Pr(d)).$$

A straightforward checking shows that $Pr(d)$ is a configuration in $\mathscr{P}_a(D)$ and thus, by Lemma 3.21, h is injective on $Pr(d)$ and $h^*(Pr(d))$ is a configuration in G, i.e., an element of $\mathscr{L}_a(G)$. Moreover g is a domain morphism. In fact it is

- $\preceq$*-preserving.* Let $d, d' \in D$, with $d \prec d'$. Then $Pr(d') - Pr(d) = \{p\}$ and thus

  $$\begin{aligned} &g(d') - g(d) = \\ &\quad = h^*(Pr(d')) - h^*(Pr(d)) \\ &\quad \subseteq \{h(p)\} \end{aligned}$$

  Therefore $|g(d') - g(d)| \leq 1$ and, since it is easy to see that $g(d) \sqsubseteq g(d')$, by Lemma 3.20 we conclude $g(d) \preceq g(d')$.

- *Additive.* Let $X \subseteq D$ be a pairwise compatible set. Then:

  $$\begin{aligned} &g(\bigsqcup X) = \\ &\quad = h^*(Pr(\bigsqcup X)) \\ &\quad = h^*(\textstyle\bigcup_{x\in X} Pr(x)) \qquad [\text{since } Pr(\bigsqcup X) = \textstyle\bigcup_{x\in X} Pr(x)] \\ &\quad = \textstyle\bigcup_{x\in X} h^*(Pr(x)) \\ &\quad = \textstyle\bigsqcup_{x\in X} g(x) \end{aligned}$$

- *Stable.* Let $d, d' \in D$ with $d \uparrow d'$, then:

  $$\begin{aligned} &g(d \sqcap d') = \\ &\quad = h^*(Pr(d \sqcap d')) \\ &\quad = h^*(Pr(d) \cap Pr(d')) \qquad [\text{since } Pr(d \sqcap d') = Pr(d) \cap Pr(d') \text{ and} \\ &\qquad\qquad\qquad\qquad\quad h \text{ injective on } Pr(d) \cup Pr(d')] \\ &\quad = h^*(Pr(d)) \cap h^*(Pr(d')) \\ &\quad = g(d) \sqcap g(d') \end{aligned}$$

The morphism g defined as above makes the diagram commute. In fact, let $p \in Pr(D)$ $(= \mathscr{P}_a(D))$ and let us use Lemma 3.24 to determine $\mathscr{P}_a(g)(p)$. We have:

$$\begin{aligned} & g(p) - \bigcup g(Pr(p) - \{p\}) = \\ & \quad = h^*(Pr(p)) - \bigcup\{h^*(Pr(p')) \mid p' \in Pr(D),\ p' \sqsubset p\} \\ & \quad = h^*(Pr(p)) - \{h(p'') \mid p'' \in Pr(D),\ p'' \sqsubset p\} \\ & \quad = h^*(Pr(p)) - h^*(Pr(p) - \{p\}) \\ & \quad = \{h(p)\} \quad \text{[since } h \text{ injective on } Pr(p)\text{]} \end{aligned}$$

Therefore, if $h(p)$ is undefined then $\mathscr{P}_a(g)(p) = \bot$ and thus $\epsilon_G(\mathscr{P}_a(g)(p)) = \bot$. If $h(p) = e$ then $\mathscr{P}_a(g)(p) = g(p)[\![e]\!]$ and thus $\epsilon_G(\mathscr{P}_a(g)(p)) = e = h(p)$. Summing up we conclude

$$\epsilon_G \circ \mathscr{P}_a(g) = h.$$

Uniqueness
Let $g' : D \to \mathscr{L}_a(G)$ be another morphism such that

$$\epsilon_G \circ \mathscr{P}_a(g') = h.$$

By Lemma 3.24, for all $p \in Pr(D)$ we have:

$$\mathscr{P}_a(g')(p) = \begin{cases} \bot & \text{if } g'(p) - \bigcup g'(Pr(p) - \{p\}) = \emptyset \\ g'(p)[\![e]\!] & \text{if } g'(p) - \bigcup g'(Pr(p) - \{p\}) = \{e\} \end{cases}$$

Therefore

$$h(p) = \epsilon_G(\mathscr{P}_a(g')(p)) = \begin{cases} \bot & \text{if } g'(p) - \bigcup g'(Pr(p) - \{p\}) = \emptyset \\ e & \text{if } g'(p) - \bigcup g'(Pr(p) - \{p\}) = \{e\} \end{cases} \tag{3.2}$$

Let us show that $g'(p) = g(p)$ for all $p \in Pr(D)$, by induction on $k = |Pr(p)|$ (that is finite, since D is finitary).

($\mathbf{k = 1}$) In this case $g'(p) - \bigcup g'(Pr(p) - \{p\}) = g'(p)$. Thus, by (3.2) above, if $h(p) = \bot$ then $g'(p) = \emptyset = g(p)$, otherwise, $g'(p) = \{h(p)\} = g(p)$.

($\mathbf{k \to k + 1}$) First notice that being g' monotonic, for all $p' \in Pr(p)$ we have $g'(p') \sqsubseteq g'(p)$, thus

$$g'(p) = (g'(p) - (\bigcup g'(Pr(p) - \{p\}))) \cup (\bigcup g'(Pr(p) - \{p\})).$$

By inductive hypothesis, $\bigcup g'(Pr(p) - \{p\}) = \bigcup g(Pr(p) - \{p\})$, thus, reasoning as in the case ($k = 1$) we conclude.

Being g and g' additive, since they coincide on the complete primes of D, which is prime algebraic, they coincide also on the whole domain D. □

3.3 The category of contextual nets

This section introduces a notion of morphism for contextual nets, turning the class of c-nets into a category **CN**. Morphisms are shown to preserve the token game in such a way that they can be thought of as simulations of the source net into the target net.

In the following when considering a c-net N, we implicitly assume that $N = \langle S, T, F, C, m\rangle$. Moreover superscripts and subscripts on the nets names carry over the names of the involved sets, functions and relations (e.g., $N_i = \langle S_i, T_i, F_i, C_i, m_i\rangle$).

A c-net morphism between two nets maps transitions and places of the first net into transitions and multisets of places of the second net, respectively, in such a way that the initial marking as well as the pre- and post-sets of transitions are "preserved". Contexts are preserved in a weak sense.

DEFINITION 3.26 (C-NET MORPHISM)
Let N_0 and N_1 be c-nets. A morphism $h : N_0 \to N_1$ *is a pair $h = \langle h_T, h_S \rangle$, where $h_T : T_0 \to T_1$ is a partial function and $h_S : S_0 \to S_1$ is a multirelation such that*

1. $\mu h_S(m_0) = m_1$
2. *for each $A \in \mu T$,*
   (a) $\mu h_S({}^\bullet A) = {}^\bullet \mu h_T(A)$;
   (b) $\mu h_S(A^\bullet) = \mu h_T(A)^\bullet$;
   (c) $[\![\underline{\mu h_T(A)}]\!] \leq \mu h_S(\underline{A}) \leq \underline{\mu h_T(A)}$.

We denote by **CN** *the category having c-nets as objects and c-net morphisms as arrows.*

Conditions (1), (2.a) and (2.b) are standard for ordinary nets, but condition (2.c), regarding contexts, deserves some comments. It can be explained by recalling that, since in our model a single token can be used as context with multiplicity greater than one, the firing of a transition t can use as context any multiset X satisfying

$$[\![\underline{t}]\!] \leq X \leq \underline{t}.$$

Given any multiset of tokens that can be used as context in the firing of a transition, its image should be a set of tokens that can be used as context by the image of the transition. This can be formalized by requiring that $[\![\underline{\mu h_T(A)}]\!] \leq \mu h_S(X) \leq \underline{\mu h_T(A)}$ for any $X \in \mu S_0$ such that $[\![\underline{A}]\!] \leq X \leq \underline{A}$, which is equivalent to the above condition (2.c). Observe that, in particular, $[\![\underline{\mu h_T(A)}]\!] = [\![\mu h_S(\underline{A})]\!]$.

It is worth remarking that if h_T is undefined on a transition $t_0 \in T_0$, written $h_T(t_0) = \perp$, then, by definition of c-net morphism, the places in the pre-, post-set and context of t_0 are forced to be mapped to the empty set, i.e., $\mu h_S({}^\bullet t + t^\bullet + \underline{t}) = \emptyset$.

A basic result to prove (to check that the definition of morphism is "meaningful") is that the token game is preserved by c-net morphisms. As an immediate consequence morphisms preserve reachable markings.

PROPOSITION 3.27 (MORPHISMS PRESERVE THE TOKEN GAME)
Let N_0 and N_1 be c-nets, and let $h = \langle h_T, h_S \rangle : N_0 \to N_1$ be a morphism. Then for each $M, M' \in \mu S$ and $A \in \mu T$

$$M\,[A\rangle\,M' \quad \Rightarrow \quad \mu h_S(M)\,[\mu h_T(A)\rangle\,\mu h_S(M').$$

Therefore c-net morphisms preserve reachable markings, i.e., if M_0 is a reachable marking in N_0 then $\mu h_S(M_0)$ is reachable in N_1.

PROOF. First notice that $\mu h_T(A)$ is enabled by $\mu h_S(M)$. In fact, since A is enabled by M, we have $M \geq {}^\bullet A + [\![\underline{A}]\!]$. Thus

$$\begin{aligned}
&\mu h_S(M) \\
&\quad \geq \mu h_S({}^\bullet A + [\![\underline{A}]\!]) \\
&\quad = \mu h_S({}^\bullet A) + \mu h_S([\![\underline{A}]\!]) \\
&\quad \geq \mu h_S({}^\bullet A) + [\![\mu h_S(\underline{A})]\!] \\
&\quad = {}^\bullet \mu h_T(A) + [\![\underline{\mu h_T(A)}]\!] \qquad \text{[by def. of c-net morphism]}
\end{aligned}$$

Moreover $\mu h_S(M') = \mu h_S(M) - {}^\bullet \mu h_T(A) + \mu h_T(A)^\bullet$. In fact, $M' = M - {}^\bullet A + A^\bullet$, therefore we have:

$$\begin{aligned}
&\mu h_S(M') \\
&\quad = \mu h_S(M) - \mu h_S({}^\bullet A) + \mu h_S(A^\bullet) \\
&\quad = \mu h_S(M) - {}^\bullet \mu h_T(A) + \mu h_T(A)^\bullet \qquad \text{[by def. of c-net morphism]}
\end{aligned}$$

□

The seminal work by Winskel [Win87a] presents a coreflection between prime event structures and a subclass of P/T nets, namely *safe* nets. In [MMS97] it is shown that essentially the same constructions work for the larger category of "semi-weighted nets" as well (while the generalization to the whole category of P/T nets requires some original technical machinery and allows one to obtain a proper adjunction rather than a coreflection [MMS96]). In the next sections we will relate by a coreflection (asymmetric and prime) event structures and "semi-weighted c-nets".

DEFINITION 3.28 (SEMI-WEIGHTED AND SAFE C-NETS)
A semi-weighted *c-net is a c-net N such that the initial marking m is a set and F_{post} is a relation (i.e., $t^\bullet$ is a set for all $t \in T$). We denote by* **SW-CN** *the full subcategory of* **CN** *having semi-weighted c-nets as objects.*

A semi-weighted c-net is called safe *if also F_{pre} and C are relations (i.e., ${}^\bullet t$ and $\underline{t}$ are sets for all $t \in T$) and each reachable marking is a set. The full subcategory of* **SW-CN** *containing all safe c-nets is denoted by* **S-CN**.

Notice that the condition characterizing safe nets involves the dynamics of the net itself, while the one defining semi-weighted nets is "syntactical" in the sense that it can be checked statically, by looking only at structure of the net.

3.4 Occurrence contextual nets

In the previous section we gave a description of the behaviour of a c-net in a dynamic way, by describing how the token game evolves. Occurrence c-nets are intended to represent, via the unfolding construction, the behaviour of general c-nets in a more static way, by expressing the events (firing of transitions) which can appear in a computation and the dependency relations between them. Occurrence c-nets will be defined as safe c-nets such that the dependency relations between transitions satisfy suitable acyclicity and well-foundedness requirements. While for traditional occurrence nets one has to take into account the causal dependency and the (symmetric)

conflict relations, by the presence of contexts, we have to consider an asymmetric conflict (or weak dependency) relation as well. The conflict relation, as already seen in the more abstract setting of AES's, turns out to be a derived relation.

3.4.1 Dependency relations on transitions

Causal dependency is defined as for traditional safe nets, with an additional clause stating that transition t causes t' if it generates a token in a context place of t'.

DEFINITION 3.29 (CAUSAL DEPENDENCY)
Let N be a safe c-net. The causal dependency relation $<_N$ *is the transitive closure of the relation $\prec$ defined by:*

1. *if $s \in {}^\bullet t$ then $s \prec t$;*
2. *if $s \in t^\bullet$ then $t \prec s$;*
3. *if $t^\bullet \cap \underline{t'} \neq \emptyset$ then $t \prec t'$.*

Given a place or transition $x \in S \cup T$, we denote by $\lfloor x \rfloor$ the set of causes *of x in T, defined as $\lfloor x \rfloor = \{t \in T \mid t \leq_N x\} \subseteq T$, where $\leq_N$ is the reflexive closure of $<_N$.*

DEFINITION 3.30 (ASYMMETRIC CONFLICT)
Let N be a safe c-net. The strict asymmetric conflict relation $\rightsquigarrow_N$ *is defined as*

$$t \rightsquigarrow_N t' \qquad \textit{iff} \qquad \underline{t} \cap {}^\bullet t' \neq \emptyset \;\; \textit{or} \;\; (t \neq t' \;\wedge\; {}^\bullet t \cap {}^\bullet t' \neq \emptyset).$$

The asymmetric conflict relation $\nearrow_N$ *is the union of the strict asymmetric conflict and causal dependency relations:*

$$t \nearrow_N t' \qquad \textit{iff} \qquad t <_N t' \;\textit{or}\; t \rightsquigarrow_N t'.$$

In our informal interpretation, if $t \nearrow_N t'$ then t must precede t' in each computation in which both fire or, equivalently, t' prevents t to be fired, namely

$$\mathit{Fired}_C(t) \wedge \mathit{Fired}_C(t') \;\Rightarrow\; \mathit{prec}_C(t, t') \qquad (\dagger)$$

As noticed in the INTRODUCTION, in an acyclic safe c-net where any transition is enabled at most once in each computation, condition (†) is surely satisfied when the same place s appears in the context of t and in the pre-set of t'. But (†) is trivially true (with t and t' in interchangeable roles) when t and t' have a common precondition, since they never fire in the same computation. This is apparently a little tricky but corresponds to the clear intuition that a (usual) symmetric (direct) conflict leads to asymmetric conflict in both directions. Furthermore, since, as noticed for the general model of AES, (†) is weaker than the condition that expresses causality, the

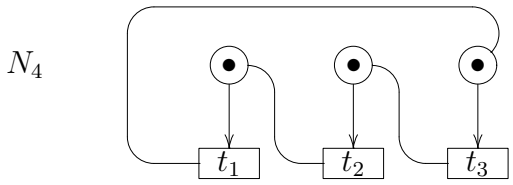


Figure 3.4: An occurrence c-net with a cycle of asymmetric conflict.

condition (†) is satisfied when t causes (in the usual sense) t'.[4] For technical reasons it is convenient to distinguish the strict asymmetric conflict from causality.

In the sequel, when the net N is clear from the context, the subscripts in the relations $\leq_N$ and $\nearrow_N$ will be omitted.

The c-net N_4 in Figure 3.4 shows that, as expected, also in this setting the relation $\nearrow$ is not transitive. In fact we have $t_1 \nearrow t_3 \nearrow t_2 \nearrow t_1$, but, for instance, it is not true that $t_1 \nearrow t_2$.

An occurrence c-net is a safe c-net that exhibits an acyclic behaviour and such that each transition can fire in some computation of the net. Furthermore, to allow for the interpretation of the places as token occurrences, each place has at most one transition in its pre-set.

Definition 3.31 (occurrence c-nets)
An occurrence c-net *is a safe c-net N such that*

1. *each place $s \in S$ is in the post-set of at most one transition, i.e., $|{}^\bullet s| \leq 1$;*
2. *the reflexive closure $\leq_N$ of the* causal relation $<_N$ *is a partial order such that $\lfloor t \rfloor$ is finite for any $t \in T$;*
3. *the* initial marking *m coincides with the set of minimal places with respect to $\leq_N$, i.e., $m = \{s \in S \mid {}^\bullet s = \emptyset\}$;*
4. *$(\nearrow_N)_{\lfloor t \rfloor}$ is acyclic for all transitions $t \in T$.*

With **O-CN** *we denote the full subcategory of* **S-CN** *having occurrence c-nets as objects.*

Conditions (1)-(3) are the same as for ordinary occurrence nets. Condition (4) corresponds to the requirement of irreflexivity for the conflict relation in ordinary occurrence nets. In fact, if a transition t has a $\nearrow_N$ cycle in its causes then it can never fire, since in an occurrence c-net N, the order in which transitions appear in a firing sequence must be compatible with the transitive closure of the (restriction to the transitions in the sequence of the) asymmetric conflict relation.

As anticipated the asymmetric conflict relation induces a symmetric conflict relation (on sets of transitions) defined in the following way:

[4] This is the origin of the weak causality interpretation of $\nearrow$.

DEFINITION 3.32 (CONFLICT)
Let N be a c-net. The conflict *relation $\# \subseteq \mathbf{2}^T_{fin}$ associated to N is defined as:*

$$\frac{t_0 \nearrow t_1 \nearrow \ldots \nearrow t_n \nearrow t_0}{\#\{t_0, t_1, \ldots, t_n\}} \qquad \frac{\#(A \cup \{t\}) \quad t \leq t'}{\#(A \cup \{t'\})}$$

where A denotes a generic finite subset of T. As for AES*'s, we use the infix notation $t\#t'$ for $\#\{t, t'\}$.*

For instance, referring to Figure 3.4, we have $\#\{t_1, t_2, t_3\}$, but not $\#\{t_i, t_j\}$ for any $i, j \in \{1, 2, 3\}$. Notice that, by definition, the binary conflict relation $\#$ is symmetric. Moreover in an occurrence c-net $\#$ is irreflexive by the fourth condition of Definition 3.31.

Finally, observe that irreflexivity of the asymmetric conflict relation $\nearrow_N$ in an occurrence c-net N implies that the pre-set, the post-set and the context of any transition t in N are disjoint (any possible intersection would lead to $t \nearrow_N t$).

3.4.2 Concurrency and reachability

As for ordinary occurrence nets, a set of places M is called concurrent if there is a reachable marking in which all the places of M contain a token. Differently from the classical case, for the presence of contexts some places that a transition needs to be fired (contexts) can be concurrent with the places it produces. However, the concurrency of a set of places can still be checked locally by looking only at the causes of such places and thus can be expressed via a "syntactical" condition. This subsection introduces such condition and then shows that it correctly formalizes the intuitive idea of concurrency.

DEFINITION 3.33 (CONCURRENCY RELATION)
Let N be an occurrence c-net. A set of places $M \subseteq S$ is called concurrent*, written $conc(M)$, if*

1. *$\forall s, s' \in M.\ \neg(s < s')$;*
2. *$\lfloor M \rfloor$ is finite, where $\lfloor M \rfloor = \bigcup\{\lfloor s \rfloor \mid s \in M\}$;*
3. *$\nearrow_{\lfloor M \rfloor}$ is acyclic (and thus well-founded, since $\lfloor M \rfloor$ is finite).*

In particular, for each transition t in an occurrence c-net, the set of places consisting of its pre-set and context is concurrent.

PROPOSITION 3.34
For any transition t of an occurrence c-net, $conc({}^\bullet t + \underline{t})$.

PROOF. Since $\lfloor {}^\bullet t + \underline{t} \rfloor \cup \{t\} = \lfloor t \rfloor$, by definition of occurrence c-net, conditions (2) and (3) of the definition of concurrency are satisfied. As for the first condition, suppose that $s < s'$ for $s, s' \in {}^\bullet t + \underline{t}$. Then there is a transition t' such that $s \in {}^\bullet t'$ and $t' < s'$. Now, since $t' < s'$ and $s' \in {}^\bullet t + \underline{t}$ we

have $t' < t$ and, since $s \in {}^\bullet t + \underline{t}$ and $s \in {}^\bullet t'$, we have also $t \nearrow t'$. Therefore $t' < t \nearrow t'$ is a $\nearrow$-cycle in $\lfloor t \rfloor$, contradicting the definition of occurrence c-net. Thus, also condition (1) is satisfied. □

The next two lemmata show that given a concurrent set of places, we can interpret it as the result of a computation and perform a backward or forward step in such a computation, still obtaining a concurrent set.

LEMMA 3.35 (BACKWARD STEPS PRESERVE CONCURRENCY)
Let N be an occurrence c-net and let $M \subseteq S$ be a set of places. If $conc(M)$ and $t \in \lfloor M \rfloor$ is maximal with respect to $(\nearrow_{\lfloor M \rfloor})^*$ then

1. $\exists s_t \in S.\ s_t \in t^\bullet \cap M$;
2. $conc(M - t^\bullet + {}^\bullet t)$.

PROOF.

1. Since $t \in \lfloor M \rfloor$, there is $s_t \in M$ and $t' \in T$ such that $t \leq t'$ and $s_t \in t'^\bullet$. But recalling that $\nearrow$ implies $<$, by using maximality of t, we can conclude that $t = t'$.

2. Let $M' = M - t^\bullet + {}^\bullet t$. Clearly $\lfloor M' \rfloor = \lfloor M \rfloor - \{t\}$ and thus $\lfloor M' \rfloor$ is finite and $\nearrow_{\lfloor M' \rfloor}$ is acyclic.

   Moreover, we have to show that there are no causal dependent (distinct) places in M'. Since $conc(M - t^\bullet)$, by hypothesis, and $conc({}^\bullet t)$, by Proposition 3.34, the only problematic case could be $s \in M - t^\bullet$ and $s' \in {}^\bullet t$. But

   - if $s < s'$ then, by transitivity of $<$, we have $s < s_t$;
   - if $s' < s$ then there is a transition t' such that $s' \in {}^\bullet t'$ and $t' \leq s$. Since $s' \in {}^\bullet t \cap {}^\bullet t'$, we have that $t \nearrow t' \nearrow t$ is a $\nearrow$-cycle in $\lfloor M \rfloor$.

   In both cases we reach a contradiction with the hypothesis $conc(M)$. □

LEMMA 3.36 (FORWARD STEPS PRESERVE CONCURRENCY)
Let N be an occurrence c-net and let $M \subseteq S$ be a set of places. If $conc(M)$ and $M\,[t\rangle\, M'$ then $conc(M')$.

PROOF. The transition t is enabled by M, i.e., ${}^\bullet t + \underline{t} \subseteq M$ and thus $\neg(t \nearrow t')$ for all $t' \in \lfloor M \rfloor$. In fact, let $t' \in \lfloor M \rfloor$, that is $t' < s'$ for some $s' \in M$. Clearly it can not be $t \rightsquigarrow t'$, otherwise, if $s \in {}^\bullet t' \cap ({}^\bullet t \cup \underline{t}) \subseteq M$ then $s < s'$, contradicting the hypothesis $conc(M)$. In the same way, if $t < t'$ then given any $s \in {}^\bullet t (\subseteq M)$, we would have $s < s'$.

Therefore, since $\lfloor M' \rfloor \subseteq \lfloor M \rfloor \cup \{t\}$ (the strict inclusion holds when $t^\bullet = \emptyset$) and, by hypothesis, $\nearrow_{\lfloor M \rfloor}$ is acyclic, we can conclude that $\nearrow_{\lfloor M' \rfloor}$ is acyclic. Moreover, being $\lfloor M \rfloor$ finite, also $\lfloor M' \rfloor$ is finite.

Finally, we have to show that there are no (distinct) causal dependent places in M'. Since $conc(M - {}^\bullet t)$ and $conc(t^\bullet)$ the only problematic case could be $s \in M - {}^\bullet t$ and $s' \in t^\bullet$. But

- if $s < s'$ then $s < s''$ for some $s'' \in {}^\bullet t \cup \underline{t}$;
- if $s' < s$ then, for $s'' \in {}^\bullet t$, by transitivity of $<$, $s'' < s$.

In both cases we reach a contradiction with the hypothesis $conc(M)$. □

It is now quite easy to conclude that, as mentioned before, the concurrent sets of places of a c-net indeed coincide with the (subsets of) reachable markings.

PROPOSITION 3.37 (CONCURRENCY AND REACHABILITY)
Let N be an occurrence c-net and let $M \subseteq S$ be a set of places. Then

$$conc(M) \qquad \textit{iff} \qquad M \subseteq M' \textit{ for some reachable marking } M'.$$

PROOF.

($\Rightarrow$) By definition of the concurrency relation, $\lfloor M \rfloor$ is finite. Moreover $\nearrow_{\lfloor M \rfloor}$ is acyclic and therefore there is an enumeration $t^{(1)}, \ldots, t^{(k)}$ of the transitions in $\lfloor M \rfloor$ compatible with $(\nearrow_{\lfloor M \rfloor})^+$. Let us show by induction on $k = |\lfloor M \rfloor|$ that

$$m = M^{(0)} \, [t^{(1)}\rangle \, M^{(1)} \, [t^{(2)}\rangle \, M^{(2)} \ldots [t^{(k)}\rangle \, M^{(k)} \supseteq M.$$

($\mathbf{k = 0}$) In this case simply $m \supseteq M$ and thus $m = M^{(0)} \supseteq M$.

($\mathbf{k > 0}$) By construction, $t^{(k)}$ is maximal in $\lfloor M \rfloor$ with respect to $(\nearrow_{\lfloor M \rfloor})^+$. Thus, by Lemma 3.35, if we define $M'' = M - t^{(k)\bullet} + {}^\bullet t^{(k)}$, we have $conc(M'')$ and $\lfloor M'' \rfloor = \{t^{(1)}, \ldots, t^{(k-1)}\}$. Therefore, by inductive hypothesis, there is a firing sequence

$$m \, [t^{(1)}\rangle \, M^{(1)} \ldots [t^{(k-1)}\rangle \, M^{(k-1)} \supseteq M''. \tag{3.3}$$

Now, by construction, ${}^\bullet t^{(k)} \subseteq M''$. Moreover also $\underline{t^{(k)}} \subseteq M''$. In fact, if $s \in \underline{t^{(k)}}$ then $s \in m$ or $s \in t^{(h)\bullet}$ for some $h < k$. Thus a token in s is generated in the firing sequence (3.3), and no transition $t^{(l)}$ can consume this token, otherwise $t^{(k)} \nearrow t^{(l)}$, contradicting the maximality of $t^{(k)}$. Finally, by definition of occurrence c-net, ${}^\bullet t^{(k)} \cap \underline{t^{(k)}} = \emptyset$, being $\nearrow$ irreflexive. Therefore $t^{(k)}$ is enabled in M'' so that we can extend the firing sequence (3.3) to

$$m \, [t^{(1)}\rangle \, M^{(1)} \ldots [t^{(k-1)}\rangle \, M^{(k-1)} \, [t^{(k)}\rangle \, M^{(k)},$$

where $M^{(k)} = M^{(k-1)} - {}^\bullet t^{(k)} + t^{(k)\bullet} \supseteq M'' - {}^\bullet t^{(k)} + t^{(k)\bullet} = M$.

($\Leftarrow$) Let us suppose that there exists a firing sequence

$$m \, [t^{(1)}\rangle \, M^{(1)} \, [t^{(2)}\rangle \, M^{(2)} \ldots [t^{(k)}\rangle \, M^{(k)} \supseteq M.$$

and let us prove that $conc(M^{(k)})$ (and thus $conc(M)$).

If ($k = 0$), then $M \subseteq m$ and clearly $conc(m)$. If $k > 0$ then an inductive reasoning that uses Lemma 3.36 allows one to conclude. □

As an immediate corollary we obtain that each transition of an occurrence c-net is firable in some computation of the net.

COROLLARY 3.38
For any transition t of an occurrence c-net N there is a reachable marking M of N which enables t.

PROOF. By Proposition 3.34, $conc({}^\bullet t + \underline{t})$ and thus, by Proposition 3.37, we can find a reachable marking M of N, such that $M \supseteq {}^\bullet t + \underline{t}$, enabling t. □

3.4.3 Morphisms on occurrence c-nets

This subsection shows some properties of c-net morphisms between occurrence c-nets that will be useful in the sequel. We start with a characterization of such morphisms.

Lemma 3.39 (occurrence c-nets morphisms)
Let N_0 and N_1 be occurrence c-nets and let $h = \langle h_T, h_S\rangle : N_0 \to N_1$ be a morphism. Then h_S is a relation *and*

- $\forall s_1 \in m_1.\ \exists! s_0 \in m_0.\ h_S(s_0, s_1)$;
- *for each $t_0 \in T_0$, if $h_T(t_0) = t_1$ then*
  - $\forall s_1 \in {}^\bullet t_1.\ \exists! s_0 \in {}^\bullet t_0.\ h_S(s_0, s_1)$;
  - $\forall s_1 \in \underline{t_1}.\ \exists! s_0 \in \underline{t_0}.\ h_S(s_0, s_1)$;
  - $\forall s_1 \in t_1{}^\bullet.\ \exists! s_0 \in t_0{}^\bullet.\ h_S(s_0, s_1)$;

Moreover given any $s_0 \in S_0$, $s_1 \in S_1$, $t_1 \in T_1$:

- $s_1 \in m_1 \ \wedge\ h_S(s_0, s_1) \quad \Rightarrow \quad s_0 \in m_0$;
- $s_1 \in t_1{}^\bullet \wedge\ h_S(s_0, s_1) \quad \Rightarrow \quad \exists! t_0 \in T_0.\ (s_0 \in t_0{}^\bullet \ \wedge\ h_T(t_0) = t_1)$.

Proof. The result is easily proved by using the structural properties of occurrence c-nets. We treat just the first point. Let $s_1 \in m_1$. Since it must be $\mu h_S(m_0) = m_1$, there exists $s_0 \in m_0$ such that $h_S(s_0, s_1)$. Such s_0 must be unique, since otherwise the initial marking of N_1 should be a proper multiset, rather than a set, contradicting the definition of occurrence c-net. □

As an easy consequence of the results in the previous subsection, c-net morphisms preserve the concurrency relation.

Corollary 3.40 (morphisms preserve concurrency)
Let N_0 and N_1 be occurrence c-nets and let $h : N_0 \to N_1$ be a morphism. Given $M_0 \subseteq S_0$, if $conc(M_0)$ then $\mu h_S(M_0)$ is a set and $conc(\mu h_S(M_0))$.

Proof. Let $M_0 \subseteq S_0$, with $conc(M_0)$. Then, by Proposition 3.37, there exists a firing sequence in N_0:

$$m_0\,[t^{(1)}\rangle\, M^{(1)} \ldots [t^{(n)}\rangle\, M^{(n)} \supseteq M_0.$$

By Proposition 3.27, morphisms preserve the token game and thus

$$m_1 = \mu h_S(m_0)\,[h_T(t^{(1)})\rangle\, \mu h_S(M^{(1)}) \ldots [h_T(t^{(n)})\rangle\, \mu h_S(M^{(n)}) \supseteq \mu h_S(M_0).$$

is a firing sequence in N_1. Hence $\mu h_S(M_0)$ is a set and, by Proposition 3.37, $conc(\mu h_S(M_0))$. □

Notice that the corollary implicitly states that morphisms are "injective" on concurrent sets, in the sense that if $conc(M)$ and $s \neq s'$ are in M, then $\mu h_S(s)$, $\mu h_S(s')$ are sets and $\mu h_S(s) \cap \mu h_S(s') = \emptyset$ (otherwise $\mu_S(M)$ would be a proper multiset).

The next lemmata show that, more generally, morphisms preserve the "amount of concurrency". Let the symbol $\prec$ denote the immediate causal dependency between transitions, namely $t \prec t'$ if $t < t'$ and there does not exists t'' such that $t < t'' < t'$. First we prove that c-net morphisms reflect $\prec$-chains. Then, by means of some other technical results, we can conclude that c-net morphisms reflect causality and conflict, while asymmetric conflict is reflected or becomes conflict.

LEMMA 3.41 (MORPHISMS REFLECT $\prec$-CHAINS)
Let N_0 and N_1 be occurrence c-nets, let $h : N_0 \to N_1$ be a morphism and let $t_1^{(0)} \prec t_1^{(1)} \prec \ldots \prec t_1^{(n)}$ be a chain of transitions in T_1 such that $t_1^{(n)} = h_T(t_0^{(n)})$. Then there exists a chain $t_0^{(0)} \prec t_0^{(1)} \prec \ldots \prec t_0^{(n)}$ in T_0 such that $t_1^{(i)} = h_T(t_0^{(i)})$ for all $i \in \{0, \ldots, n\}$.

PROOF. We proceed by induction on n:

($\mathbf{n = 0}$) Obvious.

($\mathbf{n > 0}$) Let $t_1^{(0)} \prec t_1^{(1)} \prec \ldots \prec t_1^{(n)} = h_T(t_0^{(n)})$. By inductive hypothesis (applied to the final part of the chain) there is a chain $t_0^{(1)} \prec \ldots \prec t_0^{(n)}$ such that $t_1^{(i)} = h_T(t_0^{(i)})$ for $i \in \{1, \ldots, n\}$.

Moreover, since $t_1^{(0)} \prec t_1^{(1)} = h_T(t_0^{(1)})$, two cases arise:

- $t_1^{(0)\bullet} \cap \underline{t_1^{(1)}} \neq \emptyset$.
  Let $s_1 \in t_1^{(0)\bullet} \cap \underline{t_1^{(1)}}$. Since $t_1^{(1)} = h_T(t_0^{(1)})$ there is $s_0 \in \underline{t_0^{(1)}}$ such that $f_S(s_0, s_1)$. Moreover, by Lemma 3.39, from $s_1 \in t_1^{(0)\bullet}$ we have that that $\exists! t_0^{(0)} \in T_0$ such that $h_T(t_0^{(0)}) = t_1^{(0)}$ and $s_0 \in t_0^{(0)\bullet}$. Therefore $t_0^{(0)} \prec t_0^{(1)}$ is the transition that completes the chain.

- $t_1^{(0)\bullet} \cap {}^\bullet t_1^{(1)} \neq \emptyset$.
  Analogous to the previous case.

□

LEMMA 3.42
Let N_0 and N_1 be occurrence c-nets and let $h : N_0 \to N_1$ be a morphism. Then, for all $t_0, t_0' \in T_0$, with $h_T(t_0) \neq \bot \neq h_T(t_0')$,

1. $(h_T(t_0) = h_T(t_0')) \ \wedge \ (t_0 \neq t_0') \quad \Rightarrow \quad t_0 \#_0 t_0'$;
2. $h_T(t_0) \leq_1 h_T(t_0') \quad \Rightarrow \quad \exists! t_0'' \in T_0.\ (t_0'' \leq_0 t_0' \ \wedge \ h_T(t_0'') = h_T(t_0))$;
3. (a) $h_T(t_0) \rightsquigarrow_1 h_T(t_0') \quad \Rightarrow \quad (t_0 \rightsquigarrow_0 t_0') \ \vee \ (t_0 \#_0 t_0')$;
   (b) $h_T(t_0) <_1 h_T(t_0') \quad \Rightarrow \quad (t_0 <_0 t_0') \ \vee \ (t_0 \#_0 t_0')$;

PROOF.

1. Let $h_T(t_0) = h_T(t_0')$ and $t_0 \neq t_0'$. Consider a chain of transitions $t_1^{(0)} \prec \ldots \prec t_1^{(k)} = h_T(t_0)$ such that ${}^\bullet t_1^{(0)} \subseteq m_1$ and $t_1^{(i)\bullet} \cap {}^\bullet t_1^{(i+1)} \neq \emptyset$ for all $i \in \{0, \ldots, k-1\}$ (the existence of such a finite chain is an immediate consequence of the definition of occurrence c-net). By Lemma 3.41, we can find in T_0 two $\prec$-chains of transitions,

$$t_0^{(0)} \prec \ldots \prec t_0^{(k)} \qquad \text{and} \qquad t_0'^{(0)} \prec \ldots \prec t_0'^{(k)},$$

such that, $h_T(t_0^{(i)}) = h_T(t_0'^{(i)}) = t_1^{(i)}$, for all $i \in \{1, \ldots, k\}$ and $t_0 = t_0^{(k)}$, $t_0' = t_0'^{(k)}$.

Let j be the least index such that $t_0^{(j)} \neq t_0'^{(j)}$. If $j = 0$ (and thus ${}^\bullet t_1^{(j)} \subseteq m_1$) consider a generic $s_1 \in {}^\bullet t_1^{(0)}$. By definition of morphism, there are $s_0 \in {}^\bullet t_0^{(0)}$ and $s_0' \in {}^\bullet t_0'^{(0)}$ such that $h_S(s_0, s_1)$ and $h_S(s_0', s_1)$. By Lemma 3.39, since $s_1 \in m_1$, also s_0 and s_0' are in the initial marking and thus $s_0 = s_0'$. Hence $t_0^{(0)} \nearrow_0 t_0'^{(0)} \nearrow_0 t_0^{(0)}$, thus $t_0^{(0)} \#_0^m t_0'^{(0)}$ and therefore, by definition of $\#$, $t_0 \#_0 t_0'$. If $j > 0$, then considering $s_1 \in t_1^{(j-1)\bullet} \cap {}^\bullet t_1^{(j)}$, the same reasoning applies.

2. Existence easily follows from Lemma 3.41. As for uniqueness, let $t_0''' \leq_0 t_0'$ and $h_T(t_0''') = h_T(t_0)$. If $t_0''' \neq t_0''$ then, by point (1) $t_0''' \#_0 t_0''$ and therefore $t_0' \#_0 t_0'$, contradicting the definition of occurrence c-net.

3. (a) Let $h_T(t_0) \rightsquigarrow_1 h_T(t_0')$. Then there is a place $s_1 \in (\underline{h_T(t_0)} \cup {}^\bullet h_T(t_0)) \cap {}^\bullet h_T(t_0')$. Thus there are $s_0 \in (\underline{t_0} \cup {}^\bullet t_0)$ such that $h_S(s_0, s_1)$ and $s_0' \in \overline{{}^\bullet t_0'}$ such that $h_S(s_0', s_1)$. If s_1 is in the initial marking then $s_0 = s_0'$ and thus $t_0 \rightsquigarrow_1 t_0'$. Otherwise s_0 and s_0' are in the post-sets of two transitions $t_0^{(0)}$ and $t_0'^{(0)}$, which are mapped to the same transition in N_1 (the transition which has s_1 in its post-set). By point (1), $t_0^{(0)}$ and $t_0'^{(0)}$ are identical or in conflict: in the first case $s_0 = s_0'$ and thus $t_0 \rightsquigarrow_0 t_0'$, while in the second case $t_0 \#_0 t_0'$.

   (b) Let $h_T(t_0) <_1 h_T(t_0')$. By Lemma 3.41, there exists $t_0'' \in T_0$ such that $t_0'' <_0 t_0'$ and $h_T(t_0'') = h_T(t_0)$. It follows from point (1) that either $t_0'' = t_0$ and thus $t_0 <_0 t_0'$, or $t_0'' \#_0 t_0$ and thus $t_0 \#_0 t_0'$. □

COROLLARY 3.43
Let N_0 and N_1 be occurrence c-nets and let $h : N_0 \to N_1$ be a morphism. Then, for all $t_0, t_0' \in T_0$ with $h_T(t_0) \neq \bot \neq h_T(t_0')$,

1. $\lfloor h_T(t_0) \rfloor \subseteq h_T(\lfloor t_0 \rfloor)$;
2. $(h_T(t_0) = h_T(t_0')) \ \wedge \ (t_0 \neq t_0') \quad \Rightarrow \quad t_0 \#_0 t_0'$;
3. $h_T(t_0) \nearrow_1 h_T(t_0') \quad \Rightarrow \quad (t_0 \nearrow_0 t_0') \ \vee \ (t_0 \#_0 t_0')$;
4. $\#h_T(A) \quad \Rightarrow \quad \#A'$, *for some* $A' \subseteq A$.

It is worth observing that, since the asymmetric conflict relation defined for an occurrence c-net does not satisfy the saturation condition required for AES's (see Definition 3.3) asymmetric conflict is not necessarily reflected by a c-net morphism, but it can also become conflict.

3.5 Unfolding: from semi-weighted to occurrence contextual nets

This section shows how, given a semi-weighted c-net N, an *unfolding* construction allows us to obtain an occurrence c-net $\mathscr{U}_a(N)$ that describes the behaviour of N. As for traditional nets, each transition in $\mathscr{U}_a(N)$ represents an instance of a precise firing of a transition in N, and places in $\mathscr{U}_a(N)$ represent occurrences of tokens in the places of N. Each item (place or transition) of the unfolding is mapped to the corresponding item of the original net by a c-net morphism $f_N : \mathscr{U}_a(N) \to N$, called the folding morphism. The unfolding operation can be extended to a functor $\mathscr{U}_a : \mathbf{SW\text{-}CN} \to \mathbf{O\text{-}CN}$ that is right adjoint to the inclusion functor $\mathscr{I}_O : \mathbf{O\text{-}CN} \to \mathbf{SW\text{-}CN}$ and thus establishes a coreflection between **SW-CN** and **O-CN**.

We first introduce some technical notions. We say that a c-net N_0 is a *subnet* of N_1, written $N_0 \trianglelefteq N_1$, if $S_0 \subseteq S_1$, $T_0 \subseteq T_1$ and the inclusion $\langle i_T, i_S \rangle$ (with $i_T(t) = t$, for all $t \in T_0$, and $i_S(s, s) = 1$, for all $s \in S_0$) is a c-net morphism. In words, $N_0 \trianglelefteq N_1$ if N_0 coincides with an initial segment of N_1. In the following it will be useful to consider the subnets of an occurrence c-net obtained by truncating the original net at a given "causal depth", where the notion of depth is defined in the natural way.

DEFINITION 3.44 (DEPTH)
Let N be an occurrence c-net. The function depth $: S \cup T \to \mathbb{N}$ is defined inductively as follows:

$$\begin{array}{ll} \mathit{depth}(s) = 0 & \text{for } s \in m; \\ \mathit{depth}(t) = \max\{\mathit{depth}(s) \mid s \in {}^\bullet t \cup \underline{t}\} + 1 & \text{for } t \in T; \\ \mathit{depth}(s) = \mathit{depth}(t) & \text{for } s \in t^\bullet. \end{array}$$

It is not difficult to prove that *depth* is a well-defined total function, since infinite descending chains of causality are disallowed in occurrence c-nets. Moreover, given an occurrence c-net N, the net containing only the items of *depth* less or equal to k, denoted by $N^{[k]}$, is a well-defined occurrence c-net and it is a subnet of N. The following simple result holds:

PROPOSITION 3.45 (TRUNCATION)
An occurrence c-net N is the (componentwise) union of its subnets $N^{[k]}$, of depth k.

The unfolding of a semi-weighted c-net N can be constructed inductively by starting from the initial marking of N, and then by adding, at each step, an instance of each transition of N enabled by (the image of) a concurrent subset of places in the partial unfolding currently generated. For technical reasons we prefer to give an axiomatic definition.

DEFINITION 3.46 (UNFOLDING)
Let N be a semi-weighted c-net. The unfolding $\mathscr{U}_a(N) = \langle S', T', F', C', m' \rangle$ of the net N and the folding morphism $f_N = \langle f_T, f_S \rangle : \mathscr{U}_a(N) \to N$ *are the unique occurrence c-net and c-net morphism satisfying the following equations.*

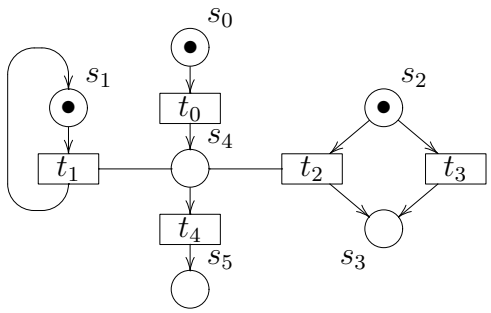
$$
\begin{array}{ll}
m' = & \{\langle \emptyset, s \rangle \mid s \in m\} \\
S' = & m' \cup \{\langle t', s \rangle \mid t' = \langle M_p, M_c, t \rangle \in T' \ \wedge \ s \in t^\bullet\} \\
T' = & \{\langle M_p, M_c, t \rangle \mid M_p, M_c \subseteq S' \ \wedge \ M_p \cap M_c = \emptyset \ \wedge \ conc(M_p \cup M_c) \ \wedge \\
& \quad t \in T \ \wedge \ \mu f_S(M_p) = {}^\bullet t \ \wedge \ [\![\underline{t}]\!] \leq \mu f_S(M_c) \leq \underline{t}\}
\end{array}
$$

$$
\begin{array}{llll}
F'_{pre}(t', s') & \text{iff} & t' = \langle M_p, M_c, t \rangle \ \wedge \ s' \in M_p & (t \in T) \\
C'(t', s') & \text{iff} & t' = \langle M_p, M_c, t \rangle \ \wedge \ s' \in M_c & (t \in T) \\
F'_{post}(t', s') & \text{iff} & s' = \langle t', s \rangle & (s \in S) \\
f_T(t') = t & \text{iff} & t' = \langle M_p, M_c, t \rangle & \\
f_S(s', s) & \text{iff} & s' = \langle x, s \rangle & (x \in T' \cup \{\emptyset\})
\end{array}
$$

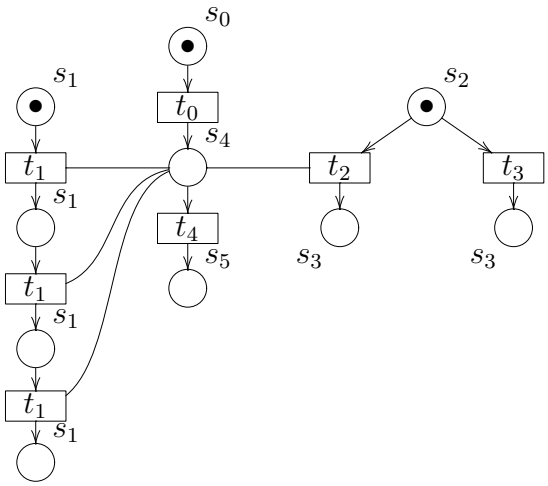
The existence of the unfolding can be proved by explicitly giving its inductive definition. Uniqueness follows from the fact that each item in a occurrence c-net has a finite depth.

Places and transitions in the unfolding of a c-net represent respectively tokens and firing of transitions in the original net. Each place in the unfolding is a pair recording the "history" of the token and the corresponding place in the original net. Each transition is a triple recording the pre-set and context used in the firing, and the corresponding transition in the original net. A new place with empty history $\langle \emptyset, s \rangle$ is generated for each place s in the initial marking of N. Moreover a new transition $t' = \langle M_p, M_c, t \rangle$ is inserted in the unfolding whenever we can find a concurrent set of places that corresponds, in the original net, to a marking that enables t (M_p is mapped to the pre-set and M_c to the context of t). For each place s in the post-set of such t, a new place $\langle t', s \rangle$ is generated, belonging to the post-set of t'. The folding morphism f maps each place (transition) of the unfolding to the corresponding place (transition) in the original net. Figure 3.5 shows a c-net N and the initial part of its unfolding (formally, it is the subnet of the unfolding of depth 3, namely $\mathscr{U}_a(N)^{[3]}$). The folding morphism is represented by labelling the items of the unfolding with the names of the corresponding items of N.

Occurrence c-nets are particular semi-weighted c-nets, thus we can consider the inclusion functor $\mathscr{I}_O$: **O-CN** $\to$ **SW-CN** that acts as identity on objects and morphisms. We show now that the unfolding of a c-net $\mathscr{U}_a(N)$ and the folding morphism f_N are cofree over N. Therefore $\mathscr{U}_a$ extends to a functor that is right adjoint to $\mathscr{I}_O$ and thus establishes a coreflection between **SW-CN** and **O-CN**.



N

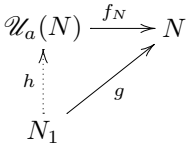


$\mathscr{U}_a(N)^{[3]}$

Figure 3.5: A c-net and (part of) its unfolding.

THEOREM 3.47 (COREFLECTION BETWEEN **SW-CN** AND **O-CN**)
$\mathscr{I}_O \dashv \mathscr{U}_a$

PROOF. Let N be a semi-weighted c-net, let $\mathscr{U}_a(N) = \langle S', T', F', C', m' \rangle$ be its unfolding and let $f_N : \mathscr{U}_a(N) \to N$ be the folding morphism as in Definition 3.46. We have to show that for any occurrence c-net N_1 and for any morphism $g : N_1 \to N$ there exists a unique morphism $h : N_1 \to \mathscr{U}_a(N)$ such that the following diagram commutes:



Existence
We define a sequence of morphisms $h^{[k]} : {N_1}^{[k]} \to \mathscr{U}_a(N)$ such that, for any k,

$$h^{[k]} \subseteq h^{[k+1]} \qquad \text{and} \qquad f_N \circ h^{[k]} = g_{|{N_1}^{[k]}},$$

then the morphism h we are looking for will be $h = \bigcup_k h^{[k]}$. We give an inductive definition:

($\mathbf{k = 0}$) The c-net ${N_1}^{[0]}$ consists only of the initial marking of N_1 with no transitions, i.e., ${N_1}^{[0]} = \langle m_1, \emptyset, \emptyset, \emptyset, m_1 \rangle$. Therefore $h^{[0]}$ has to be defined:

$$\begin{array}{l} {h_T}^{[0]} = \emptyset, \\ {h_S}^{[0]}(s_1, \langle \emptyset, s \rangle) \equiv g_S(s_1, s) \qquad \text{for all } s_1 \in {S_1}^{[0]} = m_1 \text{ and } s \in S. \end{array}$$

($\mathbf{k \to k+1}$) The morphism $h^{[k+1]}$ extends $h^{[k]}$ on items with depth equal to $k+1$ as follows. Let $t_1 \in T^{[k+1]}$ with $depth(t_1) = k+1$. By definition of depth, $depth(s) \leq k$ for all $s \in {}^\bullet t_1 \cup \underline{t_1}$ and thus $h^{[k]}$ is defined on the pre-set and on the context of t_1. We must define h_T on t_1 and h_S on its post-set. Two cases arise:

- If $g_T(t_1) = \bot$ then necessarily

$$\begin{array}{l} {h_T}^{[k+1]}(t_1) = \bot \\ {h_S}^{[k+1]}(s_1, s') = 0 \qquad \text{for all } s_1 \in {t_1}^\bullet \text{ and } s' \in S'. \end{array}$$

- If $g_T(t_1) = t$ then consider the sets

$$M_p = \mu {h_S}^{[k]}({}^\bullet t_1) \qquad\qquad M_c = \mu {h_S}^{[k]}(\underline{t_1}).$$

Since N_1 is an occurrence c-net, ${}^\bullet t_1 \cap \underline{t_1} = \emptyset$ and, by Proposition 3.34, $conc({}^\bullet t_1 \cup \underline{t_1})$. Hence, by Corollary 3.40,

$$M_p \cap M_c = \emptyset \qquad \text{and} \qquad conc(M_p \cup M_c).$$

Moreover, by construction, $f_N \circ h^{[k]} = g_{|{N_1}^{[k]}}$, and therefore

$$\begin{array}{ll} \mu f_S(M_p) & \\ \quad = \mu f_S(\mu {h_S}^{[k]}({}^\bullet t_1)) & \\ \quad = \mu g_S({}^\bullet t_1) & \\ \quad = {}^\bullet t & \text{[by def. of morphism]} \end{array}$$

and in the same way $[\![\underline{t}]\!] \leq \mu f_S(M_c) \leq \underline{t}$. Thus, by definition of unfolding, there exists a transition $t' = \langle M_p, M_c, t\rangle$ in T'.

It is clear that, to obtain a well defined morphism that makes the diagram commute, we have to define

$$h_T{}^{[k+1]}(t_1) = t'$$

and, since $\mu g_S(t_1{}^\bullet) = t^\bullet$, for all $s_1 \in t_1{}^\bullet$ and $s \in t^\bullet$

$$h_S{}^{[k+1]}(s_1, \langle t', s\rangle) = g_S(s_1, s).$$

A routine checking allows to prove that, for each k, $h^{[k]}$ is a well-defined morphism and $f_N \circ h^{[k]} = g_{|N_1{}^{[k]}}$.

Uniqueness
The morphism h is clearly unique since at each step we were forced to define it as we did to ensure commutativity. Formally, let $h' : N_1 \to \mathscr{U}_a(N)$ be a morphism such that the diagram commutes, i.e., $f_N \circ h' = g$. Then, we show, that for all k

$$h'_{|N_1{}^{[k]}} = h_{|N_1{}^{[k]}}.$$

We proceed by induction on k:
($\mathbf{k = 0}$) The c-net $N_1{}^{[0]}$ consists only of the initial marking of N_1 and thus we have:

$$h'_T{}^{[0]} = \emptyset = h_T{}^{[0]},$$
$$h'_S{}^{[0]}(s_1, \langle\emptyset, s\rangle) = g_S(s_1, s) = h_S{}^{[0]}(s_1, \langle\emptyset, s\rangle), \quad \text{for all } s_1 \in S_1{}^{[0]} = m_1 \text{ and } s \in S.$$

($\mathbf{k \to k+1}$) For all $t_1 \in T^{[k+1]}$, with $depth(t_1) = k+1$ we distinguish two cases:

- If $g_T(t_1) = \perp$ then necessarily

  $$h'_T{}^{[k+1]}(t_1) = \perp \qquad \text{and} \qquad \mu h_S{}^{[k+1]}(t_1{}^\bullet) = \emptyset,$$

  thus $h'^{[k+1]}$ coincides with $h^{[k+1]}$, on t_1 and its post-set.

- If $g_T(t_1) = t$ then

  $$h'_T{}^{[k+1]}(t_1) = t' = \langle M_p, M_c, t\rangle \in T',$$

  with $M_p = {}^\bullet t' = \mu h'_S({}^\bullet t_1)$ and $M_c = \underline{t'} = \mu h'_S(\underline{t_1})$.[5] By inductive hypothesis, since $depth(s_1) \leq k$ for all $s_1 \in {}^\bullet t_1 \cup \underline{t_1}$, we have that $\mu h_S({}^\bullet t_1) = M_p$ and $\mu h_S(\underline{t_1}) = M_c$. Therefore, by definition of h, $h_T(t_1) = \langle M_p, M_c, t\rangle = h'_T(t_1)$.

  Moreover, for all $s_1 \in t_1{}^\bullet$ and for all $s \in t^\bullet$, again by reasoning on commutativity of the diagram, $h'_S(s_1, \langle t', s\rangle) = g_S(s_1, s) = h_S(s_1, \langle t', s\rangle)$.

□

[5] Notice that here equality holds since we are working with occurrence c-nets and thus contexts can only have multiplicity 1.

3.6 Occurrence contextual nets and asymmetric event structures

This section shows that the semantics of semi-weighted c-nets given in terms of occurrence c-nets can be abstracted to an event structure and to a domain semantics. First the existence of a coreflection between **AES** and **O-CN** is proved, substantiating the claim according to which AES's represent a suitable model for giving event based semantics to c-nets. Then the coreflection between **AES** and **Dom**, defined in Section 3.1, can be exploited to complete the chain of coreflections from **SW-CN** to **Dom**.

Given an occurrence c-net we can obtain a pre-AES by simply forgetting the places, but remembering the dependency relations that they induce between transitions, namely causality and asymmetric conflict. The corresponding (saturated) AES has the same causal relation $\leq_N$, while asymmetric conflict is given by the union of asymmetric conflict $\nearrow_N$ and of the induced binary conflict $\#_N$. Furthermore a morphism between occurrence c-nets naturally restricts to a morphism between the corresponding AES's.

Definition 3.48 (from occurrence c-nets to AES's)
*Let $\mathscr{E}_a$: **O-CN** $\to$ **AES** be the functor defined as:*

- *for each occurrence c-net N, if $\#_N$ denotes the induced binary conflict in N:*

$$\mathscr{E}_a(N) = \langle T, \leq_N, \nearrow_N \cup \#_N \rangle;$$

- *for each morphism $h : N_0 \to N_1$:*

$$\mathscr{E}_a(h : N_0 \to N_1) = h_T.$$

Notice that the induced conflict relation $\#^a$ in the AES $\mathscr{E}_a(N)$ (see Definition 3.2) coincides with the induced conflict relation in the net N (see Definition 3.32). Therefore in the following we will confuse the two relations and simply write $\#$ to denote both of them.

Proposition 3.49 (well-definedness)
$\mathscr{E}_a$ is a well-defined functor.

Proof. Given any occurrence c-net N, Definition 3.31 and the considerations on the saturation of pre-AES's following Definition 3.3, immediately imply that $\mathscr{E}_a(N)$ is an AES. Furthermore, if $h : N_0 \to N_1$ is a c-net morphism, then, by Corollary 3.43, $\mathscr{E}_a(h) = h_T$ is an AES-morphism. Finally $\mathscr{E}_a$ obviously preserves arrow composition and identities. □

An AES can be identified with a canonical occurrence c-net, via a free construction that mimics Winskel's: for each set of events related in a certain way by causality and asymmetric conflict we generate a unique place that induces such kind of relations on the events.

DEFINITION 3.50 (FROM AES'S TO OCCURRENCE C-NETS)
Let $G = \langle E, \leq, \nearrow \rangle$ be an AES. *Then $\mathscr{N}_a(G)$ is the net $N = \langle S, T, F, C, m \rangle$ defined as follows:*

- $m = \left\{ \langle \emptyset, A, B \rangle \mid \begin{array}{l} A, B \subseteq E,\ \forall a \in A.\ \forall b \in B.\ a \nearrow b, \\ \forall b, b' \in B.\ b \neq b' \Rightarrow b \# b' \end{array} \right\}$;
- $S = m \cup \left\{ \langle \{e\}, A, B \rangle \mid \begin{array}{l} A, B \subseteq E,\ e \in E,\ \forall x \in A \cup B.\ e < x, \\ \forall a \in A.\ \forall b \in B.\ a \nearrow b, \\ \forall b, b' \in B.\ b \neq b' \Rightarrow b \# b' \end{array} \right\}$;
- $T = E$;
- $F = \langle F_{pre}, F_{post} \rangle$, *with*

  $$F_{pre} = \{(e, s) \mid s = \langle x, A, B \rangle \in S,\ e \in B\},$$
  $$F_{post} = \{(e, s) \mid s = \langle \{e\}, A, B \rangle \in S\};$$

- $C = \{(e, s) \mid s = \langle x, A, B \rangle \in S,\ e \in A\}$.

The transitions of $\mathscr{N}_a(G)$ are simply the events of G, while places are triples of the form $\langle x, A, B \rangle$, with $x, A, B \subseteq E$, and $|x| \leq 1$. A place $\langle x, A, B \rangle$ is a precondition for all the events in B and a context for all the events in A. Moreover, if $x = \{e\}$, such a place is a postcondition for e, otherwise if $x = \emptyset$ the place belongs to the initial marking. Therefore each place gives rise to a conflict between each pair of (distinct) events in B and to an asymmetric conflict between each pair of events $a \in A$ and $b \in B$. Figure 3.6 presents some examples of basic AES's with the corresponding c-nets. The cases of an AES with two events related, respectively, by causality, asymmetric conflict and (immediate symmetric) conflict are considered. Pictorially, an asymmetric conflict $e_0 \nearrow e_1$ is represented by a dotted arrow from e_0 to e_1. Causality is represented, as usual, by plain arrows.

The next proposition relates the causality and asymmetric conflict relations of an AES with the corresponding relations of the c-net $\mathscr{N}_a(G)$. In particular it is useful in proving that $\mathscr{N}_a(G)$ is indeed an occurrence c-net.

LEMMA 3.51
Let $G = \langle E, \leq, \nearrow \rangle$ be an AES *and let $\mathscr{N}_a(G)$ be the net $N = \langle S, T, F, C, m \rangle$. Then for all $e, e' \in E$:*

1. $e <_N e'$ *iff* $e < e'$;
2. $e \rightsquigarrow_N e'$ *iff* $e \nearrow e'$;
3. $e \nearrow_N e'$ *iff* $e \nearrow e'$.

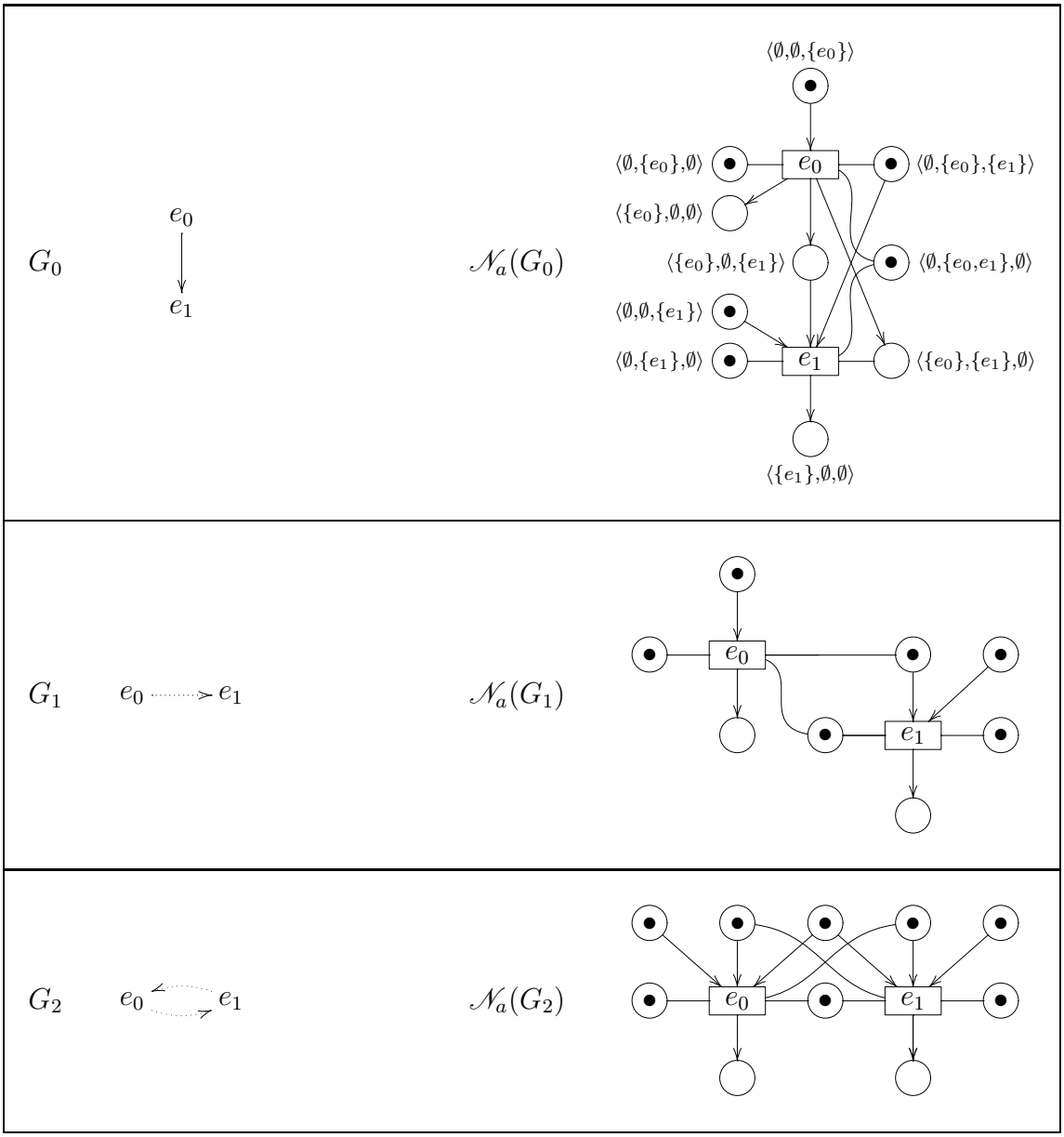


Figure 3.6: Three simple AES's and the corresponding occurrence c-nets produced by the functor $\mathscr{N}_a$.

PROOF.

1. Let $\prec_N$ denote the immediate causality relation in N. If $e \prec_N e'$ then there exists a place $\langle \{e\}, A, B\rangle \in S$ with $e' \in A \cup B$ and thus, by definition of $\mathscr{N}_a$, $e < e'$. Vice versa, if $e < e'$ then $\langle \{e\}, \emptyset, \{e'\}\rangle \in S$ and thus $e \prec_N e'$. Since $<_N$ is the transitive closure of $\prec_N$ and $<$ is a transitive relation we conclude the thesis.

2. If $e \rightsquigarrow_N e'$ then there exists a place $\langle x, A, B\rangle \in S$ with $e \in A \cup B$ and $e' \in B$ and thus either $e \nearrow e'$ or $e \# e'$. But since G is an AES, the binary conflict is included in the asymmetric conflict and thus, also in the second case, $e \nearrow e'$. Vice versa, if $e \nearrow e'$ then $\langle \emptyset, \{e\}, \{e'\}\rangle \in S$ and thus $e \rightsquigarrow_N e'$.

3. Immediate consequence of points (1) and (2).

□

As an immediate corollary we have:

COROLLARY 3.52
Let $G = \langle E, \leq, \nearrow\rangle$ be an AES. *Then $\mathscr{N}_a(G) = N = \langle S, T, F, C, m\rangle$ is an occurrence c-net.*

PROOF. By Lemma 3.51 the causality relation $\leq_N = \leq$ and the asymmetric conflict $\nearrow_N = \nearrow$ inherit the necessary properties from those of G. □

Let $G = \langle E, \leq, \nearrow\rangle$ be an AES. For $e \in E$, we define the set of *consequences* $\lceil \{e\} \rceil$, as follows (considering the singleton $\{e\}$ instead of e itself will simplify the notation later).

$$\lceil \{e\} \rceil = \{e' \in E \mid e < e'\}.$$

This function is extended also to the empty set, by $\lceil \emptyset \rceil = E$. We use the same notation for occurrence c-nets, referring to the underlying AES.

The next technical lemma gives a property of morphisms between occurrence c-nets which will be useful in the proof of the coreflection result.

LEMMA 3.53
Let N_0 and N_1 be occurrence c-nets and let $h : N_0 \to N_1$ be a morphism. For all $s_0 \in S_0$ and $s_1 \in S_1$, if $h_S(s_0, s_1)$ then

1. $h_T({}^\bullet s_0) = {}^\bullet s_1$;
2. $s_0{}^\bullet = h_T^{-1}(s_1{}^\bullet) \cap \lceil {}^\bullet s_0 \rceil$;
3. $\underline{s_0} = h_T^{-1}(\underline{s_1}) \cap \lceil {}^\bullet s_0 \rceil$.

Proof. Let $s_0 \in S_0$ and $s_1 \in S_1$ such that $h_S(s_0, s_1)$.

1. If ${}^\bullet s_0 = \emptyset$, i.e., $s_0 \in m_0$ then $s_1 \in m_1$ and thus ${}^\bullet s_1 = \emptyset = h_T({}^\bullet s_0)$. Otherwise, let ${}^\bullet s_0 = \{t_0\}$.[6] Therefore $h_T(t_0) = t_1$ is defined (see the remark after Definition 3.26) and $s_1 \in t_1{}^\bullet$. Thus ${}^\bullet s_1 = \{t_1\} = h_T({}^\bullet s_0)$.

2. Let $t_0 \in s_0{}^\bullet$, i.e., $s_0 \in {}^\bullet t_0$. Since $h_S(s_0, s_1)$, we have that $h_T(t_0) = t_1$ is defined and $s_1 \in {}^\bullet t_1$. Thus $t_0 \in h_T^{-1}(s_1{}^\bullet) \cap \lceil {}^\bullet s_0 \rceil$.

   For the converse inclusion, let $t_0 \in h_T^{-1}(s_1{}^\bullet) \cap \lceil {}^\bullet s_0 \rceil$. Then $s_1 \in {}^\bullet h_T(t_0)$ and thus there is $s_0' \in {}^\bullet t_0$ such that $h_S(s_0', s_1)$. Now, reasoning as in Lemma 3.42.(1), we conclude that s_0' and s_0 necessarily coincide, otherwise they would be in the post-condition of conflicting transitions and thus, since $t_0 \in \lceil {}^\bullet s_0 \rceil$, we would have $t_0 \# t_0$.

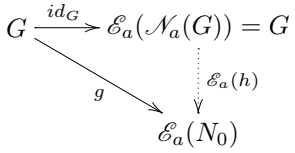
3. Analogous to (2). □

Recall that, by Lemma 3.51, for any aes $G = \langle E, \leq, \nearrow \rangle$ the causality and asymmetric conflict relations in $\mathscr{N}_a(G)$ coincide with $\leq$ and $\nearrow$. Hence $\mathscr{E}_a(\mathscr{N}_a(G)) = \langle E, \leq, \nearrow' \rangle$, where $\nearrow' = \nearrow \cup \# = \nearrow$, where the last equality is justified by the fact that in an aes $\# \subseteq \nearrow$. Hence $\mathscr{E}_a \circ \mathscr{N}_a$ is the identity on objects.

We next prove that $\mathscr{N}_a$ extends to a functor from **AES** to **O-CN**, which is left adjoint to $\mathscr{E}_a$ (with unit the identity id_G). More precisely they establish a coreflection from **AES** to **O-CN**.

Theorem 3.54 (coreflection between **O-CN** and **AES**)
$\mathscr{N}_a \dashv \mathscr{E}_a$

Proof. Let $G = \langle E, \leq, \nearrow \rangle$ be an aes and let $\mathscr{N}_a(G) = \langle S, T, F, C, m \rangle$ be as in Definition 3.50. We have to show that for any occurrence c-net N_0 and for any morphism $g : G \to \mathscr{E}_a(N_0)$ there exists a unique morphism $h : \mathscr{N}_a(G) \to N_0$, such that the following diagram commutes:



The behaviour of h on transitions is determined immediately by g:

$$h_T = g.$$

Therefore we only have to show that a multirelation $h_S : S \to S_0$ such that $\langle h_T, h_S \rangle$ is a morphism exists and is uniquely determined by h_T.

Existence
Let us define h_S in such a way it satisfies the conditions of Lemma 3.53, specialized to the net $\mathscr{N}_a(G)$, that is, for all $s = \langle x, A, B \rangle \in S$ and $s_0 \in S_0$:

[6] There is a unique transition generating s_0, since N_0 is an occurrence c-net.

$$h_S(s,s_0) \quad \text{iff} \quad ((x=\emptyset \ \wedge \ s_0 \in m_0) \ \vee \ (x=\{t\} \ \wedge \ s_0 \in h_T(t)^\bullet)) \\ \wedge \ B = h_T^{-1}(s_0{}^\bullet) \cap \lceil x \rceil \\ \wedge \ A = h_T^{-1}(\underline{s_0}) \cap \lceil x \rceil$$

To prove that the pair $h = \langle h_T, h_S \rangle$ is indeed a morphism, let us verify the conditions on the preservation of the initial marking and of the pre-set, post-set and context of transitions.

First observe that $\mu h_S(m) = m_0$. In fact, if $s = \langle x, A, B \rangle \in m$ and $h_S(s, s_0)$ then $x = \emptyset$ and thus, by definition of h_S, $s_0 \in m_0$. Vice versa, let $s_0 \in m_0$ and let

$$A = h_T^{-1}(\underline{s_0}) \qquad \text{and} \qquad B = h_T^{-1}(s_0{}^\bullet)$$

Since $t_0 \# t_0'$ for all $t_0, t_0' \in s_0{}^\bullet$ and $t_0 \nearrow t_0'$ for all $t_0 \in \underline{s_0}$, $t_0' \in s_0{}^\bullet$, by definition of AES-morphism, $t \# t'$ for all $t, t' \in B$ and $t \nearrow t'$ for all $t \in A$ and $t' \in \overline{B}$. Hence there is a place $s = \langle \emptyset, A, B \rangle \in m$ and $h_S(s, s_0)$.

Now, let $t \in T$ be any transition, such that $h_T(t)$ is defined. Then

- $\mu h_S({}^\bullet t) = {}^\bullet h_T(t)$.
  In fact, let $s = \langle x, A, B \rangle \in {}^\bullet t$, that is $t \in B$, and let $h_S(s, s_0)$. Then, by definition of h_S, $h_T(t) \in s_0{}^\bullet$, or equivalently $s_0 \in {}^\bullet h_T(t)$. For the converse inclusion, let $s_0 \in {}^\bullet h_T(t)$ and let $x = h_T^{-1}({}^\bullet s_0) \cap \lfloor t \rfloor$. Since N_0 is an occurrence c-net $\mid {}^\bullet s_0 \mid \leq 1$ and thus $\mid x \mid \leq 1$ (more precisely $x = \emptyset$ if $s_0 \in m_0$, otherwise, x contains the unique $t' \leq t$, such that $h_T(t') = t_0$, with ${}^\bullet s_0 = \{t_0\}$). Consider

  $$A = h_T^{-1}(\underline{s_0}) \cap \lceil x \rceil \quad \text{and} \quad B = h_T^{-1}(s_0{}^\bullet) \cap \lceil x \rceil.$$

  Since $t_0 \# t_0'$ for all $t_0, t_0' \in s_0{}^\bullet$ and $t_0 \nearrow t_0'$ for all $t_0 \in \underline{s_0}$, $t_0' \in s_0{}^\bullet$, as in the previous case, we have that $s = \langle x, A, B \rangle \in S$ is a place such that $h_S(s, s_0)$. Clearly $t \in \lceil x \rceil$, thus $t \in B$ and therefore $s \in {}^\bullet t$ and $s_0 \in \mu h_S({}^\bullet t)$.

- $\mu h_S(\underline{t}) = \underline{h_T(t)}$.
  Analogous to the previous case.

- $\mu h_S(t^\bullet) = h_T(t)^\bullet$.
  If $s = \langle x, A, B \rangle \in t^\bullet$, that is $x = \{t\}$, and $h_S(s, s_0)$, then, by definition of h_S, we have $s_0 \in h_T(t)^\bullet$. For the converse, let $s_0 \in h_T(t)^\bullet$. As above, consider

  $$A = h_T^{-1}(\underline{s_0}) \cap \lceil \{t\} \rceil \quad \text{and} \quad B = h_T^{-1}(s_0{}^\bullet) \cap \lceil \{t\} \rceil.$$

  Then $s = \langle \{t\}, A, B \rangle \in t^\bullet$ and, by definition of h_S, we have $h_S(s, s_0)$.

Finally, if $h_T(t)$ is not defined, then the definition of h_S implies that $\mu h_S({}^\bullet t) = \mu h_S(\underline{t}) = \mu h_S(t^\bullet) = \emptyset$. This concludes the proof that h is a morphism.

Uniqueness

The multirelation h_S such that $\langle h_T, h_S \rangle$ is a c-net morphism is unique essentially because it must satisfy the conditions of Lemma 3.53. More precisely, if $h_S' : S \to S_0$ is another multirelation, such that $\langle h_T, h_S' \rangle$ is a morphism and $h_S'(s, s_0)$ then necessarily by Lemma 3.53, $h_S(s, s_0)$. Conversely, let $h_S(s, s_0)$, with $s = \langle x, A, B \rangle$. Then, if $x = \emptyset$, by properties of net morphisms, $s_0 \in m_0$. Therefore there must be $s' \in m$ such that $h_S'(s', s_0)$. But, by Lemma 3.53 and definition of h_S, $\underline{s'} = h_T^{-1}(\underline{s_0}) = A$ and similarly $s'^\bullet = h_T^{-1}(s_0{}^\bullet) = B$. Therefore $s' = \langle \emptyset, A, B \rangle = s$ and thus $h_S'(s, s_0)$. An analogous reasoning allow us to conclude in the case $x = \{t\}$. □

We know by the previous theorem that $\mathscr{N}_a$ extends to a functor from **AES** to **O-CN**. The behaviour of $\mathscr{N}_a$ on morphisms is suggested by the proof of the theorem. Let $h : G_0 \to G_1$, be an AES-morphism and let $\mathscr{N}_a(G_i) = \langle S_i, T_i, F_i, C_i, m_i \rangle$ for $i \in \{0, 1\}$. Then $\mathscr{N}_a(h) = \langle h, h_S \rangle$, with h_S defined as follows:

- for all places $\langle \emptyset, A_1, B_1 \rangle$

$$h_S(\langle \emptyset, h^{-1}(A_1), h^{-1}(B_1) \rangle, \langle \emptyset, A_1, B_1 \rangle)$$

- for all $e_0 \in T_0$ such that $h_T(e_0) = e_1$ and for all places $\langle \{e_1\}, A_1, B_1 \rangle$

$$h_S(\langle \{e_0\}, h^{-1}(A_1) \cap \lceil e_0 \rceil, h^{-1}(B_1) \cap \lceil e_0 \rceil \rangle, \langle \{e_1\}, A_1, B_1 \rangle)$$

Finally, notice that the equivalence between **PES** and **Dom** (see Section 2.4) can be used to "translate" the domain semantics of semi-weighted c-nets into a prime event structure semantics. This completes the following chain of coreflections between **SW-CN** and **PES**

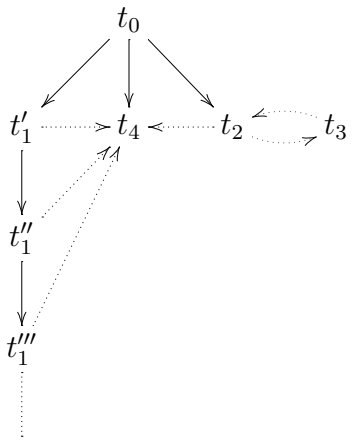
$$\mathbf{SW\text{-}CN} \underset{\mathscr{U}_a}{\overset{\mathscr{I}_O}{\underset{\longrightarrow}{\overset{\longleftarrow}{\bot}}}} \mathbf{O\text{-}CN} \underset{\mathscr{E}_a}{\overset{\mathscr{N}_a}{\underset{\longrightarrow}{\overset{\longleftarrow}{\bot}}}} \mathbf{AES} \underset{\mathscr{L}_a}{\overset{\mathscr{P}_a}{\underset{\longrightarrow}{\overset{\longleftarrow}{\bot}}}} \mathbf{Dom} \underset{\mathscr{P}}{\overset{\mathscr{L}}{\underset{\longrightarrow}{\overset{\longleftarrow}{\sim}}}} \mathbf{PES}$$

Figure 3.7 shows (part of) the AES, the domain and the PES associated to the c-net of Figure 3.5. Although (for the sake of readability) not explicitly drawn, in the PES all the "copies" of t_4, namely the events t_4^x are in conflict.

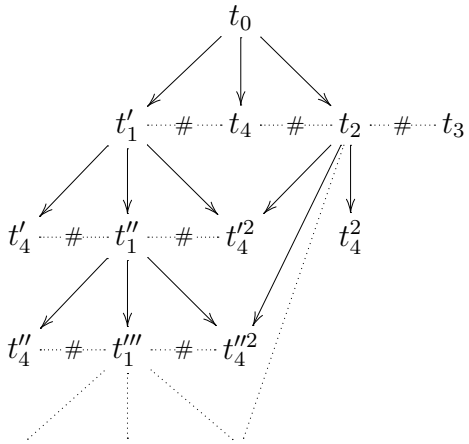
We remark that the PES semantics is obtained from the AES semantics by introducing an event for each possible different history of events in the AES. For instance, the PES semantics of the net N_0 in Figure 3.8 is given by P, where e_1' represents the firing of the transition t_1 by itself, with an empty history, and e_1'' the firing of the transition t_1 after t_0. Obviously the AES semantics is finer than the PES semantics, or in other words the translation from **AES** to **PES** causes a loss of information. For example, the nets N_3 and N_3' in Figure 3.8 have the same PES semantics, but different AES semantics.

3.7 Processes of c-nets and their relation with the unfolding

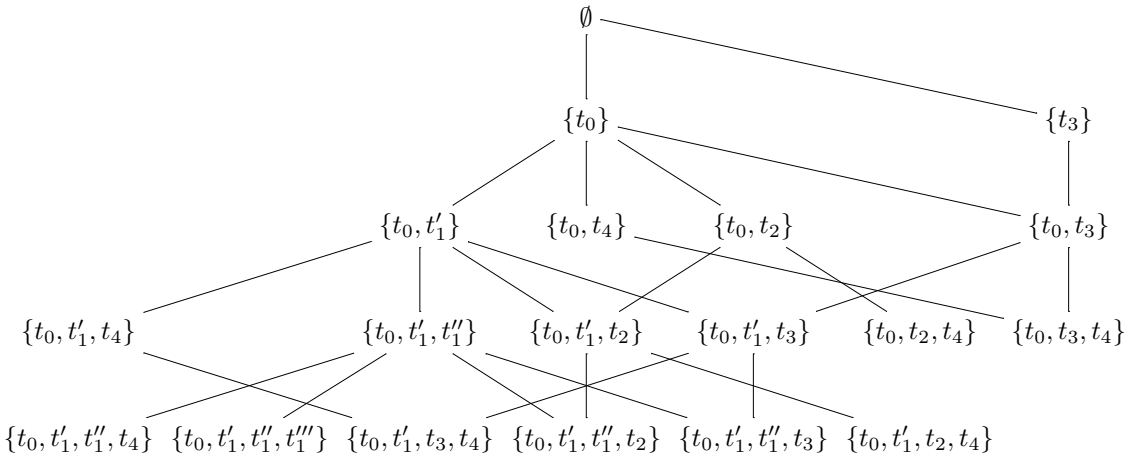
The notion of occurrence c-net introduced in Section 3.4 naturally suggests a notion of nondeterministic process for c-nets, which can be defined as an occurrence c-net with a morphism (mapping places into places and total on transitions) to the original net. Deterministic c-net processes can then be defined as particular nondeterministic processes such that the underlying occurrence c-net satisfies a further conflict-freeness requirement. Interestingly, the resulting notion



(a) $\mathscr{E}_a(\mathscr{U}_a(N))$



(c) $\mathscr{P}(\mathscr{L}_a(\mathscr{E}_a(\mathscr{U}_a(N))))$



(b) $\mathscr{L}_a(\mathscr{E}_a(\mathscr{U}_a(N)))$

Figure 3.7: The (a) AES (b) domain and (c) PES for the c-net N of Figure 3.5

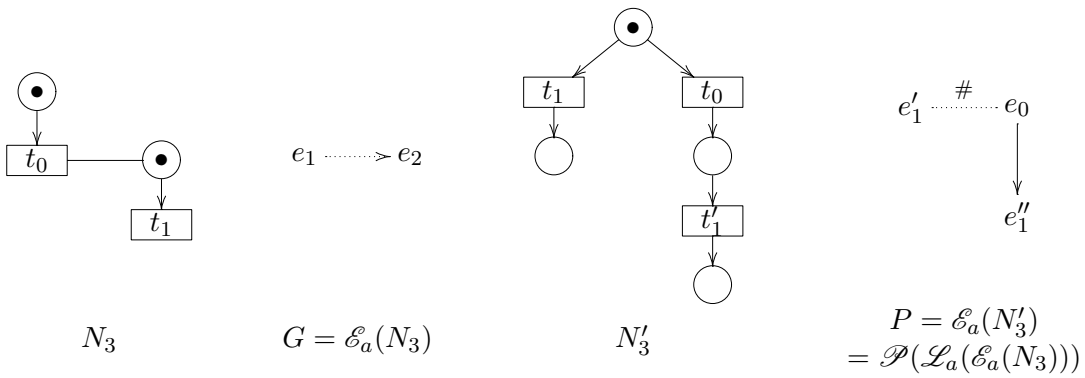


Figure 3.8: AES semantics is finer than PES semantics.

of deterministic process turns out to coincide with those proposed by other authors [Vog97b, Ris94, GM98, Win98, Bus98] (see Section 2.3). It is worth recalling that the stress on the necessity of using an additional relation of "weak-causality" to be able to fully express the causal structure of net computations in the presence of read or inhibitor arcs can be found already in [JK93, JK95]. However, we already observed that a different notion of enabling allowing for the simultaneous firing of weakly dependent transitions is used in [JK95], making difficult a complete direct comparison. For the same reason, although "syntactically" the processes of [Vog97b] coincide with ours, they are intended to represent the same firing sequences, but different step sequences.

The papers [GM98, Win98] extend the theory of concatenable processes of ordinary nets [DMM89] to c-nets, by showing that the concatenable processes of a c-net N form the arrows of a symmetric monoidal category $\mathbf{CP}[N]$, where objects are the elements of the free commutative monoid over the set of places (multisets of places). In particular, in [GM98] a purely algebraic characterization of such category is given.

Since the category $\mathbf{CP}[N]$ of concatenable processes of a net N provides a computational model for N, describing its operational behaviour, we are naturally lead to compare such semantics with the one based on the unfolding. This section, relying on the notion of concatenable c-net process and exploiting the chain of coreflections leading from **SW-CN** to **Dom**, establishes a close relationship between process and unfolding semantics for c-nets. More precisely, we generalize to c-nets (in the semi-weighted case) a result proved in [MMS96] for ordinary nets, stating that the domain $\mathscr{L}_a(\mathscr{E}_a(\mathscr{U}_a(N)))$ associated to a semi-weighted net N coincides with the completion of the preorder obtained as the comma category of $\mathbf{CP}[N]$ under the initial marking. Roughly speaking, the result says that the domain obtained via the unfolding of a c-net can be equivalently described as the collection of the deterministic processes of the net, ordered by prefix.

3.7.1 Contextual nets processes

A process of a c-net N can be naturally defined as an occurrence c-net N_φ, together with a morphism φ to the original net. In fact, since morphisms preserve the token game, φ maps computations of N_φ into computations of N in such a way that the process can be seen as a representative of a set of possible computations of N. The occurrence c-net N_φ makes explicit the causal structure of such computations since each transition is fired at most once and each place is filled at most with one token during each computation. In this way (as it happens in the unfolding) transitions and places of N_φ can be thought of, respectively, as instances of firing of transitions and tokens in places of the original net. Actually, to allow for such an interpretation, some further restrictions have to be imposed on the morphism φ, namely it must map places into places (rather than into multisets of places) and it must be total on transitions.

Besides "marked processes", representing computations of the net starting from the initial marking, we will introduce also "unmarked processes", representing computations starting from a generic marking. This is needed to be able to define a meaningful notion of concatenation between processes.

DEFINITION 3.55 ((NONDETERMINISTIC) PROCESS)
A marked process *of a c-net $N = \langle S, T, F, C, m\rangle$ is a mapping $\varphi : N_\varphi \to N$, where N_φ is an occurrence c-net and φ is a strong c-net morphism, namely a c-net morphism such that φ_T is total and φ_S maps places into places. The process is called* discrete *if N_φ has no transitions.*

An unmarked process *of N is defined in the same way, where the mapping φ is an "unmarked morphism", namely φ is not required to preserve the initial marking (it satisfies all conditions of Definition 3.26, but (1)).*

Equivalently, if we denote by $\mathbf{CN}^*$ the subcategory of $\mathbf{CN}$ where the arrows are *strong c-net* morphisms, the processes of N can be seen as objects of the comma category $\langle \mathbf{O\text{-}CN} \downarrow N\rangle$ in $\mathbf{CN}^*$. This gives also the (obvious) notion of isomorphism between processes, which is an isomorphism between the underlying occurrence nets "consistent" with the mappings to the original net. Analogous definitions can be given also for the unmarked processes of a net N.[7]

A deterministic process represents a set of computations which differ only for the order in which independent transitions are fired. In our setting a deterministic process is thus defined as a process such that, in the underlying occurrence net, the transitive closure of asymmetric conflict is a finitary partial order, in such a way

[7]It is worth remarking that if we want each truly concurrent computation of the net N to be represented by at most one configuration of the nondeterministic process, an additional constraint must be imposed on φ, requiring that ${}^\bullet t_1 = {}^\bullet t_2$, $\underline{t_1} = \underline{t_2}$ and $\varphi(t_1) = \varphi(t_2)$ implies $t_1 = t_2$, as in [VSY98]. However, the two notions of process collapse when we restrict to deterministic processes which are the focus of this section.

that all transitions can be fired in a single computation of the net. Deterministic occurrence c-nets will be always denoted by O, possibly with subscripts.

Definition 3.56 (deterministic occurrence c-net)
An occurrence c-net O is called deterministic *if the asymmetric conflict $\nearrow_O$ is acyclic and well-founded.*

Equivalently, one could have asked the transitive closure of the asymmetric conflict relation $(\nearrow_O)^*$ to be a partial order, such that for each transition t in O, the set $\{t' \mid t'(\nearrow_O)^* t\}$ is finite. Alternatively, it can be easily seen that a finite occurrence c-net is deterministic if and only if the corresponding AES is conflict free.

We denote by $\min(O)$ and $\max(O)$ the sets of minimal and maximal places of O with respect to the partial order $\leq_O$.

Definition 3.57 (deterministic process)
A (marked or unmarked) process φ is called deterministic *if the occurrence c-net O_φ is deterministic. The process is* finite *if the set of transitions in O_φ is finite. In this case, we denote by* $\min(\varphi)$ *and* $\max(\varphi)$ *the sets* $\min(O_\varphi)$ *and* $\max(O_\varphi)$*, respectively. Moreover we denote with ${}^\bullet\varphi$ and $\varphi^\bullet$ the multisets $\mu\varphi_S(\min(\varphi))$ and $\mu\varphi_S(\max(\varphi))$, called respectively the* source *and the* target *of φ.*

Clearly, in the case of a marked process φ of a c-net N, the marking ${}^\bullet\varphi$ coincides with the initial marking of N.

3.7.2 Concatenable processes

As in [GM98, Win98] a notion of concatenable process for contextual nets, endowed with an operation of sequential (and parallel) composition, can be easily defined, generalizing the concatenable processes of [DMM89]. Obviously a meaningful operation of sequential composition can be defined only on the unmarked processes of a c-net. In order to properly define the operation of concatenation of processes, we need to impose a suitable ordering over the places in $\min(\varphi)$ and $\max(\varphi)$ for each process φ. Such ordering allows one to distinguish among "interface" places of O_φ which are mapped to the same place of the original net, a capability which is essential to make sequential composition consistent with the causal dependencies.

Definition 3.58
*Let A and B be sets and let $f : A \to B$ be a function. An f-*indexed ordering *is a family $\alpha = \{\alpha_b \mid b \in B\}$ of bijections $\alpha_b : f^{-1}(b) \to [|f^{-1}(b)|]$, where $[i]$ denotes the subset $\{1, \ldots, i\}$ of $\mathbb{N}$, and $f^{-1}(b) = \{a \in A \mid f(a) = b\}$.*

The f-indexed ordering α will be often identified with the function from A to $\mathbb{N}$ it naturally induces (formally defined as $\bigcup_{b \in B} \alpha_b$).

DEFINITION 3.59 (CONCATENABLE PROCESS)
A concatenable process *of a c-net N is a triple $\gamma = \langle \mu, \varphi, \nu \rangle$, where*

- *φ is a finite deterministic unmarked process of N;*
- *μ is φ-indexed ordering of* $\min(\varphi)$*;*
- *ν is φ-indexed ordering of* $\max(\varphi)$*.*

Two concatenable processes $\gamma_1 = \langle \mu_1, \varphi_1, \nu_1 \rangle$ and $\gamma_2 = \langle \mu_2, \varphi_2, \nu_2 \rangle$ of a c-net N are *isomorphic* if there exists an isomorphism of processes $f : \varphi_1 \to \varphi_2$, consistent with the decorations, i.e., such that $\mu_2(f_S(s_1)) = \mu_1(s_1)$ for each $s_1 \in \min(\varphi_1)$ and $\nu_2(f_S(s_1)) = \nu_1(s_1)$ for each $s_1 \in \max(\varphi_1)$. An isomorphism class of processes is called *(abstract) concatenable process* and denoted by $[\gamma]$, where γ is a member of that class. In the following we will often omit the word "abstract" and write γ to denote the corresponding equivalence class.

The operation of sequential composition on concatenable processes is defined in the natural way. Given two concatenable processes $\langle \mu_1, \varphi_1, \nu_1 \rangle$ and $\langle \mu_2, \varphi_2, \nu_2 \rangle$, such that ${\varphi_1}^\bullet = {}^\bullet\varphi_2$ their concatenation is defined as the process obtained by gluing the maximal places of φ_1 and the minimal places of φ_2 according to the ordering of such places.

DEFINITION 3.60 (SEQUENTIAL COMPOSITION)
Let $\gamma_1 = \langle \mu_1, \varphi_1, \nu_1 \rangle$ and $\gamma_2 = \langle \mu_2, \varphi_2, \nu_2 \rangle$ be two concatenable processes of a c-net N such that ${\varphi_1}^\bullet = {}^\bullet\varphi_2$. Suppose $T_1 \cap T_2 = \emptyset$ and $S_1 \cap S_2 = \max(\varphi_1) = \min(\varphi_2)$, with $\varphi_1(s) = \varphi_2(s)$ and $\nu_1(s) = \mu_2(s)$ for each $s \in S_1 \cap S_2$. In words γ_1 and γ_2 overlap only on $\max(\varphi_1) = \min(\varphi_2)$*, and on such places the labelling on the original net and the ordering coincide. Then their concatenation $\gamma_1; \gamma_2$ is the concatenable process $\gamma = \langle \mu_1, \varphi, \nu_2 \rangle$, where the process φ is the (componentwise) union of φ_1 and φ_2*

It is easy to see that concatenation induces a well-defined operation of sequential composition between abstract processes. In particular, if $[\gamma_1]$ and $[\gamma_2]$ are abstract concatenable processes such that ${\gamma_1}^\bullet = {}^\bullet\gamma_2$ then we can always find $\gamma_2' \in [\gamma_2]$ such that $\gamma_1; \gamma_2'$ is defined. Moreover the result of the composition at abstract level, namely $[\gamma_1; \gamma_2']$, does not depend on the particular choice of the representatives.

DEFINITION 3.61 (CATEGORY OF CONCATENABLE PROCESSES)
Let N be a c-net. The category of (abstract) concatenable processes *of N, denoted by $\mathbf{CP}[N]$, is defined as follows. Objects are multisets of places of N, namely elements of μS. Each (abstract) concatenable process $[\langle \mu, \varphi, \nu \rangle]$ of N is an arrow from ${}^\bullet\varphi$ to ${\varphi}^\bullet$.*

One could also define a tensor operation $\otimes$, modelling parallel composition of processes, making the category $\mathbf{CP}[N]$ a symmetric monoidal category (the symmetries being the discrete processes). Since such operation is not relevant for our present aim, we refer the interested reader to [GM98, Win98].

3.7.3 Relating processes and unfolding

Let $N = \langle S, T, F, C, m\rangle$ be a c-net and consider the comma category $\langle m \downarrow \mathbf{CP}[N]\rangle$. The objects of such category are concatenable processes of N starting from the initial marking. An arrow exists from a process γ_1 to γ_2 if the second one can be obtained by concatenating the first one with a third process γ. This can be interpreted as a kind of prefix ordering.

Lemma 3.62
Let $N = \langle S, T, F, C, m\rangle$ be a c-net. Then the comma category $\langle m \downarrow \mathbf{CP}[N]\rangle$ is a preorder.

Proof. Let $\gamma_i : m \to M_i$ ($i \in \{1, 2\}$) be two objects in $\langle m \downarrow \mathbf{CP}[N]\rangle$, and suppose there are two arrows $\gamma', \gamma'' : \gamma_1 \to \gamma_2$. By definition of comma category $\gamma_1; \gamma' = \gamma_1; \gamma'' = \gamma_2$, which, by definition of sequential composition, easily implies $\gamma' = \gamma''$. □

In the sequel the preorder relation over $\langle m \downarrow \mathbf{CP}[N]\rangle$ (induced by sequential composition) will be denoted by $\lesssim_N$ or simply by $\lesssim$, when the net N is clear from the context. Therefore $\gamma_1 \lesssim \gamma_2$ if there exists γ such that $\gamma_1; \gamma = \gamma_2$.

We provide an alternative characterization of the preorder relation $\lesssim_N$ which will be useful in the sequel. It essentially formalizes the intuition given above, according to which the preorder on $\langle m \downarrow \mathbf{CP}[N]\rangle$ is a generalization of the prefix relation. First, we need to introduce the notion of left-injection for processes.

Definition 3.63 (left injection)
Let $\gamma_i : m \to M_i$ ($i \in \{1, 2\}$) be two objects in $\langle m \downarrow \mathbf{CP}[N]\rangle$, with $\gamma_i = \langle \mu_i, \varphi_i, \nu_i\rangle$. A left injection *$\iota : \gamma_1 \to \gamma_2$ is a morphism of marked processes $\iota : \varphi_1 \to \varphi_2$, such that*

1. *ι is consistent with the indexing of minimal places, namely $\mu_1(s) = \mu_2(\iota(s))$ for all $s \in \min(\varphi_1)$;*
2. *ι is "rigid" on transitions, namely for t'_2 in O_{φ_2} and t_1 in O_{φ_1}, if $t'_2 \nearrow \iota(t_1)$ then $t'_2 = \iota(t'_1)$ for some t'_1 in O_{φ_1}.*

The name "injection" is justified by the fact that a morphism ι between marked deterministic processes (being a morphism between the underlying deterministic occurrence c-nets) is injective on places and transitions, as it can be shown easily by using the properties of (occurrence) c-nets morphisms proved in Section 3.4. The word "left" is instead related to the requirement of consistency with the decoration of the minimal items. Finally, the rigidity of the morphism ensures that γ_2 does not extend γ_1 with transitions inhibited in γ_1.

Lemma 3.64
Let $\gamma_i : m \to M_i$ ($i \in \{1, 2\}$) be two objects in $\langle m \downarrow \mathbf{CP}[N]\rangle$, with $\gamma_i = \langle \mu_i, \varphi_i, \nu_i\rangle$. Then

$$\gamma_1 \lesssim \gamma_2 \quad \textit{iff} \quad \textit{there exists a left injection } \iota : \gamma_1 \to \gamma_2.$$

PROOF. ($\Rightarrow$) Let $\gamma_1 \lesssim \gamma_2$, namely $\gamma_2 = \gamma_1;\gamma$ for some process $\gamma = \langle \mu, \varphi, \nu \rangle$. Without loss of generality, we can assume that φ_2 is obtained as the componentwise union of φ_1 and φ and this immediately gives a morphism of marked processes (the inclusion) $\iota : \varphi_1 \to \varphi_2$, consistent with the indexing of minimal places. To conclude it remains only to show that ι is rigid. Suppose that $t_2' \nearrow \iota(t_1)$ for some transitions t_1 in O_{φ_1} and t_2' in O_{φ_2}, and thus, by Definition 3.30, either $t_2' \rightsquigarrow \iota(t_1)$ or $t_2' < \iota(t_1)$. To conclude that ι is rigid we must show that in both cases t_2' is in O_{φ_1}.

- If $t_2' \rightsquigarrow \iota(t_1)$, since the process φ_2 is deterministic, t_2' and $\iota(t_1)$ cannot be in conflict and thus it must be $\underline{t_2'} \cap {}^\bullet\iota(t_1) \neq \emptyset$. Since t_2' uses as context a place which is not maximal in O_{φ_1}, necessarily $\underline{t_2'}$ is in O_{φ_1}, otherwise it could not be added by concatenating φ to φ_1.
- If $t_2' < \iota(t_1)$ then we can find a transition t_3' in O_{φ_2} such that $t_2' < t_3'$ and $t_3'^\bullet \cap ({}^\bullet\iota(t_1) \cup \underline{\iota(t_1)})$. As above, t_3' must be in O_{φ_1} since it uses as postcondition a place in O_{φ_1}. An inductive reasoning based on this argument shows that also t_2' is in O_{φ_1}.

($\Leftarrow$) Let $\iota : \gamma_1 \to \gamma_2$ be a left injection. We can suppose without loss of generality that O_{φ_1} is a subnet of O_{φ_2}, in such a way that ι is the inclusion and $\mu_1 = \mu_2$. Let O_φ be the net $(O_{\varphi_2} - O_{\varphi_1}) \cup \max(O_{\varphi_1})$, where difference and union are defined componentwise. More precisely $O_\varphi = \langle S, T, F, C \rangle$, with:

- $S = (S_2 - S_1) \cup \max(\varphi_1)$
- $T = T_2 - T_1$
- the relations F and C are the restrictions of F_2 and C_2 to T.

It is easy to see that O_φ is a well-defined occurrence c-net and $\min(O_\varphi) = \max(O_{\varphi_1})$. In particular, the fact that F is well-defined namely that if $t \in T$ then ${}^\bullet t, t^\bullet \subseteq S$ immediately derives from the fact that the inclusion ι is a morphism of deterministic occurrence c-nets. Instead the well-definedness of C is related to the fact that the injection is rigid. In fact, let $s \in \underline{t}$ for $t \in T$ and suppose that $s \notin S$. Therefore $s \in {}^\bullet t_1$, for some $t_1 \in T_1$ and thus $t \nearrow t_1$, which, by rigidity, implies $t \in T_1$, contradicting $t \in T$.

Therefore if we denote by γ the concatenable process $\langle \nu_1, \varphi, \nu_2 \rangle$, then $\gamma_1;\gamma = \gamma_2$, and thus $\gamma_1 \lesssim \gamma_2$. □

We can show now that the ideal completion of the preorder $\langle m \downarrow \mathbf{CP}[N] \rangle$ is isomorphic to the domain obtained from the unfolding of the net N, namely $\mathscr{L}_a(\mathscr{E}_a(\mathscr{U}_a(N)))$. Besides exploiting the characterization of the preorder relation on $\langle m \downarrow \mathbf{CP}[N] \rangle$ given above, the result strongly relies on the description of the unfolding construction as chain of adjunctions.

First, it is worth recalling some definitions and results on the ideal completion of (pre)orders.

DEFINITION 3.65 (IDEAL)
Let P be a preorder. An ideal *of P is a subset $S \subseteq P$, directed and downward closed (namely $S = \bigcup\{\downarrow x \mid x \in S\}$). The set of ideals of P, ordered by subset inclusion is denoted by $\mathsf{Idl}(P)$.*

Given a preorder P, the partial order $\mathsf{Idl}(P)$ is an algebraic CPO, with compact elements $\mathsf{K}(\mathsf{Idl}(P)) = \{\downarrow p \mid p \in P\}$. Moreover $\mathsf{Idl}(P) \simeq \mathsf{Idl}(P/_\equiv)$, where $P/_\equiv$ is the partial order induced by the preorder P. Finally, recall that if D is an algebraic CPO, then $\mathsf{Idl}(\mathsf{K}(D)) \simeq D$.

LEMMA 3.66
Let P_1 and P_2 be preorders and let $f : P_1 \to P_2$ be a surjective function such that $p_1 \sqsubseteq p_1'$ iff $f(p_1) \sqsubseteq f(p_1')$. Then the function $f^* : \mathsf{Idl}(P_1) \to \mathsf{Idl}(P_2)$, defined by $f^*(I) = \{f(x) \mid x \in I\}$, for $I \in \mathsf{Idl}(P_1)$, is an isomorphism of partial orders.

PROOF. The function f^* is surjective since for every ideal $I_2 \in \mathsf{Idl}(P_2)$ it can be easily proved that $f^{-1}(I_2)$ is an ideal and $f^*(f^{-1}(I_2)) = I_2$ by surjectivity of f. Moreover, notice that if $I_1, I_1' \in \mathsf{Idl}(P_1)$ are two ideals then $I_1 \subseteq I_1'$ if and only if $f^*(I_1) \subseteq f^*(I_1')$. The right implication is obvious. For the left one, assume $f^*(I_1) \subseteq f^*(I_1')$. Then observe that if $x \in I_1$ then $f(x) \in f^*(I_1) \subseteq f^*(I_1')$. Hence there exists $x' \in I_1'$ such that $f(x') = f(x)$. Thus by hypothesis on f we have $x \sqsubseteq x'$ and therefore, by definition of ideal, $x \in I_1'$.

Then we can conclude that f^* is also injective, thus it is a bijection, and clearly f^* as well as its inverse are monotone functions. □

Notice that in particular, if P is a preorder, D is an algebraic CPO and $f : P \to \mathsf{K}(D)$ is a surjection such that $p \sqsubseteq p'$ iff $f(p) \sqsubseteq f(p')$, then $\mathsf{Idl}(P) \simeq \mathsf{Idl}(\mathsf{K}(D)) \simeq D$.

We can now prove the main result of this section, which establishes a tight relationship between the unfolding and the process semantics of semi-weighted c-nets. We show that the ideal completion of the preorder $\langle m \downarrow \mathbf{CP}[N]\rangle$ and the domain associated to the net N through the unfolding construction are isomorphic. To understand which is the meaning of taking the ideal completion of the preorder $\langle m \downarrow \mathbf{CP}[N]\rangle$, first notice that the elements of the partial order induced by the preorder $\langle m \downarrow \mathbf{CP}[N]\rangle$ are classes of concatenable processes with respect to an equivalence $\equiv_l$ defined by $\gamma_1 \equiv_l \gamma_2$ if there exist a discrete concatenable process γ such that $\gamma_1;\gamma = \gamma_2$. In other words, $\gamma_1 \equiv_l \gamma_2$ can be read as "γ_1 and γ_2 left isomorphic", where "left" means that the isomorphism is required to be consistent only with respect to the ordering of the minimal places. Since the net N is semi-weighted, the equivalence $\equiv_l$ turns out to coincide with the isomorphism of marked processes. In fact, being the initial marking of N a set, only one possible ordering function exists for the minimal places of a marked process. Finally, since processes are finite, taking the ideal completion of the partial order induced by the preorder $\langle m \downarrow \mathbf{CP}[N]\rangle$ (which produces the same result as taking directly the ideal completion of $\langle m \downarrow \mathbf{CP}[N]\rangle$) is necessary to move from finite computations to arbitrary ones.

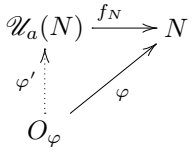
THEOREM 3.67 (UNFOLDING VS. CONCATENABLE PROCESSES)
Let N be a semi-weighted c-net. Then the ideal completion of $\langle m \downarrow \mathbf{CP}[N]\rangle$ is isomorphic to the domain $\mathscr{L}_a(\mathscr{E}_a(\mathscr{U}_a(N)))$.

PROOF. Let $N = \langle S, T, F, C, m\rangle$ be a c-net. It is worth recalling that the compact elements of the domain $\mathscr{L}_a(\mathscr{E}_a(\mathscr{U}_a(N)))$, associated to N, are exactly the finite configurations of $\mathscr{E}_a(\mathscr{U}_a(N))$, as shown in Theorem 3.19. By Lemma 3.66, to prove the thesis it suffices to show that it is possible to define a function $\xi : \langle m \downarrow \mathbf{CP}[N]\rangle \to \mathsf{K}(\mathscr{L}_a(\mathscr{E}_a(\mathscr{U}_a(N))))$ such that f is *surjective*, and for all γ_1, γ_2 in $\langle m \downarrow \mathbf{CP}[N]\rangle$,

$$\gamma_1 \lesssim \gamma_2 \qquad \text{iff} \qquad \xi(\gamma_1) \sqsubseteq \xi(\gamma_2).$$

The function ξ can be defined as follows. Let $\gamma = \langle \mu, \varphi, \nu\rangle$ be a concatenable process in $\langle m \downarrow \mathbf{CP}[N]\rangle$. Being φ a marked process of N (and thus a c-net morphism $\varphi : O_\varphi \to N$), by

the universal property of coreflections, there exists a unique arrow $\varphi' : O_\varphi \to \mathscr{U}_a(N)$, making the diagram below commute.

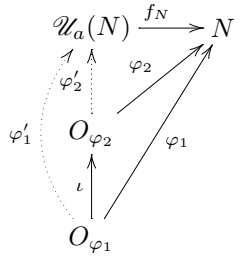


In other words, the coreflection between **SW-CN** and **O-CN** gives a one-to-one correspondence between the (marked) processes of N and of those of its unfolding $\mathscr{U}_a(N)$.

Then we define $\xi(\gamma) = \varphi'_T(T_\varphi)$, where T_φ is the set of transitions of O_φ. To see that ξ is a well defined function, just observe that it could have been written, more precisely, as $\mathscr{E}_a(\mathscr{U}_a(\varphi))(T_\varphi)$ and T_φ is a configuration of $\mathscr{E}_a(\mathscr{U}_a(O_\varphi)) = \mathscr{E}_a(O_\varphi)$ since O_φ is a deterministic occurrence c-net.

- ξ is *surjective*
  Let $C \in \mathsf{K}(\mathscr{L}_a(\mathscr{E}_a(\mathscr{U}_a(N))))$ be a finite configuration. Then C determines a deterministic process $\varphi'_C : O_{\varphi'_C} \to \mathscr{U}_a(N)$ of the unfolding of N, having C as set of transitions.[8] Thus $\varphi = f_N \circ \varphi'_C$ is a deterministic process of N, and, by the definition of ξ, we immediately get that $\xi(\varphi) = \varphi'_C(T_{\varphi'_C}) = C$.

- ξ is *monotone*
  Let γ_1 and γ_2 be processes in $\langle m \downarrow \mathbf{CP}[N]\rangle$ and let $\gamma_1 \lesssim \gamma_2$. Then, by Lemma 3.64 there exists a left-injection $\iota : \gamma_1 \to \gamma_2$. The picture below illustrates the situation, by depicting also the processes φ'_1 and φ'_2 of the unfolding of N, induced by φ_1 and φ_2, respectively.



  We have that $\xi(\gamma_1) = \varphi'_1(T_{\varphi_1}) = \varphi'_2(\iota(T_{\varphi_1})) \subseteq \varphi'_2(T_{\varphi_2}) = \xi(\gamma_2)$. Therefore, to conclude that $\xi(\gamma_1) \sqsubseteq \xi(\gamma_2)$ we must show that also the second condition of Definition 3.13 is satisfied. Let $t_2 \in \xi(\gamma_2)$ and $t_1 \in \xi(\gamma_1)$, with $t_2 \nearrow t_1$. By definition of ξ, $t_i = \varphi_i(t'_i)$ with t'_i in O_{φ_i}, for $i \in \{1, 2\}$ and thus:

$$\varphi'_2(t'_2) \nearrow \varphi'_1(t'_1) = \varphi'_2(\iota(t'_1))$$

  By properties of occurrence net morphisms (Corollary 3.43 and the fact that O_{φ_2} is deterministic), this implies $t'_2 \nearrow \iota(t'_1)$ and thus, being ι a left injection, by rigidity $t'_2 = \iota(t)$ for some t in O_{φ_1}. Therefore $t_2 = \varphi'_2(t'_2) = \varphi'_2(\iota(t)) = \varphi'_1(t)$ belongs to $\xi(\gamma_1)$, as desired.

[8] Essentially $O_{\varphi'_C}$ is the obvious subnet of $\mathscr{U}_a(N)$ having C as set of transitions and φ'_C is an inclusion.

- $\xi(\gamma_1) \sqsubseteq \xi(\gamma_2)$ implies $\gamma_1 \lesssim \gamma_2$.
  Let $\xi(\gamma_1) \sqsubseteq \xi(\gamma_2)$. The inclusion $\xi(\gamma_1) \subseteq \xi(\gamma_2)$, immediately induces a mapping ι of the transitions of O_{φ_1} into the transitions of O_{φ_2}, defined by $\iota(t_1) = t_2$ if $\varphi'_1(t_1) = \varphi'_2(t_2)$ (see the picture above). This function is well-defined since processes are deterministic and thus morphisms φ'_i are injective. Since the initial marking of N is a set, the mapping of $\min(\varphi_1)$ into $\min(\varphi_2)$ is uniquely determined and thus ι uniquely extends to places becoming a (marked) process morphism between φ_1 and φ_2. Again for the fact that N is semi-weighted (and thus there exists a unique indexing for the minimal places of each process starting from the initial marking) such morphism is consistent with the indexing of minimal places. Finally, ι is rigid. In fact, let $t_2 \nearrow \iota(t_1)$, for t_1 in O_{φ_1} and t_2 in O_{φ_2}. By definition of c-net morphism (Definition 3.26), recalling that φ_2 is total both on places and transitions, we deduce $\varphi'_2(t_2) \nearrow \varphi'_2(\iota(t_1))$. The way ι is defined implies that $\varphi'_2(\iota(t_1)) = \varphi'_1(t_1)$, and thus

  $$\varphi'_2(t_2) \nearrow \varphi'_1(t_1).$$

  Since $\varphi'_i(t_i) \in \xi(\gamma_i)$ for $i \in \{1, 2\}$, by definition of the order on configurations, we immediately have that $\varphi'_2(t_2) \in \xi(\gamma_1)$, thus there is t'_1 in O_{φ_1} such that $\varphi'_1(t'_1) = \varphi'_2(t_2)$, and thus $\iota(t'_1) = t_2$.

  By lemma 3.64, the existence of the left injection $\iota : \gamma_1 \to \gamma_2$, implies $\gamma_1 \lesssim \gamma_2$.

  □

Chapter 4

Semantics of Inhibitor Nets

The work developed in the previous chapter for contextual nets is extended here to nets with contextual and inhibitor arcs, called *inhibitor nets*, where transitions can check both for the presence and the absence of tokens in the places of the net. To deal with the non-monotonic features introduced by the presence of inhibitor arcs, we define *inhibitor event structures* (or IES for short), a new event structure model which properly generalizes AES's. In such structures a relation, called *disabling-enabling relation*, allows one to model, in a compact way, the presence of disjunctive conflicting causes and the situations of relative atomicity of pairs of events, determined by inhibitor arcs. The same relation permits to represent, as particular cases, also the relations of causality and asymmetric conflict already present in asymmetric event structures.

Following the general methodology outlined in the INTRODUCTION, the truly concurrent semantics for inhibitor nets is given via a chain of functors leading from the category **SW-IN** of semi-weighted inhibitor nets to the category **Dom** of finitary prime algebraic domains:

$$\mathbf{SW\text{-}IN} \underset{\mathscr{U}_i}{\overset{\mathscr{I}_O}{\underset{\perp}{\hookleftarrow}}} \mathbf{O\text{-}IN} \xrightarrow[\mathscr{E}_i]{} \mathbf{IES} \underset{\mathscr{L}_i}{\overset{\mathscr{P}_i}{\underset{\perp}{\leftarrow}}} \mathbf{Dom} \underset{\mathscr{P}}{\overset{\mathscr{L}}{\underset{\sim}{\leftarrow}}} \mathbf{PES}$$

The unfolding and its characterization as a universal construction are "lifted" from contextual to inhibitor nets. Roughly speaking, to construct the unfolding of an inhibitor net N we unfold the contextual net obtained from N by forgetting the inhibitor arcs and then we enrich the resulting occurrence contextual net by adding again the original inhibitor arcs. In this way the unfolding remains decidable, the price to pay being the presence of non-executable events in the resulting occurrence inhibitor net. The unfolding can be naturally abstracted to an IES and then to a prime algebraic domain. The main difference with respect to contextual nets is the absence of a functor performing the backward step from IES's to occurrence inhibitor nets. We argue that, under reasonable assumptions on the notions of occurrence net and of unfolding, the problem has no solutions, being basically related to the the complex kind of causality expressible in IES's.

Finally, the above semantics is shown to fit nicely with some of the deterministic process semantics proposed in the literature for inhibitor nets. We compare the two approaches by characterizing the domain associated to the unfolding of a net as the partial order of processes starting from its initial marking. To this aim a small gap existing between the process semantics of ordinary and inhibitor nets has to be filled with the introduction of an appropriate notion of *concatenable* process for inhibitor nets.

The rest of the chapter is organized as follows. Section 4.1 motivates and defines inhibitor event structures, by showing that they properly generalize asymmetric event structures. Section 4.2 studies the relation between IES's and domains, which is formalized as a categorical coreflection between **IES** and **Dom**. Section 4.3 presents the category of inhibitor nets and focuses on the subcategory **SW-IN** of (semi-weighted) inhibitor nets which we shall work with. Section 4.4 introduces the category of occurrence inhibitor nets and generalizes the unfolding construction from contextual to inhibitor nets. Section 4.5 shows how the unfolding can be abstracted to an IES semantics. Finally Section 4.6 discusses the relation between the unfolding and the deterministic process semantics of inhibitor nets.

4.1 Inhibitor event structures

In the previous chapter, to model in a direct way the behaviour of contextual nets, we generalized prime event structures by replacing the symmetric conflict with an asymmetric conflict relation. Such a feature is obviously still necessary to be able to model the dependencies arising between events in i-nets, but, as observed in the INTRODUCTION, the nonmonotonic features related to the presence of inhibitor arcs (negative conditions) makes the situation far more complicated.

First if a place s is in the post-set of a transition t', in the inhibitor set of t and in the pre-set of t_0 (see the net N_0 in Figure 4.1), then the execution of t' inhibits the firing of t, which can be enabled again by the firing of t_0. Thus t can fire before or after the "sequence" $t'; t_0$, but not in between the two transitions. Roughly speaking there is a sort of atomicity of the sequence $t'; t_0$ with respect to t. The situation can be more involved since many transitions $t_0, \ldots, t_n$ may have the place s in their pre-set (see the net N_1 in Figure 4.1). Therefore, after t' has been fired, t can be re-enabled by any of the conflicting transitions $t_0, \ldots, t_n$. This leads to a sort of *or*-causality, but only when t fires after t'. With a logical terminology we can say that t causally depends on the implication $t' \Rightarrow t_0 \vee t_1 \vee \ldots \vee t_n$.

To face these additional complications we introduce *inhibitor event structures* (IES's), which enrich asymmetric event structures with a ternary relation, called *DE-relation (disabling-enabling relation)*, denoted by $\mapsto(\cdot,\cdot,\cdot)$. Such a relation is used to model the dependency between transitions in N_1 as $\mapsto(\{t'\}, t, \{t_0, \ldots, t_n\})$. The first argument of the relation can be a singleton or also the empty set $\emptyset$, $\mapsto(\emptyset, e, A)$

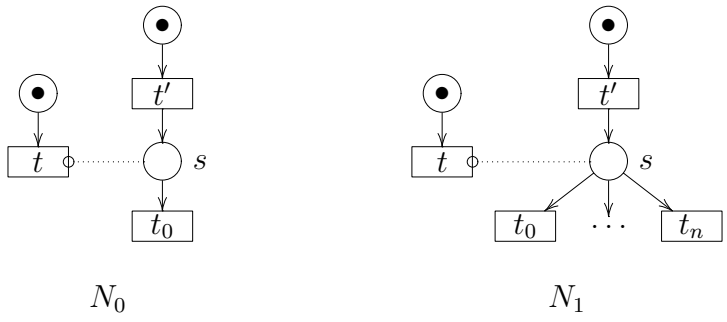


Figure 4.1: Two basic inhibitor nets.

meaning that the event e is inhibited in the initial state of the system. Equivalently $\emptyset$ can be thought of as a special event that must precede any other event of the system. Moreover the third argument (the set of events A) can be empty $\mapsto(\{e'\}, e, \emptyset)$ meaning that there are no events that can re-enable e' after it has been disabled by e. The *DE-relation* is sufficient to represent both causality and asymmetric conflict and thus concretely it is the only relation of a IES.

4.1.1 Inhibitor event structures and their dependency relations

Before giving the formal definitions, let us fix some notational conventions. Recall that the powerset of a set X is denoted by $\mathbf{2}^X$, while $\mathbf{2}^X_{fin}$ denotes the set of finite subsets of X and $\mathbf{2}^X_1$ the set of subsets of X of cardinality at most one (singletons and the empty set). In the sequel generic subsets of events will be denoted by upper case letters $A, B, \ldots$, and singletons or empty subsets by $a, b, \ldots$

DEFINITION 4.1 (PRE-INHIBITOR EVENT STRUCTURE)
A pre-inhibitor event structure *(pre-*IES*) is a pair* $I = \langle E, \mapsto \rangle$*, where* E *is a set of* events *and* $\mapsto \subseteq \mathbf{2}^E_1 \times E \times \mathbf{2}^E$ *is a ternary relation called* disabling-enabling relation (DE-relation *for short*).

According to the discussion above, it is not difficult to see that the relation $\mapsto$ is powerful enough to represent both causality and asymmetric conflict.

In fact, if $\mapsto(\emptyset, e, \{e'\})$ then the event e can be executed only after e' has been fired. This is exactly what happens in traditional PES's when e' causes e, or in symbols when $e' < e$. Notice that more generally, if $\mapsto(\emptyset, e, A)$ then we can imagine A as a set of disjunctive causes for e, since at least one of the events in A will appear in every history of the event e; intuitively we can think that e causally depends on $\bigvee A$. This generalization of causality, restricted to the case in which the set A is pairwise conflictual (namely all distinct events in A are in conflict), will be represented in symbols as $A < e$. Notice that the under the assumption that A is

pairwise conflictual, when $A < e$ *exactly one* event in A appears in each history of e. Therefore, in particular, for any event $e' \in A$, if e and e' are executed in the same computation then surely e' must precede e. As discussed in Chapter 2 (Section 2.4), or-causality has been already investigated in general event structures [Win87a], flow event structures [Bou90] and in bundle event structures [Lan92a, Lan92b]. Indeed, if $A < e$ in an IES then A plays exactly the role of the bundle set in bundle event structures.

Furthermore, if $\mapsto(\{e'\}, e, \emptyset)$ then e can never follow e' in a computation since there are no events which can re-enable e after the execution of e'. Instead the converse order of execution is admitted, namely e *can* fire before e'. This situation is naturally interpreted as an *asymmetric conflict* between the two events, and by analogy with the case of asymmetric event structures it is written $e \nearrow e'$. Recall that asymmetric conflict can be also seen as a *weak* form of *causal dependency*, in the sense that if $e \nearrow e'$ then e precedes e' in all computations containing both events. This explains why we impose asymmetric conflict to include (also generalized) causality, by asking that $A < e$ implies $e' \nearrow e$ for all $e' \in A$.

Finally, as for AES's, cycles of asymmetric conflict are used to define a notion of conflict on sets of events. If $e_0 \nearrow e_1 \ldots e_n \nearrow e_0$ then not all such events can appear in the same computation, a fact that is formalized via a conflict relation on sets of events $\#\{e_0, e_1 \ldots, e_n\}$. In particular, binary (symmetric) conflict is represented by asymmetric conflict in both directions.

The intended meaning of the relations $<$, $\nearrow$ and $\#$ is summarized below.

$A < e$	means that in every computation where e is executed, there is exactly one event $e' \in A$ which is executed and it precedes e;
$e' \nearrow e$	means that in every computation where both e and e' are executed, e' precedes e;
$\#A$	means that there are no computations where all events in A are executed.

We are now ready to present the definition of the dependency relations in a pre-IES, making precise their informal interpretation discussed above.

DEFINITION 4.2 (DEPENDENCY RELATIONS)
*Let $I = \langle E, \mapsto \rangle$ be a pre-*IES*. The relations of* (generalized) causality $< \subseteq \mathbf{2}^E \times E$, asymmetric conflict $\nearrow \subseteq E \times E$ *and* conflict $\# \subseteq \mathbf{2}^E_{fin}$ *are defined by the set of rules in Table 4.1, where $\#_p A$ means that the events in A are pairwise conflictual, namely $\#\{e, e'\}$ for all $e, e' \in A$ with $e \neq e'$.*

We will use the infix notation for the binary conflicts, writing $e\#e'$ instead of $\#\{e, e'\}$. Moreover we will write $e < e'$ to indicate $\{e\} < e'$.

The basic rules (< 1), ($\nearrow 1$) and ($\#1$), as well as ($\nearrow 2$) and ($\nearrow 3$) are justified by the discussion above.

$$\frac{\vdash\!\circ(\emptyset, e, A) \quad \#_p A}{A < e} \ (< 1)$$

$$\frac{A < e \quad \forall e' \in A.\ A_{e'} < e' \quad \#_p(\cup\{A_{e'} \mid e' \in A\})}{(\cup\{A_{e'} \mid e' \in A\}) < e} \ (< 2)$$

$$\frac{\vdash\!\circ(\{e'\}, e, \emptyset)}{e \nearrow e'} \ (\nearrow 1) \qquad \frac{e \in A < e'}{e \nearrow e'} \ (\nearrow 2) \qquad \frac{\#\{e, e'\}}{e \nearrow e'} \ (\nearrow 3)$$

$$\frac{e_0 \nearrow \ldots \nearrow e_n \nearrow e_0}{\#\{e_0, \ldots, e_n\}} \ (\#1) \qquad \frac{A' < e \quad \forall e' \in A'.\ \#(A \cup \{e'\})}{\#(A \cup \{e\})} \ (\#2)$$

Table 4.1: Causality, asymmetric conflict and conflict in a pre-IES.

Rule (< 2) generalizes the transitivity of the causality relation in AES's. If $A < e$ and for every event $e' \in A$ we can find a set of events $A_{e'}$ such that $A_{e'} < e'$, then the union of all such sets, namely $\cup\{A_{e'} \mid e' \in A\}$, can be seen as (generalized) cause of e, provided that it is pairwise conflictual. Observe that in particular, if $\{e'\} < e$ and $\{e''\} < e'$ then $\{e''\} < e$.

Rule $(\#2)$ expresses a kind of hereditarity of the conflict with respect to causality, generalizing the inductive definition of the conflict in AES's (see Definition 3.2). Suppose $A' < e$ and that any event $e' \in A'$ is in conflict with A, namely $\#(A \cup \{e'\})$ for each $e' \in A$. Since by definition of $<$ the execution of e must be preceded by an event in A' we can conclude that also e is in conflict with A, i.e., $\#(A \cup \{e\})$. In particular by taking $A' = \{e'\}$ and $A = \{e''\}$ we obtain that if $\{e'\} < e$ and $\#\{e', e''\}$ then $\#\{e, e''\}$.

Notice that, due to the greater generality of IES's, the rules defining the dependency relations are more involved than in AES's, and it is not possible to give a separate definition of the various relations. In fact, according to rules (< 1) and (< 2) one can derive $A' < e$ only provided that the events in A' are pairwise conflictual. Asymmetric conflict is in turn induced both by generalized causality (rule $(\nearrow 2)$) and by conflict (rule $(\nearrow 3)$). Finally, the conflict relation is defined by using the asymmetric conflict (rule $(\#1)$) and it is inherited along causality (rule $(\#2)$).

We will see in the following that, as expected, inhibitor event structures properly generalize asymmetric event structures; moreover, when applied to (the encoding into IES's of) asymmetric event structures the above rules induce the usual relations of causality and (asymmetric) conflict.

It is worth observing that in an IES we can have $\vdash\!\circ(\emptyset, e, \emptyset)$, meaning that the event e can never be executed. Indeed, observe that, by rule (< 1), we deduce $\emptyset < e$ and thus, by rule $(\#2)$, we have $\#\{e\}$, namely the event e is in conflict with itself.

REMARK 4.3
The set of rules defining the relations of (generalized) causality, asymmetric conflict and conflict are not intended to be complete in any sense. In fact, a natural notion of completeness would require the conflict relation to capture precisely the non-executability of events, namely that if e is not executable in any computation then $\#\{e\}$. But the correspondence between IES's and i-nets suggests that the problem of establishing if an event does not appear in any computation is not semidecidable. Thus trying to define a proof system complete with respect to the computational properties of IES's seems hopeless.

An inhibitor event structure is defined as a pre-IES where events related by the DE-relation satisfy a few further requirements suggested by the intended meaning of such relation. Furthermore the causality and asymmetric conflict relations must be induced "directly" by the DE-relation. This will allow us to have a simpler notion of IES-morphism.

DEFINITION 4.4 (INHIBITOR EVENT STRUCTURE)
An inhibitor event structure (IES) *is a pre-*IES $I = \langle E, \mapsto \rangle$ *satisfying, for all* $e \in E$, $a \in \mathbf{2}_1^E$ *and* $A \subseteq E$,

1. *if* $\mapsto(a, e, A)$ *then* $\#_p A$ *and* $\forall e' \in a.\ \forall e'' \in A.\ e' < e''$;
2. *if* $A < e$ *then* $\mapsto(\emptyset, e, A)$;
3. *if* $e \nearrow e'$ *then* $\mapsto(\{e'\}, e, \emptyset)$.

It is easy to see that given a pre-IES I satisfying only (1) it is always possible to "saturate" the relation $\mapsto$ in order to obtain an IES where the relations of causality and (asymmetric) conflict are exactly the same as in I.

PROPOSITION 4.5 (SATURATION OF A PRE-IES)
Let $I = \langle E, \mapsto \rangle$ *be a pre-*IES *satisfying condition (1) of Definition 4.4. Then* $\overline{I} = \langle E, \mapsto' \rangle$, *where* $\mapsto' = \mapsto \cup \{(\emptyset, e, A) \mid A < e\} \cup \{(\{e\}, e', \emptyset) \mid e \nearrow e'\}$ *is a* IES. *Moreover the relations of causality, asymmetric conflict and conflict in* $\overline{I}$ *are the same as in* I.

4.1.2 Morphisms of inhibitor event structures

The notion of IES-morphism is introduced as a generalization of AES-morphisms. The requirement of saturation in the definition of IES allows us to formulate the conditions on IES-morphisms in a compact way.

DEFINITION 4.6 (CATEGORY **IES**)
Let $I_0 = \langle E_0, \mapsto_0 \rangle$ and $I_1 = \langle E_1, \mapsto_1 \rangle$ be two IES*'s. An* IES-morphism *$f : I_0 \to I_1$ is a partial function $f : E_0 \to E_1$ such that for all $e_0, e_0' \in E_0$, $A_1 \subseteq E_1$*

1. $(f(e_0) = f(e_0')) \wedge (e_0 \neq e_0') \quad \Rightarrow \quad e_0 \#_0 e_0'$;
2. $A_1 < f(e_0) \quad \Rightarrow \quad \exists A_0 \subseteq f^{-1}(A_1).\ A_0 < e_0$;
3. $\mapsto_1(\{f(e_0')\}, f(e_0), A_1) \quad \Rightarrow \quad \exists A_0 \subseteq f^{-1}(A_1).\ \exists a_0 \subseteq \{e_0'\}.\ \mapsto_0(a_0, e_0, A_0)$.

where it is intended that $f(e_0)$ and $f(e_0')$ are defined. We denote by **IES** *the category of inhibitor event structures and* IES*-morphisms.*

Condition (1) is the usual condition of event structure morphisms which allows one to confuse only conflictual branches of computations. As formally proved later in Proposition 4.10, condition (2) can be seen as a generalization of the requirement of preservation of causes, namely of the property $\lfloor f(e) \rfloor \subseteq f(\lfloor e \rfloor)$, of PES (and AES) morphisms. Finally, condition (3), as it commonly happens for event structures morphisms, just imposes the preservation of computations by asking, whenever some events in the image are constrained in some way, that stronger constraints are present in the pre-image. More precisely suppose that $\mapsto(\{f(e_0')\}, f(e_0), A_1)$. Thus we can have a computation where $f(e_0')$ is executed first and $f(e_0)$ can be executed only after one of the events in A_1. Otherwise the computation can start with the execution of $f(e_0)$. According to condition (3), e_0 and e_0' are subject in I_0 to the same constraint of their images or, when $a_0 = \emptyset$ or $A_0 = \emptyset$, to stronger constraints selecting one of the possible orders of execution. It is worth stressing that, since $A_i < f(e_i)$ can be equivalently expressed as $\mapsto(\emptyset, f(e_i), A_i)$, condition (2) just represents a reformulation of (3) aimed at covering the case in which the first argument of the DE-relation is the empty set.

The next proposition proves some useful properties of IES-morphisms, which are basically generalizations of analogous properties holding in the case of prime and asymmetric event structures. They will help in showing that IES-morphisms are closed under composition and thus category **IES** is well-defined.

PROPOSITION 4.7
Let I_0 and I_1 be IES*'s and let $f : I_0 \to I_1$ be an* IES*-morphism. For any $e_0, e_0' \in E_0$,*

1. *if $f(e_0) < f(e_0')$ then $\exists A_0.\ e_0 \in A_0 < e_0'$ or $e_0 \# e_0'$;*
2. *if $f(e_0) \nearrow f(e_0')$ then $e_0 \nearrow e_0'$.*

PROOF.

1. Let $f(e_0) < f(e_0')$, namely $\{f(e_0)\} < f(e_0')$. By condition (2) in the definition of IES-morphisms, there exists $A_0 \subseteq f^{-1}(\{f(e_0)\})$ such that $A_0 < e_0'$. Now, if $e_0 \in A_0$ the desired property is proved. Otherwise for each $e_0'' \in A_0$, $e_0'' \neq e_0$ and, by construction $f(e_0'') = f(e_0)$. Hence by condition (1) in the definition of IES-morphism, it must be $e_0 \# e_0''$ for each $e_0'' \in A_0$. Hence, by rule (#2), we conclude $e_0 \# e_0'$.

2. Let $f(e_0) \nearrow f(e_0')$. Then, by definition of IES, $\vdash\!\circ(\{f(e_0')\}, f(e_0), \emptyset)$. By condition (3) in the definition of IES-morphism there must exist $a_0 \subseteq \{e_0'\}$ and $A_0 \subseteq f^{-1}(\emptyset) = \emptyset$ such that $\vdash\!\circ(a_0, e_0, A_0)$. Therefore, if $a_0 = \{e_0'\}$ then $\vdash\!\circ(\{e_0'\}, e_0, \emptyset)$ and thus, by rule $(\nearrow 1)$, we conclude $e_0 \nearrow e_0'$. If instead $a_0 = \emptyset$ then $\vdash\!\circ(\emptyset, e_0, \emptyset)$ and thus, by rule (< 1), $\emptyset < e_0$. Hence, by rule $(\#2)$, we deduce $\#\{e_0, e_0'\}$ and thus $e_0 \nearrow e_0'$ by $(\nearrow 3)$. □

PROPOSITION 4.8
The IES*-morphisms are closed under composition.*

PROOF. Let $f_0 : I_0 \to I_1$ and $f_1 : I_1 \to I_2$ be IES-morphisms. We want to show that their composition $f_1 \circ f_0$ still satisfies conditions (1)-(3) of Definition 4.6.

1. Let $e_0, e_0' \in E_0$ be events such that $e_0 \neq e_0'$ and $f_1(f_0(e_0)) = f_1(f_0(e_0'))$. If $f_0(e_0) = f_0(e_0')$ then, being f_0 a morphism, $e_0 \# e_0'$. Otherwise, since also f_1 is a morphism, $f_0(e_0) \# f_0(e_0')$ and thus, by rule $(\nearrow 3)$, $f_0(e_0) \nearrow f(e_0') \nearrow f(e_0)$. Hence, by Proposition 4.7.(2), it must hold $e_0 \nearrow e_0' \nearrow e_0$, which in turn, by rule $(\#1)$ allows us to deduce $e_0 \# e_0'$.

2. Consider $A_2 \subseteq E_2$ and $e_0 \in E_0$ such that $A_2 < f_1(f_0(e_0))$. Since f_1 is an IES-morphism there exists $A_1 \subseteq f_1^{-1}(A_2)$ such that $A_1 < f_0(e_0)$. By using again condition (2) in the definition of IES-morphism, applied to f_0, we obtain the existence of $A_0 \subseteq f_0^{-1}(A_1)$ satisfying $A_0 < e_0$. We conclude observing that $A_0 \subseteq f_0^{-1}(A_1) \subseteq f_0^{-1}(f_1^{-1}(A_2)) = (f_1 \circ f_0)^{-1}(A_2)$.

3. Let us assume $\vdash\!\circ(\{f_1(f_0(e_0'))\}, f_1(f_0(e_0)), A_2)$. By definition of IES-morphism there exist $A_1 \subseteq f_1^{-1}(A_2)$ and $a_1 \subseteq \{f_0(e_0')\}$ such that $\vdash\!\circ(a_1, f_0(e_0), A_1)$. We can distinguish two cases according to the form of a_1.

   - If $a_1 = \emptyset$ and thus $A_1 < f_0(e_0)$, by definition of IES-morphism there will be $A_0 \subseteq f_0^{-1}(A_1)$ such that $A_0 < e_0$. By definition of IES this implies $\vdash\!\circ(\emptyset, e_0, A_0)$. Moreover $A_0 \subseteq f_0^{-1}(A_1) \subseteq f_0^{-1}(f_1^{-1}(A_2))$ and thus condition (3) is satisfied.
   - If $a_1 = \{f_0(e_0')\}$ and thus $\vdash\!\circ(\{f_0(e_0')\}, f_0(e_0), A_1)$ reasoning as above, but using point (3) in the definition of morphism, we deduce the existence of $A_0 \subseteq f_0^{-1}(A_1) \subseteq f_0^{-1}(f_1^{-1}(A_2))$ and $a_0 \subseteq \{e_0\}$ such that $\vdash\!\circ(a_0, e_0, A_0)$, thus satisfying condition (3).

□

We conclude this subsection with a technical lemma, which will be useful later. It gives some sufficient conditions for a function between pre-IES's to be a well-defined IES-morphism between the IES's obtained by saturating the original pre-IES's.

LEMMA 4.9
Let $I_i = \langle E_i, \vdash\!\circ_i \rangle$ $(i \in \{0, 1\})$ *be pre-*IES*'s satisfying condition (1) of Definition 4.4, let* $\overline{I_i} = \langle E_i, \vdash\!\circ_i^s \rangle$*, and let* $<_i$*,* $\nearrow_i$ *and* $\#_i$ *be the relations of causality, asymmetric conflict and conflict in* I_i*. Let* $f : E_0 \to E_1$ *be a partial function such that for each* $e_0, e_0' \in E_0$ *and* $A_1 \subseteq E_1$*:*

1. $f(e_0) = f(e_0') \ \wedge \ e_0 \neq e_0' \quad \Rightarrow \quad e_0 \#_0 e_0'$;
2. $\vdash\!\circ_1(\emptyset, f(e_0), A_1) \quad \Rightarrow \quad \exists A_0 \subseteq f^{-1}(A_1).\ A_0 <_0 e_0$;

3. $\vdash\!\!\circ_1(f(e_0'), f(e_0), \emptyset) \quad \Rightarrow \quad e_0 \nearrow_0 e_0'$;

4. $\vdash\!\!\circ_1(\{f(e_0')\}, f(e_0), A_1) \wedge A_1 \neq \emptyset \Rightarrow \exists A_0 \subseteq f^{-1}(A_1).\, \exists a_0 \subseteq \{e_0'\}.\, \vdash\!\!\circ_0^s(a_0, e_0, A_0)$.

Then $f : \overline{I_0} \to \overline{I_1}$ *is an* IES*-morphism.*

PROOF. We first show that f satisfies the following properties:

a. $A_1 <_1 f(e_0) \quad \Rightarrow \quad \exists A_0 \subseteq f^{-1}(A_1).\, A_0 <_0 e_0$;

b. $f(e_0) \nearrow_1 f(e_0') \quad \Rightarrow \quad e_0 \nearrow_0 e_0'$.

c. $\#_1 f(A_0) \quad \Rightarrow \quad \#_0 A_0$.

The three points are proved simultaneously by induction on the height of the derivation of the judgement, involving the relations $<_1$, $\nearrow_1$ and $\#_1$, which appears in the premise of each implication and by cases on the form of the judgement.

a. Judgement $A_1 <_1 f(e_0)$.
We distinguish various subcases according to the last rule used in the derivation:

(< 1) Let the last rule be

$$\frac{\vdash\!\!\circ_1(\emptyset, f(e_0), A_1) \quad \#_p A_1}{A_1 <_1 f(e_0)} \quad (< 1)$$

In this case, since $\vdash\!\!\circ_1(\emptyset, f(e_0), A_1)$, we immediately conclude by using point (2) in the hypotheses.

(< 2) Let the last rule be

$$\frac{A_1 <_1 f(e_0) \quad \forall e_1 \in A_1.\, A_{e_1} <_1 e_1 \quad \#_p(\cup\{A_{e_1} \mid e_1 \in A_1\})}{(\cup\{A_{e_1} \mid e_1 \in A_1\}) <_1 f(e_0)} \quad (< 2)$$

By inductive hypothesis from $A_1 <_1 f(e_0)$ we deduce that

$$\exists A_0 \subseteq f^{-1}(A_1).\, A_0 <_0 e_0 \qquad (\dagger)$$

Now, for all $e_0' \in A_0$, by ($\dagger$), $f(e_0') \in A_1$. Therefore, by the second premise of the rule above, $A_{f(e_0')} <_1 f(e_0')$, and thus, by inductive hypothesis, there exists $A_{e_0'} \subseteq f^{-1}(A_{f(e_0')})$ such that $A_{e_0'} <_0 e_0'$. Finally, $\cup\{A_{e_0'} \mid e_0' \in A_0\}$ is pairwise conflictual. In fact if $e_0^1, e_0^2 \in \cup\{A_{e_0'} \mid e_0' \in A_0\}$ with $e_0^1 \neq e_0^2$, we have $f(e_0^1), f(e_0^2) \in \bigcup_{e_1 \in A_1} A_{e_1}$, which is pairwise conflictual. Therefore $f(e_0^1) = f(e_0^2)$ or $f(e_0^1) \#_1 f(e_0^2)$ and, by using point (1) in the hypotheses in the first case, and by inductive hypothesis in the second case, we conclude $e_0^1 \#_0 e_0^2$.
By using the facts proved so far we can apply rule (< 2) as follows:

$$\frac{A_0 <_0 e_0 \quad \forall e_0' \in A_0.\, A_{e_0'} <_0 e_0' \quad \#_p(\cup\{A_{e_0'} \mid e_0' \in A_0\})}{(\cup\{A_{e_0'} \mid e_0' \in A_0\}) <_0 e_0} \quad (< 2)$$

This concludes the proof of this case since

$$\begin{aligned} &\cup\{A_{e_0'} \mid e_0' \in A_0\} \subseteq \\ &\quad \subseteq \cup\{f^{-1}(A_{f(e_0')}) \mid e_0' \in A_0\} \\ &\quad \subseteq \{f^{-1}(A_{e_1}) \mid e_1 \in A_1\} \\ &\quad = f^{-1}(\cup\{A_{e_1} \mid e_1 \in A_1\}) \end{aligned}$$

b. Judgement $f(e_0) \nearrow_1 f(e_0')$.
We distinguish various subcases according to the last rule used in the derivation:

($\nearrow$ 1) Let the last rule be

$$\frac{\vdash\!\circ_1(\{f(e_0')\}, f(e_0), \emptyset)}{f(e_0) \nearrow_1 f(e_0')} \quad (\nearrow 1)$$

From $\vdash\!\circ_1(\{f(e_0')\}, f(e_0), \emptyset)$, by point (3) in the hypotheses, we immediately have that $e_0' \nearrow_0 e_0$.

($\nearrow$ 2) Let the last rule be

$$\frac{f(e_0) \in A_1 <_1 f(e_0')}{f(e_0) \nearrow_1 f(e_0')} \quad (\nearrow 2)$$

By inductive hypothesis there exists $A_0 \subseteq f^{-1}(A_1)$ such that $A_0 <_0 e_0'$.
For all $e_0'' \in A_0$, we have $f(e_0'') \in A_1$. Thus recalling that, since $A_1 <_1 f(e_0')$, the set A_1 is pairwise conflictual, it follows that $f(e_0'') = f(e_0)$ or $f(e_0'')\#_1 f(e_0)$. By using point (1) of the hypotheses in the first case and the inductive hypothesis in the second case, we can conclude that for all $e_0'' \in A_0$, $e_0 = e_0''$ or $e_0 \#_0 e_0''$.
Consequently there are two possibilities. One is that $e_0 = e_0'' \in A_0$ for some $e_0'' \in A_0$, which allows us to conclude since $A_0 <_0 e_0'$. The other one is that $e_0 \#_0 e_0''$ for all $e_0'' \in A_0$. Thus, by rule (#2), we can derive that $\#_0\{e_0, e_0'\}$, and therefore $e_0 \nearrow_0 e_0'$ by rule ($\nearrow$ 3).

($\nearrow$ 3) Let the last rule be

$$\frac{\#_1\{f(e_0), f(e_0')\}}{f(e_0) \nearrow_1 f(e_0')} \quad (\nearrow 3)$$

In this case by inductive hypothesis $\#_0\{e_0, e_0'\}$ and therefore, by rule ($\nearrow$ 3), $e_0 \nearrow_0 e_0'$.

c. Judgement $\#_1 f(A_0)$.
We distinguish various subcases according to the last rule used in the derivation:

(#1) Let the last rule be

$$\frac{f(e_0^{(0)}) \nearrow_1 \ldots \nearrow_1 f(e_0^{(n)}) \nearrow_1 f(e_0^{(0)})}{\#_1\{f(e_0^{(0)}), \ldots, f(e_0^{(n)})\}} \quad (\#1)$$

where $A_0 = \{e_0^{(0)}, \ldots, e_0^{(n)}\}$. By inductive hypothesis $e_0^{(0)} \nearrow_0 \ldots \nearrow_0 e_0^{(n)} \nearrow_0 e_0^{(0)}$, and therefore $\#A_0$.

(#2) Let the last rule be

$$\frac{A_1 <_1 f(e_0) \quad \forall e_1 \in A_1.\ \#_1(f(A_0') \cup \{e_1\})}{\#_1(f(A_0') \cup \{f(e_0)\})} \quad (\#2)$$

where $A_0 = A_0' \cup \{e_0\}$.
By inductive hypothesis, from $A_1 <_1 f(e_0)$ it follows that

$$\exists A_0'' \subseteq f^{-1}(A_1).\ A_0'' <_0 e_0 \qquad (\dagger)$$

Now, for all $e_0' \in A_0''$, by ($\dagger$), $f(e_0') \in A_1$. Therefore, by the second premise of the rule above, $\#_1(f(A_0') \cup \{f(e_0')\})$, namely $\#_1 f(A_0' \cup \{e_0'\})$. Thus, by inductive hypothesis, $\#_0(A_0' \cup \{e_0'\})$ for all $e_0' \in A_0''$. Recalling that $A_0'' <_0 e_0$, by using rule (#2), we obtain

$$\frac{A_0'' <_0 e_0 \quad \forall e_0' \in A_0''.\ \#_0(A_0' \cup \{e_0'\})}{\#_0(A_0' \cup \{e_0\})} \quad (\#2)$$

which is the desired result.

This completes the proof of the properties (a), (b) and (c).

It is now easy to conclude that $f : \overline{I_0} \to \overline{I_1}$ is a IES-morphism. Let $\overline{I_i} = \langle E_i, \mapsto_i^s \rangle$ for $i \in \{1, 2\}$. Conditions (1) and (2) of the definition of IES-morphism (Definition 4.6) are clearly satisfied. In fact, by Proposition 4.5 the relations of causality and conflict in I_i and $\overline{I_i}$ coincide, and thus the mentioned conditions coincide with point (1) in the hypotheses and point (a) proved above.

Hence it remains to verify condition (3) of Definition 4.6, that is

$$\mapsto_1^s(\{f(e_0')\}, f(e_0), A_1) \quad \Rightarrow \quad \exists A_0 \subseteq f^{-1}(A_1).\ \exists a_0 \subseteq \{e_0'\}.\ \mapsto_0^s(a_0, e_0, A_0).$$

Suppose that $\mapsto_1^s(\{f(e_0')\}, f(e_0), A_1)$. If $A_1 \neq \emptyset$, by definition of $\overline{I_i}$, it must be $\mapsto_1(\{f(e_0')\}, f(e_0), A_1)$ and thus the thesis trivially holds by point (4) in the hypotheses. If instead $A_1 = \emptyset$ then, by rule ($\nearrow$ 1), $f(e_0) \nearrow_1 f(e_0')$. Hence, by point (b) proved above, $e_0 \nearrow_0 e_0'$ and therefore $\mapsto_0^s(\{e_0'\}, e_0, \emptyset)$, which satisfies the desired condition. □

4.1.3 Relating asymmetric and inhibitor event structures

The category **AES** of asymmetric event structures can be viewed as a full subcategory of **IES**. This result substantiates the claim according to which IES's (and constructions on them) are a proper "conservative" extension of AES's and thus of PES's.

PROPOSITION 4.10 (FROM AES'S TO IES'S)
Let $\mathscr{J}_a : \mathbf{AES} \to \mathbf{IES}$ be the functor defined as follows. To any AES $G = \langle E, \leq, \nearrow \rangle$ *the functor $\mathscr{J}_a$ associates the* IES $I_G = \langle E, \mapsto \rangle$ *where*

$$\mapsto(\emptyset, e, \{e''\}) \text{ if } e'' < e \qquad \text{and} \qquad \mapsto(\{e'\}, e, \emptyset) \text{ if } e \nearrow e'.$$

and for any $f : G_1 \to G_2$ its image $\mathscr{J}_a(f)$ is f itself. Then the functor $\mathscr{J}_a$ is a full embedding of **AES** *into* **IES**.

PROOF. We start noticing that, given an AES $G = \langle E, \leq, \nearrow \rangle$, if $<_{I_G}$, $\nearrow_{I_G}$ and $\#_{I_G}$ denote the dependency relations in I_G then

- $A <_{I_G} e' \quad$ iff $\quad A = \{e\} \ \wedge\ e < e'$;
- $e \nearrow_{I_G} e' \quad$ iff $\quad e \nearrow e'$; (†)
- $\#_{I_G} A \quad$ iff $\quad \#A$;

Since the causality, asymmetric conflict and conflict relations of G_i and I_{G_i} coincide, in the following they will be denoted by the same symbols $<$, $\nearrow$ and $\#$.

By using the characterization (†) of the dependency relations in I_G it is easy to prove that the functor $\mathscr{J}_a$ is well-defined. First, given any AES $G = \langle E, \leq, \nearrow \rangle$, the structure I_G satisfies the conditions (1)-(3) of Definition 4.4, and thus I_G is indeed an IES. Observing that if $\mapsto(a, e, A)$, then

either $A = \emptyset$ or $a = \emptyset$, and A contains at most one element we see that condition (1) is trivially satisfied. The validity of conditions (2) and (3) immediately follows from (†).

Moreover, let $f : G_0 \to G_1$ be an AES-morphism. To see that $f : I_{G_0} \to I_{G_1}$ is an IES-morphism first notice that condition (1) of Definition 4.6 is present also in the definition of AES-morphism (Definition 3.4). As for condition (2), if $A_1 < f(e_0)$ then, by (†) above, $A_1 = \{e_1\}$ and $e_1 < f(e_0)$. Hence, by condition (1) in the definition of AES-morphisms, there exists $e_0'' \in E_0$ such that $e_0'' < e_0$ and $e_1 = f(e_0'')$ (thus $e_0'' \in f^{-1}(e_1)$). Finally, also condition (3) is satisfied. In fact, if $\vdash\!\circ(\{f(e_0')\}, f(e_0), A_1)$, according to the definition of I_{G_1} it must be $A_1 = \emptyset$. Hence $f(e_0) \nearrow f(e_0')$ and thus, by definition of AES-morphisms, $e_0 \nearrow e_0'$. Thus, by construction of I_{G_0}, it follows that $\vdash\!\circ(\{e_0'\}, e_0, \emptyset)$ proving (3).

To conclude it remains to show that also the converse holds, namely if $f : I_{G_0} \to I_{G_1}$ is an IES-morphism then $f : G_0 \to G_1$ is an AES-morphism. Let us verify the validity of the three conditions of the definition of AES-morphism (Definition 3.4).

1. $\lfloor f(e_0) \rfloor \subseteq f(\lfloor e_0 \rfloor)$.
   Let $e_1 \in \lfloor f(e_0) \rfloor$, namely $e_1 < f(e_0)$. Thus, by condition (2) in the definition of IES-morphism, there exists $A_0 \subseteq f^{-1}(e_1)$ such that $A_0 <_I e_0$. By the characterization (†) above of the dependency relations in I_{G_i}, this means that $A_0 = \{e_0'\}$ and $e_0' < e_0$, with $f(e_0') = e_1$. Hence $e_1 \in f(\lfloor e_0 \rfloor)$.

2. $f(e_0) = f(e_0') \ \wedge \ e_0 \neq e_0' \quad \Rightarrow \quad e_0 \# e_0'$.
   This condition is trivially satisfied since it is also a condition for AES-morphisms and, as observed above, the conflict relations in A_i and I_{G_i} coincide.

3. $f(e_0) \nearrow f(e_0') \quad \Rightarrow \quad e_0 \nearrow e_0' \ \vee \ e_0 \# e_0'$.
   Immediate by Proposition 4.7.(2). □

Observe that by composing $\mathscr{J}_a$ with the full embedding $\mathscr{J} : \mathbf{PES} \to \mathbf{AES}$ (Proposition 3.9) we obtain a full embedding of **PES** into **IES**.

4.2 From inhibitor event structures to domains

This section establishes a connection between IES's and prime algebraic domains. The relation is expressed as a categorical coreflection between **IES** and **Dom**, which allows one to translate each IES into the domain (or equivalently the PES) which, in a sense, represents its best approximation. Then we study the problem of removing the non-executable events from an IES, by investigating the relation between **IES** and its subcategory $\mathbf{IES}^*$, consisting of the IES's where all events are executable.

4.2.1 The domain of configurations of an IES

As for AES's the domain associated to an IES is obtained by considering the family of its configurations with a suitable order. Since here computations involving the same events may be different from the point of view of causality, a configuration is not uniquely identified as a set of events, but some additional information has to be added which plays a basic role also in the definition of the order on configurations. More concretely, a configuration of an IES is a set of events endowed with a *choice*

relation which chooses among the possible different orders of execution of events constrained by the DE-relation.

Consider a set of events C of an inhibitor event structure I, suppose that $e', e, e'' \in C$ and assume $\mapsto(\{e'\}, e, A)$ for some A, with $e'' \in A$. We already noticed that in this case there are two possible orders of execution of the three events (either $e; e'; e''$ or $e'; e''; e$), which cannot be identified from the point of view of causality. A choice relation for C must choose one of them by specifying that e precedes e' or that e'' precedes e. To ease the definition of the notion of choice relation, we first introduce, for a given set of events C, the set $choices(C)$, a relation on C which "collects" *all* the possible precedences between events induced by the DE-relation. A choice relation for C is then defined as suitable subset of $choices(C)$. To ensure that all the events in the configuration are executable in the specified order, the choice relation is also required to satisfy suitable properties of acyclicity and finitariness.

DEFINITION 4.11
Let $I = \langle E, \mapsto \rangle$ be an IES *and let $C \subseteq E$. We denote by $choices(C)$ the subset of $C \times C$*

$$choices(C) = \{(e, e') \mid \mapsto_C(\{e'\}, e, A)\} \cup \{(e'', e) \mid \mapsto_C(a, e, A) \wedge\ e'' \in A\}.$$

where the restriction of $\mapsto(,,)$ to C is defined by $\mapsto_C(a, e, A)$ if and only if $\mapsto(a, e, A')$, for $e \in C$, $a \subseteq C$ and $A = A' \cap C$.

DEFINITION 4.12 (CHOICE)
Let $I = \langle E, \mapsto \rangle$ be an IES *and let $C \subseteq E$. A* choice *for C is a relation $\hookrightarrow_C \subseteq choices(C)$ such that*

1. *if $\mapsto_C(a, e, A)$ then $\exists e' \in a.\ e \hookrightarrow_C e'$ or $\exists e'' \in A.\ e'' \hookrightarrow_C e$;*
2. *$\hookrightarrow_C$ is* acyclic*;*
3. *$\forall e \in C.\ \{e' \in C : e' \hookrightarrow_C^* e\}$ is* finite.

Condition (1) intuitively requires that whenever the DE-relation permits two possible orders of execution, the relation $\hookrightarrow_C$ chooses one of them. The fact that $\hookrightarrow_C \subseteq choices(C)$ ensures that $\hookrightarrow_C$ does not impose more precedences than necessary. Conditions (2) and (3) guarantee that the precedences specified by $\hookrightarrow_C$ do not give rise to cyclic situations and that each event must be preceded only by finitely many others. Notice that the acyclicity of $\hookrightarrow_C$ ensures that exactly one of the two possible choices in condition (1), namely either $\exists e' \in a.\ e \hookrightarrow_C e'$ or $\exists e'' \in A.\ e'' \hookrightarrow_C e$ is taken. It is worth observing that conditions (2) and (3) can be equivalently rephrased by saying that $\hookrightarrow_C^*$ is a finitary partial order.

Configurations of PES's and AES's are required to be downward closed with respect to causality. The following simple proposition observes that the property of admitting a choice implies a generalization of causal closedness.

PROPOSITION 4.13
Let $I = \langle E, \vdash\!\!\circ \rangle$ be an IES *and let $C \subseteq E$ be a subset of events such that there exists a choice for C. For any $e \in C$, if $A < e$ then $A \cap C \neq \emptyset$.*

PROOF. Observe that if $A < e$, by definition of IES, $\vdash\!\!\circ(\emptyset, e, A)$. Therefore, if $A \cap C = \emptyset$ then we would have $\vdash\!\!\circ_C(\emptyset, e, \emptyset)$. Therefore no relation over C could be a choice, since condition (1) of Definition 4.12 could not be satisfied. □

Another simple but important property is the fact that each choice agrees with the asymmetric conflict relation, in the sense expressed by the following proposition.

PROPOSITION 4.14
For every subset C of an IES *I and for every choice $\hookrightarrow_C$ for C, $\nearrow_C \subseteq \hookrightarrow_C$.*

PROOF. Consider $C \subseteq E$ and $e, e' \in C$. If $e \nearrow e'$ then, by definition of IES, $\vdash\!\!\circ(\{e'\}, e, \emptyset)$ and thus $\vdash\!\!\circ_C(\{e'\}, e, \emptyset)$. Therefore, if $\hookrightarrow_C$ is a choice for C, by condition (1) in Definition 4.12, necessarily $e \hookrightarrow_C e'$. □

It is now easy to verify that a set of events for which a choice exists cannot contain conflicts.

PROPOSITION 4.15
Let $I = \langle E, \vdash\!\!\circ \rangle$ be an IES *and let $C \subseteq E$ be a subset of events such that there exists a choice for C. For any $A \subseteq C$ it is not the case that $\#A$.*

PROOF. Let $A \subseteq C$ and suppose that $\#A$. Then it is easy to show that C contains a cycle of asymmetric conflict, and thus by Proposition 4.14, any choice for C would be cyclic as well, contradicting the definition.

The proof of the fact that if $\#A$ for some $A \subseteq C$ then C contains a cycle of asymmetric conflict proceeds by induction on the height of the derivation of $\#A$. The base case in which the the last rule in the derivation is (#1), namely

$$\frac{e_0 \nearrow \ldots \nearrow e_n \nearrow e_0}{\#\{e_0, \ldots, e_n\}} \quad (\#1)$$

is trivial. Suppose instead that the last rule in the derivation is (#2), namely

$$\frac{A'' < e \quad \forall e' \in A''.\ \#(A' \cup \{e'\})}{\#(A' \cup \{e\})} \quad (\#2)$$

In this case, by Proposition 4.13, there exists $e'' \in A'' \cap C$. Since $\#(A' \cup \{e''\})$ by the second premise of the rule, and $A' \cup \{e''\} \subseteq C$ we conclude by inductive hypothesis. □

A configuration of an IES is now introduced as a set of events endowed with a choice relation. The properties expressed by Propositions 4.13 - 4.15 show how this definition generalizes the notion of AES configuration.

DEFINITION 4.16 (CONFIGURATION)
Let $I = \langle E, \vdash\!\circ \rangle$ be an IES. *A* configuration *of I is a pair $\langle C, \hookrightarrow_C \rangle$, where $C \subseteq E$ is a set of events and $\hookrightarrow_C \subseteq C \times C$ is a* choice *for C.*

In the sequel, with abuse of notation, we will often denote a configuration and the underlying set of events with the same symbol C, referring to the corresponding choice relation as $\hookrightarrow_C$. We already know that the existence of a choice implies the causal closedness and conflict freeness of configurations. Moreover, if C is a configuration, given any $e \in C$ and $A < e$, not only $A \cap C \neq \emptyset$, but since by definition of $<$ necessarily $\#_p A$, we have that $A \cap C$ contains exactly one event. More generally, for the same reason, if C is a configuration and $\vdash\!\circ(a, e, A)$ for some $e \in C$, then $A \cap C$ contains at most one element, and if it is non-empty then $a \subseteq C$. The last assertion is obvious if $a = \emptyset$, while if $a = \{e'\}$ it follows from Proposition 4.13, recalling that $e' < e''$ for all $e'' \in A$.

The next technical proposition shows a kind of maximality property of the choice relation for a configuration. It states that if a choice for C relates two events, then any other choice for C must establish an order between such events. Consequently two compatible choices on the same set of events must coincide.

PROPOSITION 4.17
Let $\langle C_i, \hookrightarrow_{C_i} \rangle$ for $i \in \{1, 2\}$ be configurations of an IES *I.*

1. *If $e, e' \in C_1 \cap C_2$ and $e \hookrightarrow_{C_1} e'$ then $e \hookrightarrow_{C_2} e'$ or $e' \hookrightarrow^*_{C_2} e$.*

2. *If $C_1 = C_2$ and $\hookrightarrow^*_{C_1} \subseteq \hookrightarrow^*_{C_2}$ then $\hookrightarrow_{C_1} = \hookrightarrow_{C_2}$, namely the two configurations coincide.*

PROOF.

1. Let $e, e' \in C_1 \cap C_2$ with $e \hookrightarrow_{C_1} e'$. By definition of choice, it follows that $\vdash\!\circ_{C_1}(\{e'\}, e, A)$ or $\vdash\!\circ_{C_1}(a, e', A')$, with $e \in A'$. Assume that $\vdash\!\circ_{C_1}(\{e'\}, e, A)$ and thus $\vdash\!\circ(\{e'\}, e, A'')$ with $A = A'' \cap C_1$ (the other case can be treated in a similar way). Since $e, e' \in C_2$, $\vdash\!\circ_{C_2}(\{e'\}, e, A'' \cap C_2)$, and thus, by definition of choice, also C_2 must choose among the two possible orders of executions, namely $e \hookrightarrow_{C_2} e'$ or $e'' \hookrightarrow_{C_2} e$ for $e'' \in A'' \cap C_2$. In the second case, since by definition of IES $e' < e''$, by Proposition 4.14, we have $e' \hookrightarrow_{C_2} e''$ and thus $e' \hookrightarrow^*_{C_2} e$.

2. If $e \hookrightarrow_{C_1} e'$, by point (1), $e \hookrightarrow_{C_2} e'$ or $e' \hookrightarrow^*_{C_2} e$. But the second possibility cannot arise, since $e \hookrightarrow_{C_1} e'$ implies $e \hookrightarrow^*_{C_1} e'$ and thus $e \hookrightarrow^*_{C_2} e'$. Vice versa, if $e \hookrightarrow_{C_2} e'$, by point (1), $e \hookrightarrow_{C_1} e'$ or $e' \hookrightarrow^*_{C_1} e$. Again the second possibility cannot arise, otherwise we would have $e' \hookrightarrow^*_{C_2} e$, contradicting the acyclicity of $\hookrightarrow_{C_2}$.

The computational order on configurations is a generalization of the one introduced in the previous chapter for AES's.

DEFINITION 4.18 (EXTENSION)
Let $I = \langle E, \mapsto \rangle$ be an IES *and let C and C' be configurations of I. We say that C'* extends *C and we write $C \sqsubseteq C'$, if*

1. $C \subseteq C'$;
2. $\forall e \in C.\ \forall e' \in C'.\ e' \hookrightarrow_{C'} e \quad \Rightarrow e' \in C$;
3. $\hookrightarrow_C \subseteq \hookrightarrow_{C'}$

The poset of all configurations of I, ordered by extension, is denoted by $Conf(I)$.

The basic idea is still that a configuration C cannot be extended by adding events which are supposed to happen before the events already in C, as expressed by condition (2). Moreover the extension relation takes into account the choice relations of the two configurations, requiring a kind of consistency between them. Condition (3) serves to ensure, together with (2), that the past history of events in C remains the same in C'. Observe that the last condition is not present in AES's, where the choice relation on a configuration is uniquely determined as the restriction of the asymmetric conflict to the set of events in the configuration.

The intuition on the extension relation can be enforced by observing that, as an easy consequence of the definition and of Proposition 4.17.(1), it can be equivalently characterized as $C \sqsubseteq C'$ iff

- $C \subseteq C'$;
- the inclusion $i : \langle C, \hookrightarrow_C^* \rangle \to \langle C', \hookrightarrow_{C'}^* \rangle$ is a rigid embedding, i.e., it is monotone and for $e \in C$, $e' \in C'$, $e' \hookrightarrow_{C'}^* e$ implies $e' \in C$.

Furthermore, it is worth noticing that if $C \sqsubseteq C'$, by Proposition 4.17.(1) we immediately get that $\hookrightarrow_C = \hookrightarrow_{C'} \cap (C \times C)$, and thus the inclusion of C into C' is also *order monic*, namely for each $e, e' \in C$, if $e \hookrightarrow_{C'}^* e'$ then $e \hookrightarrow_C^* e'$. Roughly speaking this means that $C \sqsubseteq C'$ whenever C coincides with a "truncation" of C'.

As in the case of AES's, given an event in a configuration it is possible to define the past history of the event in that configuration as a configuration itself.

DEFINITION 4.19 (HISTORY)
Let I be an IES *and let $C \in Conf(I)$ be a configuration. For any $e \in C$ we define the* history *of e in C as the configuration $\langle C[\![e]\!], \hookrightarrow_{C[\![e]\!]} \rangle$, where $C[\![e]\!] = \{e' \in C \mid e' \hookrightarrow_C^* e\}$ and $\hookrightarrow_{C[\![e]\!]} = \hookrightarrow_C \cap (C[\![e]\!] \times C[\![e]\!])$.*

It is not difficult to see that $\langle C[\![e]\!], \hookrightarrow_{C[\![e]\!]} \rangle$ is a well-defined configuration. The only fact that is not obvious is the validity of condition (1) in the definition of choice (Definition 4.12). Now, if $\mapsto_{C[\![e]\!]}(a, e', A)$ then $\mapsto_C(a, e', A')$ with $a \subseteq C[\![e]\!]$, $e' \in C[\![e]\!]$ and $A = A' \cap C[\![e]\!]$. Being C a configuration, it must be $e' \hookrightarrow_C e_0$ for $e_0 \in a$ or $e_1 \hookrightarrow_C e'$ for some $e_1 \in A'$. In the first case, $e_0 \in a \subseteq C[\![e]\!]$ and thus $e' \hookrightarrow_{C[\![e]\!]} e_0$, while in the

second case, since $e' \in C[\![e]\!]$, by definition of history we must have $e_1 \in C[\![e]\!]$, thus $e_1 \hookrightarrow_{C[\![e]\!]} e'$.

It is worth recalling that by definition the reflexive and transitive closure of a choice is a finitary partial order, and thus each history $C[\![e]\!]$ is a *finite* configuration. Furthermore, it immediately follows from its definition that $C[\![e]\!] \sqsubseteq C$.

From now on the proof of the algebraic properties of the poset of configurations of an IES follows closely the steps already performed in the case of AES's. Hence we will discuss explicitly only the points which are specific to IES's. We start with a characterization of least upper bounds and greatest lower bounds of pairwise compatible sets of configurations.

LEMMA 4.20
Let $X \subseteq Conf(I)$ be a pairwise compatible set of configurations of an IES *I and let $C_1, C_2 \in X$. Then*

1. *if $e \hookrightarrow^*_{C_1} e'$ and $e' \in C_2$ then $e \in C_2$ and $e \hookrightarrow^*_{C_2} e'$;*
2. *if $e \in C_1 \cap C_2$ then $C_1[\![e]\!] = C_2[\![e]\!]$;*
3. *$C_1 \sqcap C_2 = C_1 \cap C_2$, with $\hookrightarrow_{C_1 \cap C_2} = \hookrightarrow_{C_1} \cap \hookrightarrow_{C_2}$;*
4. *the least upper bound of X exists, and it is given by*

$$\bigsqcup X = \langle \bigcup_{C \in X} C, \bigcup_{C \in X} \hookrightarrow_C \rangle.$$

PROOF.

1. Let us first suppose that $e \hookrightarrow_{C_1} e'$ and $e' \in C_2$. Let $C \in X$ be an upper bound for C_1 and C_2, which exists since X is pairwise compatible. From $C_1 \sqsubseteq C$, by definition of extension, we have that $e, e' \in C$ and $e \hookrightarrow_C e'$. Recalling that $C_2 \sqsubseteq C$ and $e' \in C_2$ we deduce $e \in C_2$. Since $e, e' \in C_2 = C_2 \cap C$ and $e \hookrightarrow_C e'$, by Proposition 4.17.(1), it must be $e \hookrightarrow_{C_2} e'$ or $e' \hookrightarrow^*_{C_2} e$. The second possibility cannot arise, otherwise we should have $e' \hookrightarrow^*_C e$, contradicting the acyclicity of $\hookrightarrow_C$. Hence we can conclude $e \hookrightarrow_{C_2} e'$.

   In the general case in which $e \hookrightarrow^*_{C_1} e'$ the desired property is easily derived via an inductive reasoning using the above argument.

2. Immediate consequence of point (1).

3. To show that $\hookrightarrow_{C_1 \cap C_2} = \hookrightarrow_{C_1} \cap \hookrightarrow_{C_2}$ is a choice for $C_1 \cap C_2$, the only non trivial point is the proof of condition (1) of Definition 4.12. Suppose that $\vdash\!\circ_{C_1 \cap C_2}(a, e, A)$, namely $\vdash\!\circ(a, e, A')$ with $a \subseteq C_1 \cap C_2$ and $A = A' \cap (A_1 \cap A_2)$. Hence $\vdash\!\circ_{C_1}(a, e, A' \cap C_1)$ and thus either $e \hookrightarrow_{C_1} e'$ for $e' \in a$ or $e'' \hookrightarrow_{C_1} e$ with $e'' \in A' \cap C_1$. Being C_1 and C_2 compatible, by point (1) it must be $e \hookrightarrow_{C_2} e'$, or $e'' \in A' \cap C_2$ and $e'' \hookrightarrow_{C_2} e$, respectively. Therefore, as desired, $e \hookrightarrow_{C_1 \cap C_2} e'$ or $e'' \in A$ with $e'' \hookrightarrow_{C_1 \cap C_2} e$.

   Hence $C_1 \cap C_2$ is a configuration. Moreover, it is the greatest lower bound of C_1 and C_2 as one can check via a routine verification using point (1).

4. Let us verify that $\hookrightarrow_{\bigcup X} = \bigcup_{C\in X} \hookrightarrow_C$ is a choice for $\bigcup X$. First, it is quite easy to see that $\hookrightarrow_{\bigcup X} = \bigcup_{C\in X} \hookrightarrow_C \subseteq \mathit{choices}(\bigcup X)$.

   As for condition (1) of the definition of choice, suppose that $\vdash\!\circ_{\bigcup X}(a, e, A)$, namely $\vdash\!\circ(a, e, A')$ with $a \subseteq \bigcup X$ and $A = A' \cap \bigcup X$. Since $a, \{e\} \subseteq \bigcup X$ we can find $C, C' \in X$ such that $a \subseteq C$ and $e \in C'$. Moreover, being X pairwise compatible, there is $C'' \in X$, upper bound of C and C', containing both a and e. Therefore $\vdash\!\circ_{C''}(a, e, A' \cap C'')$, and thus by definition of choice $e \hookrightarrow_{C''} e'$ for $e' \in a$ or $e'' \hookrightarrow_{C''} e$ for $e'' \in A' \cap C''$. It follows that, as desired, $e \hookrightarrow_{\bigcup X} e'$ or $(e'' \in \bigcup X$ and) $e'' \hookrightarrow_{\bigcup X} e$.

   The relation $\hookrightarrow_{\bigcup X}$ is acyclic since point (1) implies that a cycle of $\hookrightarrow_{\bigcup X}$ in $\bigcup X$ should be entirely inside a single configuration $C \in X$. Furthermore it is easily seen that given an event $e \in \bigcup X$, $(\bigcup X)[\![e]\!] = C[\![e]\!]$, for any $C \in X$ such that $e \in C$. Therefore $(\bigcup X)[\![e]\!]$ is surely finite.

   Hence $\hookrightarrow_{\bigcup X}$ is a choice and thus $\bigsqcup X$ is a configuration. A routine verification, using point (1) allows one to conclude that $\bigcup X$ is the least upper bound of X. □

THEOREM 4.21 (CONFIGURATIONS FORM A DOMAIN)
Let I be an IES*. Then $\langle \mathit{Conf}(I), \sqsubseteq\rangle$ is a (finitary prime algebraic) domain. The complete primes of $\mathit{Conf}(I)$ are the possible histories of events in I, i.e.*

$$\mathit{Pr}(\mathit{Conf}(I)) = \{C[\![e]\!] \mid C \in \mathit{Conf}(I), e \in C\}.$$

PROOF. Let us start by showing that for each $C \in \mathit{Conf}(I)$ and $e \in C$, the configuration $C[\![e]\!]$ is a complete prime element. Suppose $C[\![e]\!] \sqsubseteq \bigsqcup X$ for $X \subseteq \mathit{Conf}(I)$ pairwise compatible. Therefore there exists $C_1 \in X$ such that $e \in C_1$. Since C_1 and $C[\![e]\!]$ are bounded by $\bigsqcup X$, by Lemma 4.20.(1), $C[\![e]\!] = C_1[\![e]\!]$. Observing that $C_1[\![e]\!] \sqsubseteq C_1$, it follows that, as desired, $C[\![e]\!] \sqsubseteq C_1$.

Now, by a set-theoretical calculation exploiting the definition of history (Definition 4.19) and the characterization of the least upper bound in Lemma 4.20, we obtain

$$C = \bigsqcup_{e\in C} C[\![e]\!] = \bigsqcup \mathit{Pr}(C).$$

This shows that $\mathit{Conf}(I)$ is prime algebraic and that $\mathit{Pr}(\mathit{Conf}(I)) = \{C[\![e]\!] \mid C \in \mathit{Conf}(I), e \in C\}$.

The fact that $\mathit{Conf}(I)$ is coherent has been proved in Lemma 4.20.(3). Finally, the finitariness of $\mathit{Conf}(I)$ follows from prime algebraicity and the fact that $C[\![e]\!]$ is finite for each $C \in \mathit{Conf}(I)$ and $e \in C$. □

We remark that if G is an AES and $I = \mathscr{J}_a(G)$ is its encoding into IES's, then for each configuration of I the choice relation is uniquely determined as the restriction of the asymmetric conflict to that configuration. Therefore the domain of configurations $\mathit{Conf}(I)$ defined in this section coincides with the domain $\mathit{Conf}(G)$ as defined in the previous chapter.

4.2.2 A coreflection between IES and Dom

To prove that the construction which associates to an IES its domain of configurations lifts to a functor from **IES** to **Dom** a basic result is the fact that IES-morphisms preserve configurations.

Observe that since configurations are not simply sets of events it is not completely obvious, a priori, what should be the image of a configuration through a morphism. Let $f : I_0 \to I_1$ be an IES-morphism and let C_0 be a configuration of I_0. According to the intuition underlying IES (and general event structures) morphisms, we expect that any possible execution of the events in C_0 can be simulated in $f(C_0)$. But the converse implication is not required to hold, namely the level of concurrency in $f(C_0)$ may be higher. For instance we can map two causally related events $e_0 \leq e_1$ to a pair of concurrent events. Hence we cannot pretend that the whole image of the choice relation of C_0 is a choice for $f(C_0)$, but just that there is a choice for $f(C_0)$ included in such image. By the properties of choices, there is only one choice on $f(C_0)$ included in the image of $\hookrightarrow_{C_0}$, which is obtained as the intersection of the image of $\hookrightarrow_{C_0}$ with $choices(f(C_0))$.

Given a function $f : X \to Y$ and a relation $r \subseteq X \times X$, we will denote by $f(r)$ the relation in Y defined as $f(r) = \{(y, y') \mid \exists (x, x') \in r.\ f(x) = y \ \wedge \ f(x') = y'\}$.

LEMMA 4.22 (MORPHISMS PRESERVE CONFIGURATIONS)
Let $f : I_0 \to I_1$ be an IES-morphism and let $\langle C_0, \hookrightarrow_0 \rangle \in Conf(I_0)$. Then the pair $\langle C_1, \hookrightarrow_1 \rangle$ with $C_1 = f(C_0)$ and $\hookrightarrow_1 = f(\hookrightarrow_0) \cap choices(f(C_0))$, namely the unique choice relation on C_1 included in $f(\hookrightarrow_{C_0})$, is a configuration in I_1.

Moreover the function $f^* : Conf(I_0) \to Conf(I_1)$ which associates to each configuration C_0 the configuration C_1 defined as above, is a domain morphism.

PROOF. To prove that $\hookrightarrow_1$ is a choice for $f(C_0)$ and thus $\langle f(C_0), \hookrightarrow_1 \rangle$ is a configuration, first observe that $\hookrightarrow_1 \subseteq choices(C_1)$ by definition.

Let us verify the validity of condition (1) in the definition of choice (Definition 4.12). Assume that $\vdash\!\!\circ_{f(C_0)}(a_1, f(e_0), A_1)$. This means that $\vdash\!\!\circ(a_1, f(e_0), A_1')$ with $a_1 \subseteq f(C_0)$ and $A_1 = A_1' \cap f(C_0)$. We distinguish two cases according to the shape of a_1:

- If $a_1 = \emptyset$, and thus $A_1' < f(e_0)$, by condition (2) in the definition of IES-morphism it follows that there exists $A_0 \subseteq f^{-1}(A_1')$ such that $A_0 < e_0$. Since $e_0 \in C_0$, by Proposition 4.13, $A_0 \cap C_0$ is non-empty (precisely, it is a singleton). Take $e_0'' \in A_0 \cap C_0$. By rule ($\nearrow$ 2), $e_0'' \nearrow e_0$ and thus, by Proposition 4.14, we have $e_0'' \hookrightarrow_0 e_0$. Hence, by construction, $f(e_0'') \hookrightarrow_1 f(e_0)$. Notice that $f(e_0'') \in A_1' \cap f(C_0) = A_1$.

- If $a_1 = \{f(e_0')\}$, then by condition (3) in the definition of IES-morphism we can find $a_0 \subseteq \{e_0'\}$ and $A_0 \subseteq f^{-1}(A_1')$ such that $\vdash\!\!\circ(a_0, e_0, A_0)$.

  If $a_0 = \emptyset$ we proceed as in the previous case. If instead $a_0 = \{e_0'\}$ then, by definition of choice $e_0 \hookrightarrow_0 e_0'$ or $e_0'' \hookrightarrow_0 e_0$ for $e_0'' \in A_0$. Therefore $f(e_0) \hookrightarrow_1 f(e_0')$ or $f(e_0'') \hookrightarrow_1 f(e_0)$ (and observe that $f(e_0'') \in A_1$).

As for condition (2), to show that $\hookrightarrow_1$ is acyclic, first observe that a IES-morphism is injective on a configuration. In fact, if $e_0, e_0' \in C_0$ and $f(e_0) = f(e_0')$ then $e_0 = e_0'$ or $e_0 \# e_0'$. But, by Proposition 4.15, the second possibility cannot arise. Now, if there were a cycle of $\hookrightarrow_1$ then, by the above observation and by definition of $\hookrightarrow_1$, a cycle should have been already present in $\hookrightarrow_0$, contradicting the hypothesis that C_0 is a configuration.

Finally, observe that also condition (3) holds, since by an analogous reasoning, the finitariness of the choice in C_0 implies the finitariness of the choice in $f(C_0)$.

Let us show that $f^* : \mathit{Conf}(I_0) \to \mathit{Conf}(I_1)$ is a morphism in **Dom**.

- *If C and C' are compatible then $f^*(C \sqcap C') = f^*(C) \sqcap f^*(C')$.*
  Recalling how the greatest lower bound of configurations is computed (see Lemma 4.20.(3)), we have that

  $$f^*(C \sqcap C') = \langle f(C \cap C'), f(\hookrightarrow_C \cap \hookrightarrow_{C'}) \cap \mathit{choices}(f(C \cap C'))\rangle,$$

  while

  $$\begin{aligned} & f^*(C) \sqcap f^*(C') = \\ & \quad = \langle f(C), f(\hookrightarrow_C) \cap \mathit{choices}(f(C))\rangle \sqcap \langle f(C'), f(\hookrightarrow_{C'}) \cap \mathit{choices}(f(C'))\rangle \\ & \quad = \langle f(C) \cap f(C'), f(\hookrightarrow_C) \cap f(\hookrightarrow_{C'}) \cap \mathit{choices}(f(C)) \cap \mathit{choices}(f(C'))\rangle \end{aligned}$$

  Observe that f is injective on $C \cup C'$ since C and C' have an upper bound C'', and, as already observed, f is injective on configurations. By using this fact, we can deduce that $f(C) \cap f(C') = f(C \cap C')$, $f(\hookrightarrow_C) \cap f(\hookrightarrow_{C'}) = f(\hookrightarrow_C \cap \hookrightarrow_{C'})$. Moreover it is easy to see that $\mathit{choices}(C \cap C') = \mathit{choices}(C) \cap \mathit{choices}(C')$ holds in general. Therefore we conclude that $f^*(C \sqcap C') = f^*(C) \sqcap f^*(C')$.

- *$f^*(\bigsqcup X) = \bigsqcup f^*(X)$, for $X \subseteq \mathit{Conf}(I_0)$ pairwise compatible.*
  Keeping in mind the characterization of the least upper bound given in Lemma 4.20.(4), we obtain

  $$\begin{aligned} & \bigsqcup f^*(X) = \\ & \quad = \langle \bigcup\{f(C) \mid C \in X\}, \bigcup\{f(\hookrightarrow_C) \cap \mathit{choices}(f(C)) \mid C \in X\}\rangle \\ & \quad = \langle f(\bigcup X), f(\bigcup\{\hookrightarrow_C \mid C \in X\}) \cap \mathit{choices}(f(\bigcup X))\rangle \\ & \quad = f^*(\langle \bigcup X, \bigcup\{\hookrightarrow_C \mid C \in X\}\rangle) \\ & \quad = f^*(\bigsqcup X) \end{aligned}$$

  To understand the second passage observe that

  $$\begin{aligned} & \bigcup\{f(\hookrightarrow_C) \cap \mathit{choices}(f(C)) \mid C \in X\} \subseteq \qquad \text{[by set-theoretical properties]} \\ & \quad \subseteq \bigcup\{f(\hookrightarrow_C) \mid C \in X\} \cap \bigcup\{\mathit{choices}(f(C)) \mid C \in X\} \qquad \text{[by definition of } \mathit{choices}\text{]} \\ & \quad \subseteq f(\bigcup\{\hookrightarrow_C \mid C \in X\}) \cap \mathit{choices}(f(\bigcup X)) \end{aligned}$$

  Therefore Proposition 4.17.(2) and the equality $\bigcup\{f(C) \mid C \in X\} = f(\bigcup X)$ allow us to conclude.

- *$C \prec C'$ implies $f^*(C) \preceq f^*(C')$.*
  This property immediately follows form the observation that, as in the case of AES's, $C \prec C'$ iff $C \sqsubseteq C'$ and $|C' - C| = 1$.

□

The previous lemma implies that the construction taking an IES into its domain of configurations can be viewed as a functor.

PROPOSITION 4.23
There exists a functor $\mathscr{L}_i : \mathbf{IES} \to \mathbf{Dom}$ *defined as:*

- $\mathscr{L}_i(I) = Conf(I)$, *for each* IES I;
- $\mathscr{L}_i(f) = f^*$, *for each* IES*-morphism* $f : I_0 \to I_1$.

A functor going back from domains to IES's, namely $\mathscr{P}_i : \mathbf{Dom} \to \mathbf{IES}$ can be simply obtained as the composition of the functor $\mathscr{P} : \mathbf{Dom} \to \mathbf{PES}$, defined by Winskel (see Section 2.4), with the full embedding $\mathscr{J}_a \circ \mathscr{J}$ of **PES** into **IES** discussed after Proposition 4.10, hence $\mathscr{P}_i = \mathscr{J}_a \circ \mathscr{J} \circ \mathscr{P}$.

We finally prove that $\mathscr{P}_i$ is left adjoint to $\mathscr{L}_i$ and thus that they establish a coreflection between **IES** and **Dom**. As for AES's, given an IES I, the component at I of the counit of the adjunction is the function $\epsilon_I : \mathscr{P}_i \circ \mathscr{L}_i(I) \to I$, mapping each history of an event e into the event e itself. The next preliminary lemma proves that such a mapping is indeed a IES-morphism.

LEMMA 4.24
The function $\epsilon_I : \mathscr{P}_i(\mathscr{L}_i(I)) \to I$ *defined as* $\epsilon_I(C[\![e]\!]) = e$, *for all* $C \in Conf(I)$ *and* $e \in C$, *is an* IES*-morphism.*

PROOF. Let us prove that ϵ_I satisfies conditions (1)-(3) of Definition 4.6.

1. $\epsilon_I(C[\![e]\!]) = \epsilon_I(C'[\![e']\!]) \ \wedge \ C[\![e]\!] \neq C'[\![e']\!] \quad \Rightarrow \quad C[\![e]\!] \# C'[\![e']\!]$.
   Assume that $\epsilon_I(C[\![e]\!]) = \epsilon_I(C'[\![e']\!])$, namely $e = e'$, and $C[\![e]\!] \neq C'[\![e']\!]$. By Lemma 4.20.(2) it follows that there is no upper bound for $\{C, C'\}$. In fact, if there were an upper bound C'' then necessarily $C[\![e]\!] = C''[\![e]\!] = C'[\![e]\!]$. Hence $e \# e'$.

2. $A_1 < \epsilon_I(C[\![e]\!]) \quad \Rightarrow \quad \exists A_0 \subseteq \epsilon_I^{-1}(A_1).\ A_0 < C[\![e]\!]$.
   Let us assume $A_1 < \epsilon_I(C[\![e]\!]) = e$. Since $e \in C$, by Proposition 4.13, $A_1 \cap C = \{e'\}$ for some e'. Moreover, since $e' \in A_1 < e$, by rule $(\nearrow 2)$, $e' \nearrow e$ and thus, by Proposition 4.14 and the definition of history, $e' \in C[\![e]\!]$.

   By point (2) of Lemma 4.20, one easily derives that $C[\![e']\!] \sqsubseteq C[\![e]\!]$. Therefore, according to the definition of $\mathscr{P}_i$, $C[\![e']\!] < C[\![e]\!]$ and since $e' \in A_1$, $\{C[\![e']\!]\} \subseteq \epsilon_I^{-1}(A_1)$.

3. $\vdash\!\circ(\{\epsilon_I(C'[\![e']\!])\}, \epsilon_I(C[\![e]\!]), A_1) \quad \Rightarrow \quad \exists A_0 \subseteq \epsilon_I^{-1}(A_1).\ \exists a_0 \subseteq \{C'[\![e']\!]\}.\ \vdash\!\circ(a_0, C[\![e]\!], A_0)$.
   Assume $\vdash\!\circ(\{\epsilon_I(C'[\![e']\!])\}, \epsilon_I(C[\![e]\!]), A_1)$, namely

   $$\vdash\!\circ(\{e'\}, e, A_1).$$

   If $\neg(C[\![e]\!] \uparrow C'[\![e']\!])$ then, by definition of $\mathscr{P}_i$, $C[\![e]\!] \# C'[\![e']\!]$ and thus $C[\![e]\!] \nearrow C'[\![e']\!]$. Hence $\vdash\!\circ(\{C'[\![e']\!]\}, C[\![e]\!], \emptyset)$, which clearly satisfies the desired condition.

   Suppose, instead, that $C[\![e]\!] \uparrow C'[\![e']\!]$. We distinguish two subcases:

   - If $e' \in C[\![e]\!]$ then $A_1 \cap C[\![e]\!] \neq \emptyset$. Indeed, being $C[\![e]\!]$ a configuration, $A_1 \cap C[\![e]\!]$ must be a singleton $\{e''\}$. As above, by Lemma 4.20.(2), $C[\![e'']\!] \sqsubseteq C[\![e]\!]$ and thus, by definition of $\mathscr{P}_i$, $C[\![e'']\!] < C[\![e]\!]$. Therefore $\vdash\!\circ(\emptyset, C[\![e]\!], \{C[\![e'']\!]\})$, which allows us to conclude, since $e'' \in A_1$ implies $\{C[\![e'']\!]\} \subseteq \epsilon_I^{-1}(A_1)$.

- Assume $e' \notin C[\![e]\!]$. Consider a configuration C'', upper bound of $C[\![e]\!]$ and $C'[\![e']\!]$, which exists by assumption. Since $e, e' \in C''$ it must be $e \hookrightarrow_{C''} e'$. In fact, otherwise there would be $e'' \in C'' \cap A_1$ and $e'' \hookrightarrow_{C''} e$. But then, by Lemma 4.20.(1), $e'' \in C[\![e]\!]$, an thus, being $e' < e''$, we would have $e' \in C[\![e]\!]$, contradicting the hypothesis.

  Therefore, by Lemma 4.20.(1), $e \in C'[\![e']\!]$, and thus $C[\![e]\!] \sqsubseteq C'[\![e']\!]$, implying $C[\![e]\!] < C'[\![e']\!]$. Hence $C[\![e]\!] \nearrow C'[\![e']\!]$, and therefore $\mapsto(\{C'[\![e']\!]\}, C[\![e]\!], \emptyset)$. □

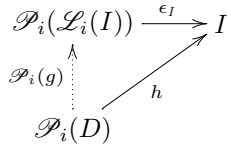
The main theorem relies on the following technical result. Its proof is a straightforward variation of the proof of the corresponding result for AES's (Lemma 3.24).

LEMMA 4.25
Let I be an IES*, D a domain and let $g : D \to \mathscr{L}_i(I)$ be a domain morphism. Then for all $p \in Pr(D)$, $\mid g(p) - \bigcup g(Pr(p) - \{p\}) \mid \leq 1$ and*

$$\mathscr{P}_i(g)(p) = \begin{cases} \bot & \text{if } g(p) - \bigcup g(Pr(p) - \{p\}) = \emptyset \\ g(p)[\![e]\!] & \text{if } g(p) - \bigcup g(Pr(p) - \{p\}) = \{e\} \end{cases}$$

THEOREM 4.26 (COREFLECTION BETWEEN **IES** AND **Dom**)
$\mathscr{P}_i \dashv \mathscr{L}_i$.

PROOF. Let I be an IES and let $\epsilon_I : \mathscr{P}_i(\mathscr{L}_i(I)) \to I$ be the morphism defined as in Lemma 4.24. We have to show that given any domain $(D, \sqsubseteq)$ and IES-morphism $h : \mathscr{P}_i(D) \to I$, there is a unique domain morphism $g : D \to \mathscr{L}_i(I)$ such that the following diagram commutes:



The proof follows the same lines of that for AES's, and thus some parts are only sketched.

Existence
The morphism $g : D \to \mathscr{L}_i(I)$ can be defined as follows. Given $d \in D$, observe that $C_d = \langle Pr(d), \sqsubset_{Pr(d)} \rangle$ is a configuration of I, where $\sqsubset_{Pr(d)} = \sqsubset \cap (Pr(d) \times Pr(d))$. Therefore we can define

$$g(d) = h^*(C_d).$$

The fact that $h^*(C_d)$ is a configuration in $\mathscr{P}_i(D)$ and thus an element of $\mathscr{L}_i(I)$, follows from Lemma 4.22.

Moreover g is a domain morphism. In fact it is

- $\preceq$*-preserving.* By prime algebraicity, $d, d' \in D$, with $d \prec d'$ then $Pr(d') - Pr(d) = \{p\}$, for some $p \in Pr(D)$. Thus

  $$\begin{aligned} &g(d') - g(d) = \\ &\quad = h^*(Pr(d')) - h^*(Pr(d)) \\ &\quad \subseteq \{h(p)\} \end{aligned}$$

  Therefore $|g(d') - g(d)| \leq 1$ and, since it is easy to see that $g(d) \sqsubseteq g(d')$, we conclude $g(d) \preceq g(d')$.

- *Additive.* Let $X \subseteq D$ be a pairwise compatible set. Then

$$g(\bigsqcup X) = h^*(\langle C_X, \hookrightarrow_{C_X}\rangle) = \langle h(C_X), h(\hookrightarrow_{C_X}) \cap \mathit{choices}(h(C_X))\rangle$$

where $C_X = Pr(\bigsqcup X) = \bigcup_{x\in X} Pr(x)$ and $\hookrightarrow_{C_X} = \sqsubset_{C_X}$. On the other hand

$$\begin{aligned}
&\textstyle\bigsqcup_{x\in X} g(x) = \\
&\quad = \textstyle\bigsqcup_{x\in X} h^*(\langle Pr(x), \sqsubset_{Pr(x)}\rangle)) \\
&\quad = \langle \textstyle\bigcup_{x\in X} h(Pr(x)), \bigcup_{x\in X}(h(\sqsubset_{Pr(x)}) \cap \mathit{choices}(h(Pr(x))))\rangle \\
&\quad = \langle h(C_X), \textstyle\bigcup_{x\in X}(h(\sqsubset_{Pr(x)}) \cap \mathit{choices}(h(Pr(x))))\rangle
\end{aligned}$$

Now, the choice relation of the configuration above is included in the choice of the configuration $g(\bigsqcup X)$, namely

$$\textstyle\bigcup_{x\in X}(h(\sqsubset_{Pr(x)}) \cap \mathit{choices}(h(Pr(x)))) \subseteq h(\hookrightarrow_{C_X}) \cap \mathit{choices}(C_X)$$

Thus by using Proposition 4.17.(2) we can conclude that $g(\bigsqcup X) = \bigsqcup_{x\in X} g(x)$.

- *Stable.* Let $d, d' \in D$ with $d \uparrow d'$, then:

$$g(d \sqcap d') = h^*(\langle C, \hookrightarrow_C\rangle) = \langle h(C), h(\hookrightarrow_C) \cap \mathit{choices}(h(C))\rangle,$$

where $C = Pr(d \sqcap d') = Pr(d) \cap Pr(d')$ and $\hookrightarrow_C = \sqsubset_C$. Moreover

$$\begin{aligned}
&g(d) \sqcap g(d') = \\
&\quad = \langle h(Pr(d)), h(\sqsubset_{Pr(d)}) \cap \mathit{choices}(h(Pr(d)))\rangle \\
&\qquad \sqcap \langle h(Pr(d')), h(\sqsubset_{Pr(d')}) \cap \mathit{choices}(h(Pr(d')))\rangle
\end{aligned}$$

Now, since $d \uparrow d'$ it is easy to see that h is injective on $Pr(d) \cup Pr(d')$ and therefore the set of events of $g(d) \sqcap g(d')$ is

$$h(Pr(d)) \cap h(Pr(d')) = h(Pr(d) \cap Pr(d')) = h(C),$$

namely it coincides with the set of events of $g(d \sqcap d')$.

By a similar argument, $h(\sqsubset_{Pr(d)}) \cap h(\sqsubset_{Pr(d')}) = h(\sqsubset_{Pr(d)\cap Pr(d')}) = h(\sqsubset_C)$. Moreover, reasoning as in the proof of Lemma 4.22, we have,

$$\begin{aligned}
&\mathit{choices}(h(Pr(d))) \cap \mathit{choices}(h(Pr(d'))) \\
&\quad = \mathit{choices}(h(Pr(d)) \cap h(Pr(d'))) \quad [\text{since } \mathit{choices}(X \cap Y) = \mathit{choices}(X) \cap \mathit{choices}(Y)] \\
&\quad = \mathit{choices}(h(Pr(d) \cap Pr(d'))) \quad [\text{by injectivity of } h \text{ on } C] \\
&\quad = \mathit{choices}(h(C))
\end{aligned}$$

and we are able to conclude that also the choice relation in $g(d) \sqcap g(d')$ is the same as in $g(d \sqcap d')$. In fact

$$\begin{aligned}
&h(\sqsubset_{Pr(d)}) \cap h(\sqsubset_{Pr(d')}) \cap \mathit{choices}(h(Pr(d))) \cap \mathit{choices}(h(Pr(d'))) \\
&\quad = h(\sqsubset_C) \cap \mathit{choices}(h(Pr(d) \cap Pr(d'))) \quad [\text{by injectivity of } h \text{ on } C \text{ and remark above}] \\
&\quad = h(\hookrightarrow_C) \cap \mathit{choices}(h(C))
\end{aligned}$$

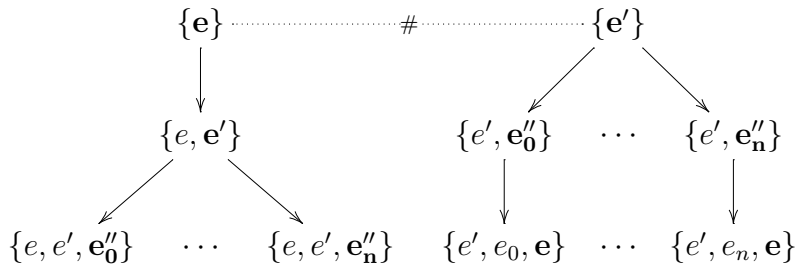


Figure 4.2: The PES corresponding to the IES where $\mapsto (\{e'\}, e, \{e_0, \ldots, e_n\})$.

The fact that the morphism g defined as above makes the diagram commute and its uniqueness can be proved, as for AES's, by exploiting essentially Lemma 4.25. □

It is worth stressing that the above result, together with Winskel's equivalence between the category **Dom** of domains and the category **PES** of prime event structures, allow to translate an IES I into a PES $\mathscr{P}(\mathscr{L}_i(I))$. The universal characterization of the construction intuitively ensures that the PES obtained in this way is the "best approximation" of I in the category **PES**. By the characterization of the complete prime elements in the domain of configurations (see Theorem 4.21) we have that the events in $\mathscr{P}(\mathscr{L}_i(I))$ are the possible histories of the events in I. Figure 4.2 shows the PES corresponding to a basic IES containing the events $\{e, e', e_0, \ldots, e_n\}$ related by the DE-relation as $\mapsto(\{e'\}, e, \{e_0, \ldots, e_n\})$. We explicitly represent history of an event e as a set of events, where e appears in boldface style.

4.2.3 Removing non-executable events

We already observed that the non-executability of events in IES's cannot be completely captured in a syntactic way, in the sense that there are no proof systems singling out exactly the non-executable events. However, we can adopt a semantic approach to rule out unused events from an IES, namely we can simply remove from a given IES all events which do not appear in any configuration. Nicely, this can be done functorially and the subcategory $\mathbf{IES}^*$ of IES's where all events are executable turns out to be a coreflective subcategory of **IES**. Moreover, the coreflection between **IES** and **Dom** can be easily shown to restrict to a coreflection between $\mathbf{IES}^*$ and **Dom**. This subsection is not essential to the remainder of the chapter, since it will be used only for discussing a negative result. Anyway, we think it answers a very natural question on IES's.

We start defining the subcategory of IES's where all events are executable.

DEFINITION 4.27
We denote by $\mathbf{IES}^*$ *the full subcategory of* **IES** *consisting of the* IES*'s* $I = \langle E, \mapsto \rangle$ *such that for any* $e \in E$ *there exists* $C \in Conf(I)$ *with* $e \in C$.

Any IES can be turned into an object of $\mathbf{IES}^*$ by forgetting the events which do not appear in any configuration. The next definition introduces the functor $\Psi : \mathbf{IES} \to \mathbf{IES}^*$ performing such construction.

DEFINITION 4.28
We denote by $\Psi : \mathbf{IES} \to \mathbf{IES}^*$ *the functor mapping each* IES I *into the* $\mathbf{IES}^*$ *object* $\Psi(I) = \overline{\langle \psi(E), \vdash_{\psi(E)} \rangle}$, *where* $\psi(E)$ *is the set of executable events in* I, *namely*

$$\psi(E) = \{e \in E \mid \exists C \in \mathit{Conf}(I).\ e \in C\}.$$

Moreover if $f : I_0 \to I_1$ *is an* IES*-morphism then* $\Psi(f) = f_{|\psi(E_0)}$. *With* $\mathscr{J}_{ies} : \mathbf{IES}^* \to \mathbf{IES}$ *we denote the inclusion.*

The fact that $\Psi(I)$ is a $\mathbf{IES}^*$ object follows easily from its definition. The well-definedness of $\Psi(f)$ for any IES-morphism f is basically a consequence of the fact that, by Lemma 4.22, an IES-morphism preserves configurations and thus also executable events.

Before formalizing the above claim, we make explicit two easy properties of the relations of causality and asymmetric conflict in $\Psi(I)$.

FACT 4.29
Let I be an IES. Then, for any $e, e' \in \psi(E)$ and $A \subseteq E$

1. $e \nearrow_I e' \quad \Rightarrow \quad e \nearrow_{\Psi(I)} e'$;
2. $A <_I e \quad \Rightarrow \quad (A \cap \psi(E)) <_{\Psi(I)} e$
3. $\#_I A \ \wedge \ A \subseteq \psi(E) \quad \Rightarrow \quad \#_{\Psi(I)} A$.

PROPOSITION 4.30
Let I_0 *and* I_1 *be* IES*'s and let* $f : I_0 \to I_1$ *be an* IES*-morphism. Then* $\Psi(f) : \Psi(I_0) \to \Psi(I_1)$, *defined as above, is an* IES*-morphism.*

PROOF. First observe that

$$f(\psi(E_0)) \subseteq \psi(E_1) \qquad (\dagger)$$

and thus the restriction $f_{|\psi(E_0)} : \psi(E_0) \to \psi(E_1)$ is a well-defined function. In fact, if $e_0 \in \psi(E_0)$ then $e_0 \in C_0$ for some configuration $C_0 \in \mathit{Conf}(I_0)$. Hence, if defined, $f(e_0) \in f(C_0)$ and, by Lemma 4.22, $f^*(C_0)$ is a configuration of I_1. Thus $f(e_0) \in \psi(E_1)$.

Now, for $i \in \{0, 1\}$, let us denote by $\vdash_i$, $<_i$, $\nearrow_i$ and $\#_i$ the relations in I_i, and by $\vdash_{\psi_i}$, $<_{\psi_i}$, $\nearrow_{\psi_i}$ and $\#_{\psi_i}$ the relations in $\langle \psi(E_i), \vdash_{\psi(E_i)} \rangle$, the pre-IES which, when saturated, gives the IES $\Psi(I_i)$. To show that $\Psi(f) : \Psi(I_0) \to \Psi(I_1)$ is an IES-morphism we verify that $\Psi(f) : \langle \psi(E_0), \vdash_{\psi(E_0)} \rangle \to \langle \psi(E_1), \vdash_{\psi(E_1)} \rangle$ satisfies conditions (1)-(4) of Lemma 4.9, namely

1. $\Psi(f)(e_0) = \Psi(f)(e_0') \ \wedge \ e_0 \neq e_0' \quad \Rightarrow \quad e_0 \#_{\psi_0} e_0'$;
2. $\vdash_{\psi_1}(\emptyset, \Psi(f)(e_0), A_1) \quad \Rightarrow \quad \exists A_0 \subseteq \Psi(f)^{-1}(A_1).\ A_0 <_{\psi_0} e_0$;

3. $\vdash\!\circ_{\psi_1}(\{\Psi(f)(e_0')\}, \Psi(f)(e_0), \emptyset) \quad \Rightarrow \quad e_0 \nearrow_{\psi_0} e_0'$;

4. $\vdash\!\circ_{\psi_1}(\{\Psi(f)(e_0')\}, \Psi(f)(e_0), A_1) \ \wedge \ A_1 \neq \emptyset \quad \Rightarrow$
$\exists A_0 \subseteq \Psi(f)^{-1}(A_1).\ \exists a_0 \subseteq \{e_0'\}.\ \vdash\!\circ_{\psi_0}(a_0, e_0, A_0)$.

To lighten the notation let f' denote $\Psi(f)$, i.e., the restriction $f_{|\psi(E_0)}$.

1. If $f'(e_0) = f'(e_0')$ and $e_0 \neq e_0'$, since $f : I_0 \to I_1$ is an IES-morphism, it must be $e_0 \#_0 e_0'$. Hence, by Fact 4.29.(3), $e_0 \#_{\psi_0} e_0'$.

2. Assume that $\vdash\!\circ_{\psi_1}(\emptyset, f'(e_0), A_1)$. By definition of $\Psi(I_1)$, recalling that $f'(e_0) = f(e_0)$, we have $\vdash\!\circ_1(\emptyset, f(e_0), A_1')$, with $A_1 = A_1' \cap \psi(E_1)$. Since, by definition of IES, $\#_p A_1'$, we can apply rule (< 1), thus obtaining

$$\frac{\vdash\!\circ_1(\emptyset, f(e_0), A_1') \quad \#_p A_1'}{A_1' <_1 f(e_0)} \ (< 1)$$

   By definition of morphism, there exists $A_0' \subseteq f^{-1}(A_1')$ such that $A_0' <_0 e_0$. If we define $A_0 = A_0' \cap \psi(E_0)$ then, by Fact 4.29.(1), $A_0 <_{\psi_0} e_0$ and, by the property (†) above, $A_0 \subseteq f'^{-1}(A_1)$.

3. Assume that $\vdash\!\circ_{\psi_1}(\{f'(e_0')\}, f'(e_0), \emptyset)$. By definition of $\vdash\!\circ_{\psi_1}$ and recalling that f' is the restriction of f, it must be $\vdash\!\circ_1(\{f(e_0')\}, f(e_0), A_1)$ with $A_1 \cap \psi(E_1) = \emptyset$. Hence, by definition of morphism, there exist $a_0 \subseteq \{e_0'\}$ and $A_0 \subseteq f^{-1}(A_1)$ such that $\vdash\!\circ_0(a_0, e_0, A_0)$. Since $A_1 \cap \psi(E_1) = \emptyset$, we deduce that $A_0 \cap \psi(E_0) = \emptyset$. Moreover, recalling that $e_0 \in \psi(E_0)$, namely it is executable, necessarily $a_0 = \{e_0'\}$. Therefore $\vdash\!\circ_{\psi_0}(\{e_0'\}, e_0, \emptyset)$, and thus $e_0 \nearrow_{\psi_0} e_0'$.

4. Assume that $\vdash\!\circ_{\psi_1}(\{f'(e_0')\}, f'(e_0), A_1)$ with $A_1 \neq \emptyset$. Then, by definition of $\vdash\!\circ_{\psi_1}$, we must have

$$\vdash\!\circ_1(\{f(e_0')\}, f'(e_0), A_1')$$

   where $A_1 = A_1' \cap \psi(E_1)$. By definition of IES-morphism, there must exist $A_0' \subseteq f^{-1}(A_1')$ and $a_0 \subseteq \{e_0'\}$ such that $\vdash\!\circ_0(a_0, e_0, A_0')$.

   If we define $A_0 = A_0' \cap \psi(E_0)$, then by definition of $\vdash\!\circ_{\psi_0}$, we have $\vdash\!\circ_{\psi_0}(a_0, e_0, A_0)$ and, by the property (†) proved above, $A_0 \subseteq f'^{-1}(A_1)$. □

It is easy to verify that, if I is a **IES*** object and I' is an arbitrary IES, then any IES-morphism $f : I \to \Psi(I')$ is also a morphism $f : I \to I'$. This simple observation allows us to conclude immediately that the inclusion of **IES*** into **IES** is left adjoint of the functor Ψ and thus that **IES*** is a coreflective subcategory of **IES**.

PROPOSITION 4.31 (RELATING **IES** AND **IES***)
$\Psi \vdash \mathscr{J}_{ies}$

Finally observe that the functor $\mathscr{P}_i : \mathbf{Dom} \to \mathbf{IES}$ maps each domain into the encoding of a PES, which is clearly an object in **IES***. Therefore it is easy to prove that the coreflection between **IES** and **Dom** restricts to a coreflection between **IES*** and **Dom**.

COROLLARY 4.32 (COREFLECTION BETWEEN **IES*** AND **Dom**)
Let $\mathscr{P}_{ie} : \mathbf{Dom} \to \mathbf{IES}^*$ *and* $\mathscr{L}_{ie} : \mathbf{IES}^* \to \mathbf{Dom}$ *denote the restrictions of the functors* $\mathscr{P}_i$ *and* $\mathscr{L}_i$. *Then* $\mathscr{P}_{ie} \dashv \mathscr{L}_{ie}$.

4.3 The category of inhibitor nets

This section defines the category **IN** of inhibitor nets by introducing a suitable notion of morphism. Morphisms of i-nets are morphisms between the underlying c-nets, satisfying a further condition which takes into account the presence of the inhibitor arcs. In this way the category **CN** of contextual nets is a (full) subcategory of **IN**.

DEFINITION 4.33 (I-NET MORPHISM)
Let N_0, N_1 be i-nets. A i-net morphism $h : N_0 \to N_1$ *is a pair $h = \langle h_T, h_S \rangle$, where $h_T : T_0 \to T_1$ is a partial function and $h_S : S_0 \to S_1$ is a multirelation such that*

1. $\mu h_S(m_0) = m_1$;
2. *for each $A \in \mu T$,*
   (a) $\mu h_S({}^\bullet A) = {}^\bullet \mu h_T(A)$,
   (b) $\mu h_S(A^\bullet) = \mu h_T(A)^\bullet$,
   (c) $[\![\underline{\mu h_T(A)}]\!] \leq \mu h_S(\underline{A}) \leq \underline{\mu h_T(A)}$.
   (d) $[\![h_S]\!]^{-1}({}^\circledcirc \mu h_T(A)) \subseteq {}^\circledcirc A$.[1]

where $[\![h_S]\!]$ is the set relation underlying the multirelation h_S. We denote by **IN** *the category having i-nets as objects and i-net morphisms as arrows.*

Conditions (1), (2.a) - (2.c) are the defining conditions of c-net morphisms (see Definition 3.26). Finally, condition (2.d) regarding the inhibitor arcs can be understood if we think of morphisms as simulations. As preconditions and contexts must be preserved to ensure that the morphism maps computations of N_0 into computations of N_1, similarly, inhibitor conditions, which are negative conditions, must be reflected. In fact condition (2.d) can be expressed, in words, by saying that if the image of a place s_0 in N_0 inhibits the image of a transition t_0, then s_0 must inhibit t_0 (see also the remark below).

REMARK 4.34
Observe that condition (2.d) on inhibiting places can be rewritten as

$$s_1 \in [\![\mu h_S(s_0)]\!] \ \wedge \ I_1(h_T(t_0), s_1) \quad \Rightarrow \quad I_0(t_0, s_0),$$

which shows more explicitly that inhibitor arcs are reflected. In particular, if h_S is a *total function* then

$$I_1(h_T(t_0), h_S(s_0)) \quad \Rightarrow \quad I_0(t_0, s_0).$$

[1] We are grateful to Nadia Busi and Michele Pinna for pointing out the appropriateness of this condition, generalizing the one appeared in a preliminary version of the work.

Although, to the best of our knowledge, no other categories of nets with inhibitor arcs have been introduced in the literature, our i-net morphisms can be seen as a generalization of the process mappings of [Bus98, BP96, BP99]. We will come back on this point in Section 4.6, where a (deterministic) process of an i-net N will be defined as a special morphism from a (deterministic) occurrence i-net to the net N.

PROPOSITION 4.35 (COMPOSITION OF MORPHISMS)
The class of i-net morphisms is closed under composition.

PROOF. Let $h_0 : N_0 \to N_1$ and $h_1 : N_1 \to N_2$ be two i-net morphisms. Their composition $h_1 \circ h_0$ obviously satisfies conditions (1) and (2.a)-(2.c) of Definition 4.33, since these are exactly the defining conditions of c-net morphisms which are known to be closed under composition.

Finally, $h_1 \circ h_0$ satisfies also condition (2.d). In fact, for any multiset of transition A in N_0:

$$\begin{aligned}
&[\![h_{1S} \circ h_{0S}]\!]^{-1}({}^{\circledcirc}\mu(h_{1T} \circ h_{0T})(A)) = \\
&\quad = [\![h_{0S}]\!]^{-1}([\![h_{1S}]\!]^{-1}({}^{\circledcirc}\mu h_{1T}(\mu h_{0T}(A)))) \\
&\quad \subseteq [\![h_{0S}]\!]^{-1}({}^{\circledcirc}\mu h_{0T}(A)) && \text{[since } h_1 \text{ is a morphism]} \\
&\quad \subseteq {}^{\circledcirc}A && \text{[since } h_0 \text{ is a morphism]}
\end{aligned}$$

Let us now verify that, as in the case of contextual nets, i-net morphisms preserve the token game, and thus the reachability of markings.

THEOREM 4.36 (MORPHISMS PRESERVE THE TOKEN GAME)
Let N_0 and N_1 be i-nets, and let $h = \langle h_T, h_S \rangle : N_0 \to N_1$ be an i-net morphism. Then for each $M, M' \in \mu S$ and $A \in \mu T$

$$M\,[A\rangle\,M' \quad \Rightarrow \quad \mu h_S(M)\,[\mu h_T(A)\rangle\,\mu h_S(M').$$

Therefore i-net morphisms preserve reachable markings, i.e. if M_0 is a reachable marking in N_0 then $\mu h_S(M_0)$ is reachable in N_1.

PROOF. Suppose that $M\,[A\rangle\,M'$, and thus, in particular, $M\,[A\rangle$, namely ${}^{\bullet}A + [\![\underline{A}]\!] \leq M$ and $[\![M + A^{\bullet}]\!] \cap {}^{\circledcirc}A = \emptyset$.

First notice that $\mu h_T(A)$ is enabled by $\mu h_S(M)$. The proof of the fact that ${}^{\bullet}\mu h_T(A) + [\![\underline{\mu h_T(A)}]\!] \leq \mu h_S(M)$ is the same as for c-net. Hence let us consider the new condition for the enabling regarding the inhibiting places. Observe that

$$\begin{aligned}
&[\![\mu h_S(M) + \mu h_T(A)^{\bullet}]\!] \cap {}^{\circledcirc}\mu h_T(A) = \\
&\quad = [\![\mu h_S(M) + \mu h_S(A^{\bullet})]\!] \cap {}^{\circledcirc}\mu h_T(A) && \text{[by (2.b) in the definition of morphism]} \\
&\quad = [\![\mu h_S(M + A^{\bullet})]\!] \cap {}^{\circledcirc}\mu h_T(A) \\
&\quad = \emptyset
\end{aligned}$$

The last passage is justified by observing that if $s_1 \in [\![\mu h_S(M + A^{\bullet})]\!] \cap {}^{\circledcirc}\mu h_T(A)$, then there is $s_0 \in [\![M + A^{\bullet}]\!]$ such that $s_1 \in [\![\mu h_S(s_0)]\!]$ and $s_1 \in {}^{\circledcirc}h_T(A)$. By condition (2.d) in the definition of i-net morphism, this implies $s_0 \in {}^{\circledcirc}A$ and therefore $s_0 \in [\![M + A^{\bullet}]\!] \cap {}^{\circledcirc}A$, which instead is empty by hypothesis.

It is now immediate to conclude that $\mu h_S(M)\,[\mu h_T(A)\rangle\,\mu h_S(M')$. □

As in the case of contextual nets, the Winskel's style semantics will be defined on the subcategory of **IN** having semi-weighted nets as objects. The next definition introduces semi-weighted and safe inhibitor nets by generalizing in the obvious way the corresponding notions for contextual nets.

Definition 4.37 (semi-weighted and safe i-nets)
A semi-weighted *i-net is an i-net N such that the initial marking m is a set and F_{post} is a relation (i.e., $t^\bullet$ is a set for all $t \in T$). We denote by* **SW-IN** *the full subcategory of* **IN** *having semi-weighted i-nets as objects.*

A semi-weighted i-net is called safe *if also F_{pre} and C are relations (i.e., ${}^\bullet t$ and $\underline{t}$ are sets for all $t \in T$) and each reachable marking is a set.*

4.4 Occurrence i-nets and unfolding construction

Generally speaking, occurrence nets provide a static representation of the computations of general nets, in which the events (firing of transitions) and the relationships between events are made explicit. In the previous chapter the notion of occurrence net has been generalized from ordinary nets to nets with context conditions. Here, the presence of the inhibitor arcs and the complex kind of dependencies they induce on transitions makes hard to find an appropriate notion of occurrence i-net.

As a first step, consider an i-net where only forward conflicts are admitted, namely where each place is in the post-set of at most one transition. The condition of finiteness and acyclicity of the causes, which ensures that each transition of an occurrence (contextual) net is firable in some computation, has no natural generalizations in this setting. Indeed also the notion of set of causes of a transition becomes unclear. In fact, as already observed, a transition t to be fired may require the absence of tokens in a place s, and therefore it can be seen as a (weak) cause of the transition t' in the pre-set of s if it fires before s is filled, or as a consequence of any of the transitions $t_1, \ldots, t_n$ in the post-set of s, otherwise. One could think to have a set of causes depending on a kind of "choice", which specifies for any inhibitor arc (t, s) if transition t is executed before or after the place s is filled and in the second case which of the transitions in the pre-set ${}^\bullet s$ of the inhibitor place consumes the token. Then the firability of a transition t would amount to the existence of a choice which is acyclic on the transitions which must be executed before t. However, relying on this idea the notion of occurrence net would be not very easy to deal with and furthermore the unfolding construction would produce a net which is not decidable, in the sense the sets of transitions and places of the net are not recursive. In fact, due to the Turing completeness of inhibitor nets (see, e.g., [Age74]) the reachability of a marking, or equivalently the existence of a computation in which a given transition fires, is undecidable.

We propose the following solution. Given an i-net N, we first consider the underlying contextual net N_c, obtained from N by forgetting the inhibitor arcs. Then,

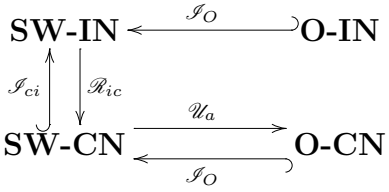


Figure 4.3: Functors relating semi-weighted (occurrence) c-nets and i-nets.

disregarding the inhibitor arcs, we apply to N_c the unfolding construction for c-nets defined in the previous chapter (Definition 3.46), which produces an occurrence c-net. Finally, if a place s and a transition t were originally related by an inhibitor arc in the net N, then we insert an inhibitor arc between each copy of s and each copy of t in $\mathscr{U}_a(N_c)$, thus obtaining the unfolding $\mathscr{U}_i(N)$ of the net N.

The characterization of the unfolding as a universal construction can be easily lifted from contextual to inhibitor nets. Furthermore, in this way the unfolding of an inhibitor net is decidable, a fact which, besides being nice from a theoretical point of view, may be helpful if one want to use the unfolding in practice to prove properties of the modelled system. The price to pay is that, differently from what happens for ordinary and contextual nets, some of the the transitions in the unfolding may not be not firable, since they are generated without taking care of inhibitor places. Therefore not all the transitions of the unfolding correspond to a concrete firing of a transition of the original net, but only those which are executable.

Since our unfolding construction disregards the inhibitor arcs, we define an occurrence i-net as an i-net which becomes an occurrence c-net when forgetting the inhibitor arcs. To formalize this fact it is useful to introduce two functors, the first one mapping each i-net to the underlying c-net, and the other one embedding the category of c-nets into the category of i-nets (see Figure 4.3).

DEFINITION 4.38
We denote by $\mathscr{R}_{ic}$: **SW-IN** $\rightarrow$ **SW-CN** *the functor which maps each i-net to the underlying c-net obtained by forgetting the inhibitor relation, namely* $\mathscr{R}_{ic}(\langle S,T,F,C,I,m\rangle) = \langle S,T,F,C,m\rangle$*, and by* $\mathscr{I}_{ci}$: **SW-CN** $\rightarrow$ **SW-IN** *the functor mapping each c-net into an i-net with an empty inhibitor relation, namely* $\mathscr{I}_{ci}(\langle S,T,F,C,m\rangle) = \langle S,T,F,C,\emptyset,m\rangle$*. Both functors act as the identity on morphisms.*

Recall that causal dependency on contextual nets is defined as the least transitive relation $<$ such that $t < t'$ if $t^\bullet \cap ({}^\bullet t' \cup \underline{t'}) \neq \emptyset$, the only novelty with respect to classical nets being the fact that a transition causally depends on transitions generating tokens in its context. Asymmetric conflict $\nearrow$ is defined by taking $t \nearrow t'$ if t' consumes a token in the context of t, namely if $\underline{t} \cap {}^\bullet t' \neq \emptyset$. Moreover $t \nearrow t'$ if $(t \neq t' \ \wedge \ {}^\bullet t \cap {}^\bullet t' \neq \emptyset)$ to capture the usual symmetric conflict, and finally,

Figure 4.4: Not all events of an occurrence i-net are executable.

according to the weak causality interpretation of $\nearrow$, $t \nearrow t'$ whenever $t < t'$. An *occurrence c-net* is then defined as a c-net N where causality is a finitary partial order, asymmetric conflict is acyclic on the causes of each transition, each place is in the post-set of at most one transition and the initial marking is the minimal set of places with respect to causality.

Definition 4.39 (occurrence i-nets)
An occurrence i-net *is a safe i-net N such that $\mathscr{R}_{ic}(N)$ is an occurrence c-net. The full subcategory of* **SW-IN** *having occurrence i-nets as objects is denoted by* **O-IN**.

We remark that, since the above definition does not take into account the inhibitor arcs of the net, we are not ensured that each transition in an occurrence i-net can be fired. For instance in the i-net N_2 of Figure 4.4, the only transition t can never fire, but, by looking at the underlying c-net $\mathscr{R}_{ic}(N_2)$ depicted aside, it is immediate to see that N_2 is an occurrence i-net.

In the previous chapter we have defined an unfolding functor $\mathscr{U}_a : \mathbf{SW\text{-}CN} \to \mathbf{O\text{-}CN}$, mapping each semi-weighted c-net to an occurrence c-net. The functor $\mathscr{U}_a$ has been shown to be right adjoint to the inclusion functor $\mathscr{I}_O : \mathbf{O\text{-}CN} \to \mathbf{SW\text{-}CN}$. By suitably using the functors $\mathscr{R}_{ic}$ and $\mathscr{I}_{ci}$ we can lift both the construction and the result to inhibitor nets.

Definition 4.40 (unfolding)
Let $N = \langle S, T, F, C, I, m\rangle$ be a semi-weighted i-net. Consider the occurrence c-net $\mathscr{U}_a(\mathscr{R}_{ic}(N)) = \langle S', T', F', C', m'\rangle$ and the folding morphism $f_N : \mathscr{U}_a(\mathscr{R}_{ic}(N)) \to \mathscr{R}_{ic}(N)$. Define an inhibiting relation on the net $\mathscr{U}_a(\mathscr{R}_{ic}(N))$ by taking for $s' \in S'$ and $t' \in T'$

$$I'(s', t') \quad \textit{iff} \quad I(f_N(s'), f_N(t')).$$

Then the unfolding $\mathscr{U}_i(N)$ of the net N is the occurrence i-net $\langle S', T', F', C', I', m'\rangle$ and the folding morphism *is given by f_N seen as a function from $\mathscr{U}_i(N)$ into N.*

The fact that $\mathscr{U}_i(N)$ is an occurrence i-net immediately follows from its construction. Furthermore, since the place component of f_N is a total function, according to what observed in Remark 4.34, it immediately follows that $\mathscr{U}_i(N)$ can be characterized as the *least* i-net which extends $\mathscr{U}_a(N)$ with the addition of inhibitor arcs in a way that $f_N : \mathscr{U}_i(N) \to N$ is a well defined i-net morphism.

The unfolding construction is functorial, namely we can define a functor $\mathscr{U}_i$: **SW-IN** $\rightarrow$ **O-IN**, which acts on arrows as $\mathscr{U}_a \circ \mathscr{R}_{ic}$. In other words, given h : $N_0 \rightarrow N_1$, the arrow $\mathscr{U}_i(h) : \mathscr{U}_i(N_0) \rightarrow \mathscr{U}_i(N_1)$ is obtained by interpreting h as a morphism between the c-nets underlying N_0 and N_1, taking its image via $\mathscr{U}_a$, and then considering the map $\mathscr{U}_a(h)$ as an arrow from $\mathscr{U}_i(N_0)$ to $\mathscr{U}_i(N_1)$.

PROPOSITION 4.41
*The unfolding construction extends to a functor $\mathscr{U}_i$: **SW-IN** $\rightarrow$ **O-IN**, which acts on arrows as $\mathscr{U}_a \circ \mathscr{R}_{ic}$.*

PROOF. The only thing to verify is that given an i-net morphism $h : N_0 \rightarrow N_1$, the c-net morphism $h' = \mathscr{U}_a \circ \mathscr{R}_{ic}(h) : \mathscr{U}_a(\mathscr{R}_{ic}(N_0)) \rightarrow \mathscr{U}_a(\mathscr{R}_{ic}(N_1))$, seen as a mapping $h' : \mathscr{U}_i(N_0) \rightarrow \mathscr{U}_i(N_1)$ is an i-net morphism.

First notice that the following diagram, where f_0 and f_1 are the folding morphisms, commutes by construction (although h', in principle, may not be an i-net morphism).

$$\begin{array}{ccc} N_0 & \xrightarrow{\quad h \quad} & N_1 \\ {\scriptstyle f_0}\uparrow & & \uparrow{\scriptstyle f_1} \\ \mathscr{U}_i(N_0) & \xrightarrow[h'=\mathscr{U}_a(h)]{} & \mathscr{U}_i(N_1) \end{array}$$

As usual, conditions (1) and (2.a)-(2.c) are automatically verified since h' is a c-net morphism. Let us prove the validity of condition (2.d), as expressed by Remark 4.34, namely

$$s'_1 \in [\![\mu h'_S(s'_0)]\!] \ \wedge \ I'_1(h'_T(t'_0), s'_1) \quad \Rightarrow \quad I'_0(t'_0, s'_0).$$

Assume $s'_1 \in [\![\mu h'_S(s'_0)]\!] \ \wedge \ I'_1(h'_T(t'_0), s'_1)$. Hence, $f_{1S}(s'_1) \in [\![\mu(f_{1S} \circ h'_S)(s'_0)]\!]$ and, by definition of the unfolding, $I_1(f_{1T}(h'_T(t'_0)), f_{1S}(s'_1))$. Therefore, by commutativity of the diagram

$$f_{1S}(s'_1) \in [\![\mu h_S(f_{0S}(s'_0))]\!] \quad \text{and} \quad I_1(h_T(f_{0T}(t'_0)), f_{1S}(s'_1))$$

Being h an i-net morphism, by condition (2.d) in Definition 4.33, we have that

$$I_0(f_{0T}(t'_0), f_{0S}(s'_0))$$

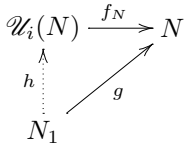
and therefore, by definition of the unfolding, $I'_0(t'_0, s'_0)$, which is the desired conclusion. □

We can now state the main result of this section, establishing a coreflection between semi-weighted i-nets and occurrence i-nets. It essentially relies on the fact that the unfolding for c-nets has been already characterized as an universal construction (Theorem 4.42).

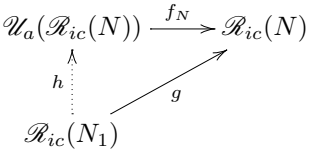
THEOREM 4.42 (COREFLECTION BETWEEN **SW-IN** AND **O-IN**)
*The unfolding functor $\mathscr{U}_i$: **SW-IN** $\rightarrow$ **O-IN** is right adjoint to the obvious inclusion functor $\mathscr{I}_O$: **O-IN** $\rightarrow$ **SW-IN** and thus establishes a coreflection between* **SW-IN** *and* **O-IN**.

The component at an object N in **SW-IN** *of the counit of the adjunction, f : $\mathscr{I}_O \circ \mathscr{U}_i \dot{\rightarrow} 1$, is the folding morphism $f_N : \mathscr{U}_i(N) \rightarrow N$.*

Proof. Let N be a semi-weighted i-net, let $\mathscr{U}_i(N) = \langle S', T', F', C', I', m' \rangle$ be its unfolding and let $f_N : \mathscr{U}_i(N) \to N$ be the folding morphism as in Definition 4.40. We have to show that for any occurrence i-net N_1 and for any morphism $g : N_1 \to N$ there exists a unique morphism $h : N_1 \to \mathscr{U}_i(N)$ such that the following diagram commutes:



The *existence* is readily proved by observing that an appropriate choice is $h = \mathscr{U}_i(g)$. The commutativity of the diagram simply follows by the commutativity of the diagram involving the underlying c-nets and c-net morphisms, namely



With a little abuse of notation, we have denoted with the same symbol the c-net morphism and the same mapping seen as an i-net morphism.

Also *uniqueness* follows easily by the universal property of the construction for c-nets. In fact let $h' : N_1 \to \mathscr{U}_i(N)$ be another i-net morphism such that $f_N \circ h' = g$. This means that h' is another c-net morphism which makes commute the diagram involving the underlying c-nets. This implies that, as desired, h and h' coincide. □

4.5 Inhibitor event structure semantics for i-nets

To give an event structure and a domain semantics for i-nets we investigate the relationship between occurrence i-nets and inhibitor event structures. The kind of dependencies arising among transitions in an occurrence i-net can be naturally represented by the DE-relation, and therefore the ies corresponding to an occurrence i-net is obtained, as in the case of c-nets, by forgetting the places and taking the transitions of the net as events. Furthermore morphisms between occurrence i-nets naturally restrict to morphisms between the corresponding ies's, and therefore the semantics can be expressed as a functor $\mathscr{E}_i$: **O-IN** → **IES**.

The converse step, from ies's to occurrence i-nets, instead, is shown to be very problematic. An object level construction can be easily performed, associating to each ies a corresponding i-net. However such a construction does not give rise to a functor. We discuss the origin of this problem, showing that it is intimately connected to or-causality, and we argue that it has no reasonable solutions in the considered framework.

4.5.1 From occurrence i-nets to IES's

Let us introduce DE-relation naturally associated to an occurrence i-net N, denoted with the symbol $\vdash\!\circ_N$. It will be used to define first a pre-IES and then an IES for the net N.

DEFINITION 4.43 (PRE-IES FOR OCCURRENCE I-NETS)
*Let N be an occurrence i-net. The pre-*IES *associated to N is defined as $I^p_N = \langle T, \vdash\!\circ^p_N \rangle$, with $\vdash\!\circ_N \subseteq \mathbf{2}^T_1 \times T \times \mathbf{2}^T$, given by: for $t, t' \in T$ and $s \in S$*

1. *if $t^\bullet \cap ({}^\bullet t' \cup \underline{t'}) \neq \emptyset$ then $\vdash\!\circ^p_N(\emptyset, t', \{t\})$*
2. *if $({}^\bullet t \cup \underline{t}) \cap {}^\bullet t' \neq \emptyset$ then $\vdash\!\circ^p_N(\{t'\}, t, \emptyset)$;*
3. *if $s \in {}^{\circledcirc}t$ then $\vdash\!\circ^p_N({}^\bullet s, t, s^\bullet)$.*

The first two clauses of the definition encode, by means of the DE-relation, the causal dependencies and the asymmetric conflicts induced by flow and read arcs. The last clause fully exploits the expressiveness of the DE-relation to represent the dependencies induced by inhibitor places.

Notice that I^p_N is a pre-IES, satisfying also condition (1) of the definition of IES. Therefore, as proved in Proposition 4.5, it can be "saturated" to obtain an IES.

DEFINITION 4.44 (IES FOR OCCURRENCE I-NETS)
The IES *associated to the occurrence i-net N, denoted by $I_N = \langle T, \vdash\!\circ_N \rangle$, is defined as $\overline{I^p_N}$.*

Recall that the causality, asymmetric conflict and conflict relations of I_N and I^p_N coincide. They will be denoted by $<_N$, $\nearrow_N$ and $\#_N$, respectively.

The next proposition shows that the above construction extends to a functor, by proving that the transition component of an i-net morphism is an IES-morphism between the corresponding IES's.

PROPOSITION 4.45
Let N_0 and N_1 be be occurrence i-nets and let $h : N_0 \to N_1$ be an i-net morphism. Then $h_T : I_{N_0} \to I_{N_1}$ is a IES*-morphism.*

PROOF. For $i \in \{0, 1\}$, let $<_i$, $\nearrow_i$ and $\#_i$ be the relations of causality, asymmetric conflict and conflict in the pre-IES $I^p_{N_i} = \langle E_i, \vdash\!\circ^p \rangle$. We show that $h_T : I^p_0 \to I^p_1$ satisfies conditions (1)-(4) in the hypotheses of Lemma 4.9 and thus h_T is an IES-morphism between the corresponding "saturated" IES's.

1. $h_T(t_0) = h_T(t'_0) \ \wedge \ t_0 \neq t'_0 \quad \Rightarrow \quad t_0 \#_0 t'_0$.
   This property can be proved exactly as for c-nets. Alternatively, we can observe that if $\#_{a0}$ denotes the conflict relation in the c-net $\mathscr{R}_{ic}(N_0)$ then it is easy to see that $\#_{a0} \subseteq \#_0$. Since $\mathscr{R}_{ic}(h) = h : \mathscr{R}_{ic}(N_0) \to \mathscr{R}_{ic}(N_1)$ is an c-net morphism, by resorting to Lemma 3.42 in the previous chapter we conclude that $t_0 \#_{a0} t'_0$ and therefore $t_0 \#_0 t'_0$.

2. $\vdash^p_1(\emptyset, h_T(t_0), A_1) \quad \Rightarrow \quad \exists A_0 \subseteq h_T^{-1}(A_1).\ A_0 <_0 t_0$.
Let us assume $\vdash^p_1(\emptyset, h_T(t_0), A_1)$. By the definition of $\vdash^p_1$ we can have

(a) $A_1 = \{t_1\}$ and $t_1{}^\bullet \cap {}^\bullet h_T(t_0) \neq \emptyset$.
Consider $s_1 \in t_1{}^\bullet \cap {}^\bullet h_T(t_0)$. By definition of i-net morphism there must exist $s_0 \in {}^\bullet t_0$ such that $h_S(s_0, s_1)$, and $t'_0 \in T_0$ such that $h_T(t'_0) = t_1$ and $s_0 \in t'_0{}^\bullet$. By definition of $\vdash^p_0$, if we define $A_0 = \{t'_0\}$, it follows that $\vdash^p_0(\emptyset, t_0, A_0)$, and thus by rule (< 1), $A_0 < t_0$. Recalling that $t'_0 \in h_T^{-1}(t_1)$ and thus $A_0 \subseteq h_T^{-1}(A_1)$ we conclude.

(b) $A_1 = \{t_1\}$ and $t_1{}^\bullet \cap \underline{h_T(t'_0)} \neq \emptyset$.
Analogous to case (a).

(c) $\exists s_1 \in {}^{\circledcirc}h_T(t_0).\ {}^\bullet s_1 = \emptyset \ \wedge \ s_1{}^\bullet = A_1$.
Since ${}^\bullet s_1 = \emptyset$, namely s_1 is in the initial marking m_1 of N_1, by definition of i-net morphism, there exists a unique $s_0 \in m_0$ such that $h_S(s_0, s_1)$. Again, by definition of i-net morphism, from $s_1 \in {}^{\circledcirc}h_T(t_0)$ and $h_S(s_0, s_1)$ it follows that $s_0 \in {}^{\circledcirc}t_0$. Hence $\vdash^p_0({}^\bullet s_0, t_0, s_0{}^\bullet)$, namely, recalling that $s_0 \in m_0$,

$$\vdash^p_0(\emptyset, t_0, s_0{}^\bullet).$$

Therefore, by rule (< 1), we have $s_0{}^\bullet <_0 t_0$. Observe that, by the condition (2.a) in the definition of i-net morphisms, $h_T(s_0{}^\bullet) \subseteq s_1{}^\bullet$ and, since $h_S(s_0, s_1)$, necessarily h is defined on each $t'_0 \in s_0{}^\bullet$. Thus $s_0{}^\bullet \subseteq h_T^{-1}(s_1{}^\bullet)$ concluding the proof for this case.

3. $\vdash^p_1(\{h_T(t'_0)\}, h_T(t_0), \emptyset) \quad \Rightarrow \quad t_0 \nearrow_0 t'_0$.
By definition of $\vdash^p_1$, we can have

(a) $({}^\bullet h_T(t_0) \cup \underline{h_T(t_0)}) \cap {}^\bullet h_T(t'_0) \neq \emptyset$.
Let $s_1 \in ({}^\bullet h_T(t_0) \cup \underline{h_T(t_0)}) \cap {}^\bullet h_T(t'_0)$. If s_1 is in the initial marking than, by the definition of i-net morphisms, one easily deduces that there exists a unique place $s_0 \in S_0$ such that $h_S(s_0, s_1)$ and moreover $s_0 \in ({}^\bullet t_0 \cup \underline{t_0}) \cap {}^\bullet t'_0$. Therefore, by definition, $\vdash^p_0(\{t'_0\}, t_0, \emptyset)$ and thus, by rule $(\nearrow 1)$, $t_0 \nearrow_0 t'_0$.

Suppose instead that $s_1 \notin m_1$. If $({}^\bullet t_0 \cup \underline{t_0}) \cap {}^\bullet t'_0 \neq \emptyset$ then we conclude as above. Otherwise, as in the case of c-nets (Lemma 3.42), one easily deduces that $t_0 \#_0 t'_0$, and therefore, by rule $(\nearrow 3)$, we can conclude $t_0 \nearrow_0 t'_0$.

(b) $\exists s \in h_T(t_0)^\bullet \cap {}^{\circledcirc}h_T(t'_0) \ \wedge \ s_0{}^\bullet = \emptyset$.
By condition (2.c) in the definition of i-net morphism (Definition 4.33), there must be $s_0 \in t_0{}^\bullet$ such that $h_S(s_0, s_1)$. By condition (2.d) in the same definition, $s_0 \in {}^{\circledcirc}t_0$. Observing that necessarily $s_0{}^\bullet = \emptyset$, we conclude $\vdash^p_0(\{t'_0\}, t_0, \emptyset)$ and thus $t_0 \nearrow_0 t'_0$.

4. $\vdash^p_1(\{h_T(t'_0)\}, h_T(t_0), A_1) \wedge A_1 \neq \emptyset \quad \Rightarrow \quad \exists A_0 \subseteq h_T^{-1}(A_1).\ \exists a_0 \subseteq \{t'_0\}.\ \vdash^p_0(a_0, t_0, A_0)$.
Assume $\vdash^p_{N_1}(\{h_T(t'_0)\}, h_T(t_0), A_1)$ and $A_1 \neq \emptyset$. Thus, by definition of $\vdash^p_{N_1}$ there is a place $s_1 \in {}^{\circledcirc}h_T(t_0) \cap h_T(t'_0)^\bullet$ such that $A_1 = s_1{}^\bullet$. Hence there is $s_0 \in t'_0{}^\bullet$ such that $h_S(s_0, s_1)$. By condition (2.a) in the definition of i-net morphism $h_T(s_0{}^\bullet) \subseteq s_1{}^\bullet = A_1$ and necessarily h_T is defined on each $t'_0 \in s_0{}^\bullet$. Therefore

$$s_0{}^\bullet \subseteq h_T^{-1}(A_1).$$

Since $s_0 \in {}^{\circledcirc}h_T(t_0)$ and $h_S(s_0, s_1)$, by condition (2.d) in the definition of i-net morphism, $s_0 \in {}^{\circledcirc}t_0$. Hence we conclude that, as desired, $\vdash_{N_0}(\{t'_0\}, t_0, s_0{}^\bullet)$. □

By the above proposition we get the existence of a functor which maps each i-net to the corresponding IES defined as in Definition 4.44 and each i-net morphism to its transition component.

DEFINITION 4.46
Let $\mathscr{E}_i$: **O-IN** $\rightarrow$ **IES** *be the functor defined as:*

- $\mathscr{E}_i(N) = I_N$, *for each occurrence i-net* N;
- $\mathscr{E}_i(h : N_0 \rightarrow N_1) = h_T$, *for each morphism* $h : N_0 \rightarrow N_1$.

The coreflection between **IES** and **Dom** can be finally used to obtain a domain semantics, and, by exploiting Winskel's equivalence, a prime event structure semantics for semi-weighted i-nets. As explained in Section 4.2, the PES semantics is obtained from the IES semantics by introducing an event for each possible different history of events in the IES.

4.5.2 From IES's to i-nets: a negative result

In [Win87a] Winskel maps each prime event structure into a canonical occurrence net, via a free construction which generates, for each set of events related in a certain way by the dependency relations, a unique place that induces that kind of relations on the events. In the previous chapter this construction has been generalized to c-nets. This section shows that the result cannot be extended to inhibitor nets and events structures, the problem being essentially the presence of or-causality.

A natural extension of Winskel's construction to inhibitor nets and IES's could be as follows.

DEFINITION 4.47 (FROM OCCURRENCE I-NETS TO IES'S)
Let $I = \langle E, \mapsto \rangle$ *be an* IES. *Then* $\mathscr{N}_i(I)$ *is the net* $N = \langle S, T, F, C, I, m \rangle$ *defined as follows:*

- $m = \left\{ \langle \emptyset, A, B, C \rangle : \begin{array}{l} A, B, C \subseteq E,\ \forall a \in A.\ \forall b \in B.\ a \nearrow b, \\ \#_p B \quad \wedge \quad \forall c \in C.\ \exists B' \subseteq B.\ B' < c \end{array} \right\}$;
- $S = m \cup \left\{ \langle \{e\}, A, B, C \rangle : \begin{array}{l} A, B, C \subseteq E,\ e \in E,\ \forall x \in A \cup B.\ e < x, \\ \forall a \in A.\ \forall b \in B.\ a \nearrow b, \\ \#_p B \quad \wedge \quad \forall c \in C.\ \exists B' \subseteq B. \mapsto (\{e\}, c, B') \end{array} \right\}$;
- $T = E$;
- $F = \langle F_{pre}, F_{post} \rangle$, *with*

$$F_{pre} = \{(e, s) : s = \langle x, A, B, C \rangle \in S,\ e \in B\},$$
$$F_{post} = \{(e, s) : s = \langle e, A, B, C \rangle \in S\};$$

- $C = \{(e, s) : s = \langle x, A, B, C\rangle \in S,\ e \in A\}$.

- $I = \{(e, s) : s = \langle x, A, B, C\rangle \in S,\ e \in C\}$

As one would expect, the main difference with respect to the construction Chapter 3 resides in the fact that here also inhibitor places are added. Since the same kind of dependency among a set of transitions can be induced by connecting the transitions to a place in different ways, all the possibilities must appear in $\mathscr{N}_i(\mathscr{E}_i(N))$. For instance a dependency $e < e'$ may be related to the presence of a place which appears both in the post-set of e and in the pre-set of e'. But it may be induced also by the presence of a place in the initial marking of the net, which inhibits e' and is consumed by e.

A first problem with the described construction is the fact that, in general, it does not map **IES** objects into **O-IN** objects. The reason is essentially the fact that in an occurrence i-net the dependencies induced by the flow and read arcs, and those which are due to the inhibitor arcs are treated differently: the former must satisfy the conditions characterizing occurrence c-nets, while the latter are not constrained in any way. Instead, in an IES all kind of dependencies are represented via the only relation $\mapsto$. Consequently, the distinction between inhibitor and flow/read arcs cannot be recovered from an IES and the net $\mathscr{N}_i(I)$ encodes the (possibly cyclic) situations of dependency among events in all possible ways. For instance, a cycle of causality is represented not only with inhibitor arcs but also with flow arcs. In this way the c-net underlying $\mathscr{N}_i(I)$ may not be an occurrence c-net and thus, in general, $\mathscr{N}_i(I)$ is not an occurrence i-net.

However this is a minor problem, since the problematic situations are clearly related to the non executable events of an IES, i.e., to events which do not appear in any configuration. The presence of such events leads to the generation of nets containing cyclic or non-well founded situations, which do not satisfy the requirements to be an occurrence i-net. The mentioned problem disappear if we remove the non-executable events and work with **IES*** objects. A functor from **O-IN** to **IES***, mapping each occurrence i-net into an IES where all events are executable, can be obtained simply as $\Psi \circ \mathscr{E}_i$, where Ψ : **IES** $\to$ **IES*** is the functor defined in Section 4.2 (Definition 4.27). The construction described in Definition 4.47, applied to a **IES*** object indeed produces an occurrence i-net.

Proposition 4.48
Let I be a **IES*** *object. Then $\mathscr{N}_i(I)$ is an occurrence i-net.*

Proof. Let $\mathscr{N}_i(I) = (S, T, F, C, I, m)$. Then by construction, the initial marking and the pre-set, post-set and context of any transition are sets. Moreover the net is safe since any transition consumes a token in a place in m and thus it can be fired at most once.

Finally, if $\leq$ and $\nearrow$ are the causality and asymmetric conflict relation in the underlying c-net, then it is easy to realize that $\leq$ is a partial order and $\nearrow$ is acyclic on the causes of each transition. In fact one can prove that such relations are included into the corresponding relations of the IES

I, and thus the presence of a non-firable transition in the net would imply the presence of a non-executable event in I. □

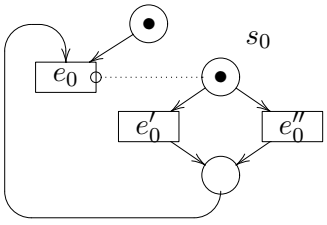
It is not difficult to verify that if I is an **IES*** object then $\mathscr{E}_i(\mathscr{N}_i(I)) \simeq I$. Hence it is still possible to obtain a net representing a given event structure.

However a second problem exists, which instead appears hard to get rid of and which makes impossible to turn the construction into a functor. The problem is illustrated by the following example. Consider two IES's I_0 and I_1, obtained by saturating the pre-IES's $\langle\{e_0, e_0', e_0''\}, \{(\emptyset, e_0, \{e_0', e_0''\})\}\rangle$, and $\langle\{e_1, e_1'\}, \{(\emptyset, e_1, \{e_1'\})\}\rangle$. The nets $\mathscr{N}_i(I_0)$ and $\mathscr{N}_i(I_1)$, are quite complicated, therefore the figure below represents only the parts of these nets which are relevant to our argument.

$\mathscr{N}_i(I_0)$ $\mathscr{N}_i(I_1)$

It is easy to see that the IES-morphism $f : I_0 \to I_1$ defined by $f(e_0') = f(e_0'') = e_1$ and $f(e_0) = e_1'$ is well defined, but it cannot be extended to the places of the nets above in order to obtain an i-net morphism. Formally, the problem is that there is no place in ${}^\bullet e_0$ which can be mapped to s_1.

Conceptually, the problem is related to the fact that in an occurrence i-net a situation of or-causality can be induced only by a place like s_0, which inhibits a transition and is consumed by other transitions. Instead, the or-causality induced by a backward conflict is forbidden, while, as already observed, at the level of IES's, where one forgets the state, the two dependency situations become indistinguishable. Indeed one could be tempted to generate, instead of $\mathscr{N}_i(I_0)$ above, a different net, where or-causality is represented also by backward conflicts (see the picture below).



However, in this way the generated net is not an occurrence i-net, but, by analogy with the flow nets of [Bou90], something which we could call a *flow i-net*. Unfortunately the naïve solution of enlarging the category of occurrence i-net to comprise also such kind of nets does not work, as one can easily check. Therefore, the unfolding construction should be changed in order to generate a flow net instead of an occurrence i-net. However this would be a drastic change, modifying the basic

intuition underlying the unfolding. In fact transitions and places in the "ordinary" unfolding are identified by their history, a concept which, in the presence of backward conflicts, would become unclear.

4.6 Processes of i-nets and their relation with the unfolding

We showed that i-nets can be given a functorial unfolding semantics which generalizes the one defined for c-nets. Unfortunately, the results obtained for c-nets cannot be fully extended to i-nets and the step which leads from occurrence i-nets to inhibitor event structures is not expressed as a coreflection. Nevertheless, the work developed so far allows us to naturally provide i-nets with a (deterministic) *process semantics*, which nicely fits with the notions already existing in the literature. Also a notion of *concatenable i-net process* can be easily defined, and, as already done for c-nets, by exploiting the characterization of the unfolding construction as a coreflection between the categories of semi-weighted i-nets and occurrence i-nets, a tight relationship can be established between the (concatenable) process semantics and the unfolding semantics of an i-net.

We will concentrate only on the peculiar aspects of i-nets, referring the reader to the previous chapter for an extensive explanation of the notions and results which are an easy adaptation of those for c-nets.

The definition of nondeterministic process remains exactly the same. A (nondeterministic) process of an i-net is defined as as on occurrence i-net O_φ with a special kind of morphism, called *strong*, to the original net. A strong morphism is total on places and transitions, and maps places into places (rather than into multisets of places). Furthermore to correctly represent a concurrent computation of the original net N, a strong morphism is required to preserve (and not only to reflect) the inhibitor arcs. In this way the net underlying a process of N turns out to have the same "structure" of the net N.

Definition 4.49 (strong i-net morphism)
An i-net morphism $f : N_0 \to N_1$ is called strong *if f_T and f_S are total functions, and for any place s_0 and transition t_0 in N_0, if $I_0(t_0, s_0)$ then $I_1(f_T(t_0), f_S(s_0))$.*

Observe that if f is a strong morphism then $I_0(t_0, s_0)$ iff $I_1(f_T(t_0), f_S(s_0))$, since the converse implication follows from the definition of general i-net morphisms (see Definition 4.33). Observe that, in particular, the folding morphism is a strong morphism.

Definition 4.50 ((nondeterministic) process)
A marked process *of an i-net $N = \langle S, T, F, C, I, m\rangle$ is a mapping $\varphi : N_\varphi \to N$, where N_φ is an occurrence i-net and φ is a strong i-net morphism. The process is called* discrete *if N_φ has no transitions.*

An unmarked process *of N is defined in the same way, where the mapping φ is an "unmarked morphism", namely φ is not required to preserve the initial marking (it satisfies all conditions of Definition 4.33, but (1)).*

As usual, an isomorphism between two processes of a net N is an isomorphism between the underlying occurrence i-nets, which is consistent with the mapping over the original net N.

A *deterministic occurrence i-net* is defined as an occurrence i-net where all the events can be executed in a single computation. However, differently from what happens for ordinary nets and c-nets, this requirement may not be sufficient to ensure that a deterministic occurrence net is a good representative of a uniquely determined computation. In fact, from the previous considerations on i-nets and IES's it should be clear that computations involving the same events may be different from the point of view of causality. For instance, consider the basic net N_0 of Figure 4.1 with just one inhibitor arc. The transitions t, t' and t_0 may be executed in two different orders, namely $t; t'; t_0$ or $t'; t_0; t$, and, while in the first case it is natural to think of t as a cause of t', in the second case we can imagine that instead t_0 (and thus t') causes t. This further information is taken into account in the so-called *enriched* occurrence i-nets, where a choice relation on the set of transitions specifies which of the possible computations we are referring to.

DEFINITION 4.51 (DETERMINISTIC OCCURRENCE I-NET)
A deterministic occurrence i-net *is an occurrence i-net $O = \langle S, T, F, C, I, m\rangle$ such that, if $\mathscr{E}_i(O) = \langle T_O, \vdash_O\rangle$ is the corresponding* IES *then $\langle T_O, \hookrightarrow_{T_O}\rangle$ is a configuration of $\mathscr{E}_i(O)$ for some choice $\hookrightarrow_{T_O}$. In this case the pair $\langle O, \hookrightarrow_{T_O}\rangle$ is called an* enriched deterministic occurrence i-net*.*

The notion of enriched deterministic occurrence i-net coincides, apart from the slightly different presentation, with that in [Bus98, BP99] (see also Section 2.3). We recall that in such papers an enriched occurrence i-net is defined as an i-net, where the set of inhibitor arcs I is partitioned into two subsets: the "before" arcs I_b and the "after" arcs I_a. Intuitively, if $(t, s) \in I_b$ is a "before" arc then t must be executed before the place s is filled, while if $(t, s) \in I_a$ is an "after" arc then t must be executed after the place s has been emptied. The precedence relation among transitions induced by such a partition is required to be acyclic. A comparison of the two definitions reveals that the only difference resides in the fact that in our case the choice is done directly on transitions, while [Bus98, BP99] imposes requirements on each single inhibiting arc. Therefore the two approaches are equivalent, although we think that imposing precedences directly on transitions is slightly more natural.

DEFINITION 4.52 (DETERMINISTIC PROCESS)
A (marked or unmarked) deterministic process *for an i-net N is a pair $\langle \varphi, \hookrightarrow_\varphi\rangle$, where $\varphi : O \to N$ is a process of N and $\langle O_\varphi, \hookrightarrow_\varphi\rangle$ is an enriched deterministic occurrence i-net.*

A process is called *finite* if the set of transitions in O_φ is finite. In this case, we denote by $\min(\varphi)$ and $\max(\varphi)$ the sets of places in O_φ which have empty pre-set and post-set, respectively. Moreover we denote with ${}^\bullet\varphi$ and $\varphi^\bullet$ the multisets $\mu\varphi_S(\min(\varphi))$ and $\mu\varphi_S(\max(\varphi))$, called respectively the *source* and the *target* of φ. We will usually denote an enriched process simply by φ, and use the symbol $\hookrightarrow_\varphi$ to refer to the associated choice relation.

Two enriched processes φ_1 and φ_2 are isomorphic if there exists a process isomorphism $h : \varphi_1 \to \varphi_2$, whose transition component is an isomorphism of the partial orders $\langle T_{\varphi_1}, \hookrightarrow^*_{\varphi_1}\rangle$ and $\langle T_{\varphi_2}, \hookrightarrow^*_{\varphi_2}\rangle$.

Also the notion of concatenable process can be naturally extended to i-nets. As usual a meaningful operation of sequential composition can be defined only on the unmarked processes and a suitable ordering over the places in $\min(\varphi)$ and $\max(\varphi)$ is needed to distinguish different occurrences of tokens in the same place.

Definition 4.53 (concatenable process)
A concatenable process *of a c-net N is a triple $\gamma = \langle \mu, \varphi, \nu\rangle$, where*

- *φ is a finite deterministic unmarked process of N;*
- *μ is φ-indexed ordering of* $\min(\varphi)$*;*
- *ν is φ-indexed ordering of* $\max(\varphi)$*.*

An isomorphism between concatenable processes is defined, as usual, as a process isomorphism which respects the ordering of the minimal and maximal places. An isomorphism class of processes is called an *(abstract) concatenable process* and denoted by $[\gamma]$, where γ is a member of that class. In the following we will often omit the word "abstract" and write γ to denote the corresponding equivalence class.

The operation of sequential composition of two concatenable processes $\gamma_1 = \langle \mu_1, \varphi_1, \nu_1\rangle$ and $\gamma_2 = \langle \mu_2, \varphi_2, \nu_2\rangle$ of an i-net N is defined as the process obtained by gluing the maximal places of φ_1 and the minimal places of φ_2 according to the ordering of such places. Once the two underlying nets O_1 and O_2 have been glued producing a new net O, a delicate point is the definition of the inhibiting relation in O. In fact, if a place s inhibits a transition t in the original net, then each copy of the place s must inhibit each copy of the transition t. To achieve this result, some inhibitor arcs must be added to O connecting places in O_1 with transitions in O_2, and vice versa. Technically, the resulting net is the unique possible enrichment of O with inhibitor arcs which makes the gluing of the two processes a process of N.

Moreover the choice relation associated to the sequential composition is given by the union of $\hookrightarrow_{\varphi_1}$ and $\hookrightarrow_{\varphi_2}$, with the new dependencies arising for the fact that the two processes are "connected". As for c-nets some of these dependencies are induced by the fact that places in $\max(\varphi_1)$ can be used used by transitions of φ_2. Other dependencies are peculiar of the construction on i-nets, being related to the newly added inhibitor arcs.

DEFINITION 4.54 (SEQUENTIAL COMPOSITION)
Let $\gamma_1 = \langle \mu_1, \varphi_1, \nu_1 \rangle$ and $\gamma_2 = \langle \mu_2, \varphi_2, \nu_2 \rangle$ be two concatenable processes of an i-net N such that ${\varphi_1}^\bullet = {}^\bullet\varphi_2$. Suppose $T_1 \cap T_2 = \emptyset$ and $S_1 \cap S_2 = \max(\varphi_1) = \min(\varphi_2)$, with $\varphi_1(s) = \varphi_2(s)$ and $\nu_1(s) = \mu_2(s)$ for each $s \in S_1 \cap S_2$. In words γ_1 and γ_2 overlap only on $\max(\varphi_1) = \min(\varphi_2)$, and on such places their labelling on the original net and the ordering coincide. Denote by $\varphi : O \to N$ the componentwise union of φ_1 and φ_2. Then the concatenation $\gamma_1;\gamma_2$ is the concatenable process $\gamma = \langle \mu_1, \varphi : O' \to N, \nu_2 \rangle$, where O' is the unique enrichment of O with new inhibitor arcs making φ an i-net morphism, namely $I_{O'}$ is defined as:

$$I_{O'}(t', s') \qquad \textit{iff} \qquad I_N(\varphi(t), \varphi(s)).$$

The choice relation is defined as $\hookrightarrow_{\varphi_1} \cup \hookrightarrow_{\varphi_2} \cup\, r$, where

$$\begin{aligned} r = \;& \{(t_1, t_2) \in T_{\varphi_1} \times T_{\varphi_2} \mid {t_1}^\bullet \cap ({}^\bullet t_2 \cup \underline{t_2}) \neq \emptyset \;\vee\; \underline{t_1} \cap {}^\bullet t_2 \neq \emptyset\} \cup \\ & \{(t_1, t_2) \in T_{\varphi_1} \times T_{\varphi_2} \mid {}^\circledcirc t_1 \cap {t_2}^\bullet \neq \emptyset \;\vee\; {}^\bullet t_1 \cap {}^\circledcirc t_2 \neq \emptyset\} \end{aligned}$$

We remark that, as expected, the choice relation associated to the sequential composition takes care of the precedences induced by the flow and context relations on the transitions connected to the "interface" places. Moreover, whenever a choice is necessary between two transitions $t_1 \in T_{\varphi_1}$ and $t_2 \in T_{\varphi_2}$ because a new inhibitor arc has been added by the construction, the transitions are ordered in the expected way, namely t_1 comes first.

It is not difficult to see that sequential composition is well-defined. The only doubt which may arise regards the acyclicity of the relation $\hookrightarrow^*_\varphi$. But observing that the new dependencies added by the concatenation are only of the kind (t_1, t_2) with $t_i \in T_{\varphi_i}$ for $i \in \{1, 2\}$, one easily realizes that a cycle should be entirely inside one of the two original processes.

The above construction induces a well-defined operation of sequential composition between abstract processes. In particular, if $[\gamma_1]$ and $[\gamma_2]$ are abstract concatenable processes such that ${\gamma_1}^\bullet = {}^\bullet\gamma_2$ then we can always find $\gamma_2' \in [\gamma_2]$ such that $\gamma_1;\gamma_2'$ is defined. Moreover the result of the composition at abstract level, namely $[\gamma_1;\gamma_2']$, does not depend on the particular choice of the representatives.

DEFINITION 4.55 (CATEGORY OF CONCATENABLE PROCESSES)
Let N be an i-net. The category of (abstract) concatenable processes *of N, denoted by $\mathbf{CP}[N]$, is defined as follows. Objects are multisets of places of N, namely elements of μS. Each (abstract) concatenable process $[\langle \mu, \varphi, \nu \rangle]$ of N is an arrow from ${}^\bullet\varphi$ to ${\varphi}^\bullet$.*

In the case of c-nets one can prove that the prime algebraic domain obtained from the unfolding of a net can be characterized as (ideal completion of) the collection of processes starting from the initial marking, endowed with a kind of subprocess order. This result, which establishes a quite intuitive correspondence between the

two kinds of semantics, still holds for i-nets. Let us outline how the approach followed for c-nets can be extended to i-nets.

First, it is immediate to see that for any i-net $N = \langle S, T, F, C, I, m\rangle$ the comma category $\langle m \downarrow \mathbf{CP}[N]\rangle$ is a preorder. The objects of such category are concatenable processes of N starting from the initial marking. An arrow exists from a process γ_1 to γ_2 if the second one can be obtained by concatenating the first one with a third process γ.

Lemma 4.56
Let $N = \langle S, T, F, C, m\rangle$ be an i-net. Then, the comma category $\langle m \downarrow \mathbf{CP}[N]\rangle$ is a preorder.

In the sequel the preorder relation over $\langle m \downarrow \mathbf{CP}[N]\rangle$ (induced by sequential composition) will be denoted by $\lesssim_N$ or simply by $\lesssim$, when the net N is clear from the context. Therefore $\gamma_1 \lesssim \gamma_2$ if there exists γ such that $\gamma_1;\gamma = \gamma_2$.

The main result is then based on a characterization of the preorder relation $\lesssim_N$ in terms of left injections, formalizing the intuition according to which the preorder on $\langle m \downarrow \mathbf{CP}[N]\rangle$ is a generalization of the prefix relation. First, we must introduce an appropriate notion of left-injection for processes of i-nets.

Definition 4.57 (left injection)
Let $\gamma_i : m \to M_i$ $(i \in \{1, 2\})$ be two objects in $\langle m \downarrow \mathbf{CP}[N]\rangle$, with $\gamma_i = \langle \mu_i, \varphi_i, \nu_i\rangle$. A left injection *$\iota : \gamma_1 \to \gamma_2$ is a morphism of marked i-net processes $\iota : \varphi_1 \to \varphi_2$, such that*

1. *ι is consistent with the indexing of minimal places, namely $\mu_1(s) = \mu_2(\iota(s))$ for all $s \in \min(\varphi_1)$;*

2. *$\iota_T : \langle T_{\varphi_1}, \hookrightarrow^*_{\varphi_1}\rangle \to \langle T_{\varphi_2}, \hookrightarrow^*_{\varphi_2}\rangle$ is a rigid embedding.*

As for c-nets, the rigidity condition ensures that γ_2 does not extend γ_1 with transitions which should occur before transitions already in γ_1. Hence the past history of each transition in γ_1 remains the same in γ_2 (and thus the inclusion is also order monic).

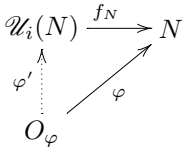
Lemma 4.58
Let $\gamma_i : m \to M_i$ $(i \in \{1, 2\})$ be two objects in $\langle m \downarrow \mathbf{CP}[N]\rangle$, with $\gamma_i = \langle \mu_i, \varphi_i, \nu_i\rangle$. Then

$$\gamma_1 \lesssim \gamma_2 \quad \textit{iff} \quad \textit{there exists a left injection } \iota : \gamma_1 \to \gamma_2.$$

Having this lemma, the theorem which characterizes the domain semantics of a semi-weighted i-net in term of its deterministic processes comes as an easy consequence.

THEOREM 4.59 (UNFOLDING VS. CONCATENABLE PROCESSES)
Let N be a semi-weighted i-net. Then the ideal completion of $\langle m \downarrow \mathbf{CP}[N]\rangle$ is isomorphic to the domain $\mathscr{L}_i(\mathscr{E}_i(\mathscr{U}_i(N)))$.

PROOF (SKETCH). Let $N = \langle S, T, F, C, I, m\rangle$ be an i-net and let $\gamma = \langle \mu, \varphi, \nu\rangle$ be a concatenable process in $\langle m \downarrow \mathbf{CP}[N]\rangle$. Being φ a marked process of N (and thus a i-net morphism $\varphi : O_\varphi \to N$), by the universal property of coreflections, there exists a unique arrow $\varphi' : O_\varphi \to \mathscr{U}_i(N)$, making the diagram below commute.



Obviously, also the converse holds. namely, each process φ' of the unfolding can be turned in a uniquely determined process $\varphi = \varphi'; f_N$ of the net N.

Recall that the compact elements of the domain $\mathscr{L}_i(\mathscr{E}_i(\mathscr{U}_i(N)))$, associated to N are the finite configurations of $\mathscr{E}_i(\mathscr{U}_i(N))$. Therefore we can define a function $\xi : \langle m \downarrow \mathbf{CP}[N]\rangle \to \mathsf{K}(\mathscr{L}_i(\mathscr{E}_i(\mathscr{U}_a(N))))$ as $\xi(\gamma) = (\varphi'_T)^*(\langle T_\varphi, \hookrightarrow_\varphi\rangle)$, where T_φ is the set of transitions of O_φ. The function ξ is well defined since by definition of deterministic process $\langle T_\varphi, \hookrightarrow_\varphi\rangle$ is a configuration of $\mathscr{E}_i(O_\varphi)$ and, by Proposition 4.45, the transition component of an i-net morphism is an IES-morphism, which, by Lemma 4.22, maps configurations into configurations.

Proceeding as for c-nets it is possible to show that the function ξ is surjective, basically because each configuration of the IES associated to the unfolding determines a deterministic subnet of the unfolding. The corresponding process of N, enriched with the order associated to the configuration, is mapped to the original configuration. The fact that ξ is monotone and reflects the order can be proved by exploiting Lemma 4.58. The existence of the function ξ with the above mentioned properties, by a general result on preorders (see Lemma 3.66), allows us to conclude that $\mathsf{Idl}(\langle m \downarrow \mathbf{CP}[N]\rangle)$ and $\mathscr{L}_i(\mathscr{E}_i(\mathscr{U}_i(N)))$ are isomorphic. □

Final Remarks on Part I

The main contribution of the first part of the thesis is a truly concurrent event-based semantics for (semi-weighted) P/T nets extended with context and inhibitor arcs.

For the simpler case of *contextual nets* the semantics is given at categorical level via a coreflection between the categories **SW-CN** of semi-weighted c-nets and **Dom** of finitary coherent prime algebraic domains (or equivalently **PES** of prime event structures). Such a coreflection factorizes through the following chain of coreflections:

$$\mathbf{SW\text{-}CN} \underset{\mathscr{U}_a}{\overset{\mathscr{J}_O}{\underset{\perp}{\rightleftarrows}}} \mathbf{O\text{-}CN} \underset{\mathscr{E}_a}{\overset{\mathscr{N}_a}{\underset{\perp}{\rightleftarrows}}} \mathbf{AES} \underset{\mathscr{L}_a}{\overset{\mathscr{P}_a}{\underset{\perp}{\rightleftarrows}}} \mathbf{Dom} \underset{\mathscr{P}}{\overset{\mathscr{L}}{\underset{\sim}{\rightleftarrows}}} \mathbf{PES}$$

A key role in the treatment of contextual nets is played by *asymmetric event structures*, an extension of Winskel's (prime) event structures (with binary conflict), introduced to deal with asymmetric conflicts. Asymmetric event structures are closely related to other models in the literature, like PES's with possible events [PP95], flow event structures with possible flow [PP95] and extended bundle event structures [Lan92b]. However, none of the above models was adequate for representing the behaviour of contextual nets: PES's with possible events are not sufficiently expressive, while the other two models look too general and unnecessarily complex for the treatment of contextual nets, due to their capability of expressing multiple disjunctive causes for an event. More technically, as it follows form the discussion in Section 4.5, the possibility of expressing or-causality in these models would have prevented us from realizing the step from occurrence c-nets to event structures as a coreflection.

Independently from the conference version of CHAPTER 3, appeared as [BCM98b], an unfolding construction for contextual nets has been proposed by Vogler, Semenov and Yakovlev in [VSY98]. The unfolding of [VSY98] is carried out in the case of safe finite c-nets and without providing a categorical characterization of the constructions. Anyway, apart from some matters of presentation, it is based on ideas analogous to ours and it leads to the same result. A very interesting result in that paper is the extension of the McMillan algorithm for the construction of a finite prefix of the unfolding to a subclass of contextual nets, called read-persistent contextual nets. The algorithm is then applied to the analysis of asynchronous circuits. We are confident that the result in CHAPTER 3, and in particular the notion of set of possible histories of an event in a contextual net, may ease the extension

of the technique proposed in [VSY98] to the whole class of semi-weighted c-nets (perhaps at the price of a growth of the complexity).

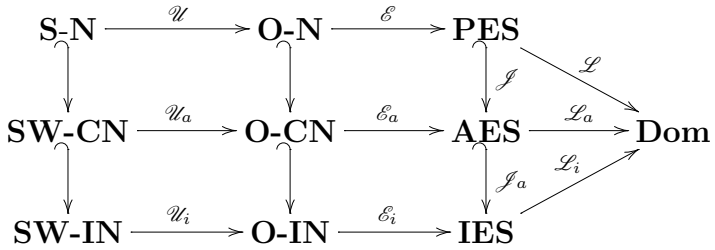
The treatment of inhibitor nets requires the introduction of *inhibitor event structures*, a new event structure model which properly generalizes AES's. In such structures a relation, called *disabling-enabling relation*, allows one to model, in a compact way, the presence of disjunctive conflicting causes and the situations of relative atomicity of pairs of events with respect to a third one, determined by inhibitor arcs.

The truly concurrent semantics for inhibitor nets is given via a chain of functorial constructions leading from the category **SW-IN** of semi-weighted i-nets to the category **Dom** of finitary prime algebraic domains:

$$\mathbf{SW\text{-}IN} \underset{\mathscr{U}_i}{\overset{\mathscr{I}_O}{\overset{\longleftarrow}{\underset{\longrightarrow}{\perp}}}} \mathbf{O\text{-}IN} \xrightarrow[\mathscr{E}_i]{} \mathbf{IES} \underset{\mathscr{L}_i}{\overset{\mathscr{P}_i}{\overset{\longleftarrow}{\underset{\longrightarrow}{\perp}}}} \mathbf{Dom} \underset{\mathscr{P}}{\overset{\mathscr{L}}{\overset{\longleftarrow}{\underset{\longrightarrow}{\sim}}}} \mathbf{PES}$$

The unfolding and its characterization as a universal construction are "lifted" from contextual to inhibitor nets. Unfortunately, in this more complex case, we cannot fully generalize Winskel's chain of coreflections. The problem is the absence of a functor performing the backward step from IES's to occurrence inhibitor nets. As shown in Section 4.5, this is essentially due to the presence of or-causality in inhibitor event structures and, under reasonable assumptions on the notions of occurrence net and of unfolding, the problem has no solutions.

It is worth noticing that the construction on inhibitor nets is a conservative extension of those on contextual nets, which in turn extends Winskel's [Win87a]. More precisely, as one can easily grasp from the discussion in the previous chapters, the following diagram, where unnamed functors are inclusions, commutes.



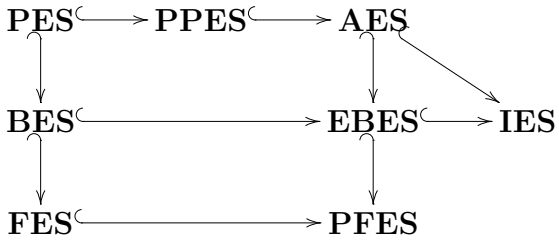
The truly concurrent semantics of contextual and inhibitor nets given via the unfolding has been shown to be closely related to various *deterministic process semantics* proposed in the literature. The natural notion of deterministic process arising from our theory coincides with the other notions in the literature (at least with those developed assuming the same notion of enabling). A formal relationships between the unfolding and the deterministic process semantics has been established by showing that the domain semantics of a net, given by the above described chains of functors, is isomorphic to the set of deterministic processes of the net starting from the initial marking, endowed with a kind of prefix ordering.

As mentioned above, the capability of expressing asymmetric conflicts in computations makes asymmetric event structures close to other extensions of event structures in the literature. Asymmetric event structures can be seen as a generalization of *prime event structures with possible* events (PPES's) [PP95]. While for AES's the asymmetric conflict relation essentially specifies for each event the set of its possible causes, which may appear or not in the history of the events, in the case of PPES's there is a distinguished global set of possible events. As observed in CHAPTER 3, PPES's are strictly less expressive than AES's.

On the other hand, an explicit asymmetric conflict relation has been considered in *extended bundle event structures* (EBES's) [Lan92b], where there is also the possibility of expressing a set of disjunctive conflictual causes for an event, called a *bundle*. In turn, EBES are generalized by *flow event structure with possible flow* (PFES's) in the same way as *bundle event structures* (BES) [Lan92a] are generalized by *flow event structures* (see Section 2.4).

Inhibitor event structures, due to their capability of expressing, by means of the DE-relation, both asymmetric conflicts and sets of conflictual disjunctive causes, generalize AES's and EBES's. The relation between IES's and PFES is still to be investigated, although likely the two models are not comparable.

The following diagram represents a hierarchy of expressiveness of the mentioned event structure models.



Since the functor $\mathscr{P}_i$: **Dom** $\rightarrow$ **IES** is obtained by composing Winskel's functor $\mathscr{P}$: **Dom** $\rightarrow$ **PES** with the inclusion of **PES** into **IES**, it is easy to see that the coreflection between **IES** and **Dom** proved in Theorem 4.26 restricts to a coreflection between each model which is included in **IES**, and **Dom**.

We also observe that the construction which associates the domain of configurations to an AES can be easily modified to generate an *event automata* [PP95] (see Section 2.4). Given an IES $I = \langle E, \mapsto \rangle$, the corresponding event automata $\mathscr{A}(I)$ is obtained by considering the domain of configurations of I and forgetting the choice relations of configurations. More formally

$$\mathscr{A}(I) = \langle E, St, \rightarrowtriangle \rangle,$$

where $St = \{C \subseteq E \mid \exists \hookrightarrow_C .\ \langle C, \hookrightarrow_C \rangle \in Conf(I)\}$, and $C_1 \rightarrowtriangle C_2$ whenever for $i \in \{1, 2\}$ there is a choice relation $\hookrightarrow_i$ such that $\langle C_i, \hookrightarrow_i \rangle \in Conf(I)$ and $\langle C_1, \hookrightarrow_1 \rangle \prec \langle C_2, \hookrightarrow_2 \rangle$. The above construction gives rise to a functor $\mathscr{A}$: **IES** $\rightarrow$ **EvAut**.

We already mentioned that Winskel's construction has been generalized in [MMS96] not only to the subclass of semi-weighted P/T nets, but also to the full class of P/T nets. In the last case, some additional effort is needed and only a proper adjunction rather than a coreflection can be obtained. We think that also the results on contextual and inhibitor nets can be extended to the full class of P/T nets, by following the guidelines traced in [MMS96] and exploiting, in particular, suitable generalizations to c-nets and i-nets of the notions of decorated occurrence net and of family morphism introduced in that work.

Part II

Semantics of DPO Graph Grammars

Chapter 5

Typed Graph Grammars in the DPO Approach

This chapter provides an introduction to the *algebraic approach* to graph transformation based on the *double-pushout* (DPO) construction [Ehr79]. We first give the basic definitions of typed DPO graph grammar, rewriting step and derivation, formalizing the intuitive description of the rewriting mechanism described in the INTRODUCTION. This allows us also to give a more precise account of the relationship between DPO typed graph grammar and P/T Petri nets. Then we introduce the fundamental notions of the concurrency theory of DPO graph transformation by presenting the *trace semantics* based on the *shift equivalence* [Kre77, CEL$^+$94a, CMR$^+$97]. The presentation slightly differs from the classical one since we adopt an equivalent, in our opinion more convenient, approach to the sequential composition of traces, based on *canonical graphs* rather than on *standard isomorphisms*. Finally, we introduce the *category of graph grammars* we shall work with. Our morphisms on graph grammars, which arise as a generalization of Winskel's morphisms for Petri nets, are a slight variation of the morphisms in [CEL$^+$96a].

5.1 Basic definitions

A common element of all the algebraic approaches to graph rewriting is the use of category theory as a basic tool: the structures on which the rewriting takes place are turned into a category $\mathbf{C}$ and then the fundamental notions and constructions are expressed via diagrams and constructions in $\mathbf{C}$. As a consequence the resulting theory turns out to be very general and flexible, easily adaptable to a wide range of structures (from several kind of graphs to more general structures [EKT94]) just changing the underlying category.

Originally the DPO approach to graph transformation has been defined for labelled graphs. Following a slightly different but well-established approach, here we consider a generalization of basic graph grammars called *typed* graph gram-

mars [CMR96], where a more sophisticated labelling technique for graphs is considered: each graph is typed on a structure that is itself a graph (called the *graph of types*) and the labelling function is required to be a graph morphism, i.e., it must preserve the graphical structure. This gives, in a sense, more control on the labelling mechanism. From the formal side, working with typed graph grammars just means changing the category over which the rewriting takes place, from labelled graphs to typed graphs.

DEFINITION 5.1 (GRAPH)
A (directed, unlabelled) graph *is a tuple* $G = \langle N_G, E_G, s_G, t_G \rangle$*, where* N_G *is a set of* nodes*,* E_G *is a set of* edges*, and* $s_G, t_G : E_G \to N_G$ *are the* source *and* target *functions. A graph is* discrete *if it has no edges. A* graph morphism $f : G \to G'$ *is a pair of functions* $f = \langle f_N : N_G \to N_{G'}, f_E : E_G \to E_{G'} \rangle$ *which preserve sources and targets, i.e., such that* $f_N \circ s_G = s_{G'} \circ f_E$ *and* $f_N \circ t_G = t_{G'} \circ f_E$*; it is an* isomorphism *if both* f_N *and* f_E *are bijections; moreover, an* abstract graph $[G]$ *is an isomorphism class of graphs, i.e.,* $[G] = \{H \mid H \simeq G\}$*. An* automorphism *of* G *is an isomorphism* $h : G \to G$*; it is* non-trivial *if* $h \neq id_G$*. The category having graphs as objects and graph morphisms as arrows is called* **Graph**.

In the following it will be useful to consider graphs as unstructured sets of nodes and edges. For a given graph G, with $Items(G)$ we denote the disjoint union of nodes and edges of G; for simplicity, we will assume that all involved sets are disjoint, to be allowed to see $Items(G)$ as a set-theoretical union. Often, when the meaning is clear from the context, we will identify a graph G with the set $Items(G)$, writing, for instance, $x \in G$ instead of $x \in Items(G)$.

DEFINITION 5.2 (CATEGORY OF TYPED GRAPHS)
Given a graph TG*, a* typed graph G *over* TG *(or* TG*-typed graph) is a graph* $|G|$*, together with a morphism* $t_G : |G| \to TG$*. A morphism between* TG*-typed graphs* $f : G_1 \to G_2$ *is a graph morphism* $f : |G_1| \to |G_2|$ *consistent with the typing, i.e., such that* $t_{G_1} = t_{G_2} \circ f$*. A typed graph* G *is called* injective *if the typing morphism* t_G *is injective. The category of* TG*-typed graphs and typed graph morphisms is denoted by* TG-**Graph** *and can be sinthetically defined as the category of graphs over* TG*, i.e.,* $\langle \mathbf{Graph} \downarrow TG \rangle$ *(see Definition A.22 in the* APPENDIX*).*

In the sequel, if TG is clear from the context, TG-type graphs will be called simply *typed graphs* and, similarly, morphisms of TG-typed graphs will be called *typed graph morphisms*.

By general categorical arguments, the fact that TG-**Graph** is defined via a comma category construction ensures us that it retains the properties of completeness and cocompleteness of **Graph**. Limits and colimits of diagrams in TG-**Graph** can be computed in **Graph**. Hence, to move from labelled to typed graphs we just syntactically replace the category of labelled graphs with that of typed graphs in all the definitions and results of the DPO approach to graph transformation. Furthermore, it is worth remarking that the typing mechanism subsumes the usual

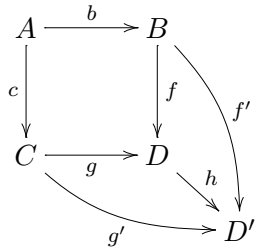


Figure 5.1: Pushout diagram.

labelling technique, where there are two label alphabets Ω_N for nodes and Ω_E for edges, and, correspondingly, two labelling functions. In fact, consider the graph of types $TG_\Omega = \langle \Omega_N, \Omega_E \times \Omega_N \times \Omega_N, s, t \rangle$, with $s(e, n_1, n_2) = n_1$ and $t(e, n_1, n_2) = n_2$. It is easily seen that there is an isomorphism between the category of labelled graphs over $\langle \Omega_N, \Omega_E \rangle$ and the category of TG_Ω-typed graphs.

The core of the algebraic approach to graph transformation is the idea of expressing the *gluing of graphs* in categorical terms as a *pushout construction* (see also Definition A.17 in the APPENDIX). We next recall the notions of pushout and pushout complement, and the underlying intuition which will guide the definition of the rewriting mechanism.

DEFINITION 5.3 (PUSHOUT)
Let $\mathbf{C}$ *be a category and let* $b : A \to B$, $c : A \to C$ *be a pair of arrows of* $\mathbf{C}$. *A triple* $\langle D, f : B \to D, g : C \to D \rangle$ *as in Figure 5.1 is called a* pushout *of* $\langle b, c \rangle$ *if the following conditions are satisfied:*

[Commutativity]
$f \circ b = g \circ c$;

[Universal Property]
for any object D' *and arrows* $f' : B \to D'$ *and* $g' : C \to D'$, *with* $f' \circ b = g' \circ c$, *there exists a unique arrow* $h : D \to D'$ *such that* $h \circ f = f'$ *and* $h \circ g = g'$.

In this situation, D *is called a* pushout object *of* $\langle b, c \rangle$. *Moreover, given arrows* $b : A \to B$ *and* $f : B \to D$, *a* pushout complement *of the pair* $\langle b, f \rangle$ *is a triple* $\langle C, c : A \to C, g : C \to D \rangle$ *such that* $\langle D, f, g \rangle$ *is a pushout of* b *and* c. *In this case* C *is called a* pushout complement object *of* $\langle b, f \rangle$.

In the category **Set** of sets and (total) functions the pushout object can be characterized as $D = (B + C)/_\equiv$, where $\equiv$ is the least equivalence on $B + C = \{\langle 0, x \rangle \mid x \in B\} \cup \{\langle 1, y \rangle \mid y \in C\}$ such that, for each $a \in A$, $\langle 0, b(a) \rangle \equiv \langle 1, c(a) \rangle$, or in words, D is the disjoint union of B and C, where the images of A through b and c are equated. The morphisms f and g map each element of B and C, respectively,

to the corresponding equivalence class in D. One can see that, analogously, in the category of graphs and (total) graph morphisms the pushout object can be thought of as the *gluing* of B and C, obtained by identifying the images of A through b and c. According to this interpretation, the pushout complement object C of b and f, is obtained by removing from D the elements of $f(B)$ which are not images of $b(A)$.

A typed production in the double-pushout approach is a *span*, i.e., a pair of typed graph morphisms with common source. Moreover each production has an associated name which allows one to distinguish productions with the same associated span. Such a name plays no role when the production is applied to a graph, but it is relevant in certain transformations of derivations and when relating different derivations.

DEFINITION 5.4 (TYPED PRODUCTION)
A (TG-typed graph) production $(L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R)$ *is a pair of injective typed graph morphisms* $l : K \rightarrow L$ *and* $r : K \rightarrow R$*, with* $|L|$, $|K|$ *and* $|R|$ *finite graphs. It is called* consuming *if morphism* $l : K \rightarrow L$ *is not surjective. The typed graphs* L*,* K*, and* R *are called the* left-hand side*, the* interface*, and the* right-hand side *of the production, respectively.*

Although sometimes we will consider derivations starting from a generic typed graph, a typed graph grammar comes equipped with a start graph, playing the same role of the initial symbol in string grammars, or of the initial marking for Petri nets. Conceptually, it represents the initial state of the system modelled by the grammar.

DEFINITION 5.5 (TYPED GRAPH GRAMMAR)
A (TG-typed) graph grammar $\mathscr{G}$ *is a tuple* $\langle TG, G_s, P, \pi \rangle$*, where* G_s *is the* start (typed) graph*,* P *is a set of* production names*, and* π *a function mapping each production name in* P *to a graph production. Sometimes to indicate that* $\pi(q) = (L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R)$ *we shall write* $q : (L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R)$*. The grammar* $\mathscr{G}$ *is called* consuming *if all its productions are consuming, and* finite *if the set of productions* P *is finite.*

Since here we work only with typed notions, when it is clear from the context we will often omit the qualification "typed" and do not indicate explicitly the typing morphisms.

REMARK 5.6 (CONSUMING GRAMMARS)
In the following we will implicitly assume that all the considered graph grammars are consuming. As already discussed for Petri nets, this restriction becomes essential only when developing a semantics based on an unfolding construction. In fact, occurrence grammars, event structures and domains are not suited to represent computations of non-consuming grammars: in the presence of a non-consuming production $q : L \stackrel{\leftarrow}{l} K \stackrel{\rightarrow}{r} R$ where the set $L - l(K)$, which can be thought of as the pre-set of q, is empty, an unbounded number of indistinguishable copies of production q can be applied in parallel in a derivation. Instead, at the price of some

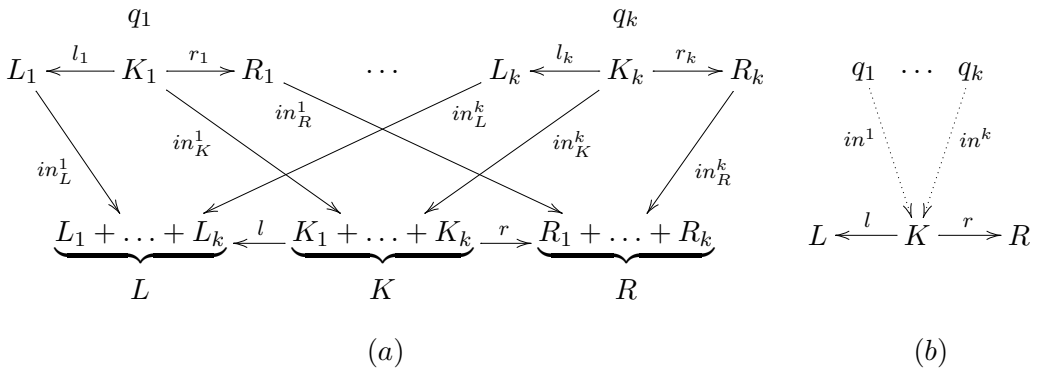


Figure 5.2: (a) The parallel production $\langle (q_1, in^1), \ldots, (q_k, in^k) \rangle : (L \xleftarrow{l} K \xrightarrow{r} R)$ and (b) its compact representation.

technical complications, a (deterministic) process semantics can be still developed for general grammars (see, e.g., [BCE+99]).

The application of a production produces a *local* transformation in the rewritten graph; hence the idea of allowing for the concurrent application of more than one production naturally emerges. The idea of "parallel composition" of productions is naturally formalized in the categorical setting by the notion of parallel production.

DEFINITION 5.7 ((TYPED) PARALLEL PRODUCTIONS)
A (typed) parallel production *(of a given typed graph grammar $\mathscr{G}$) has the form* $\langle (q_1, in^1), \ldots, (q_k, in^k) \rangle : (L \xleftarrow{l} K \xrightarrow{r} R)$ *(see Figure 5.2), where* $k \geq 0$, $q_i : (L_i \xleftarrow{l_i} K_i \xrightarrow{r_i} R_i)$ *is a production of $\mathscr{G}$ for each* $i \in \underline{k}$,[1] *L is a coproduct object of the typed graphs in* $\langle L_1, \ldots, L_k \rangle$, *and similarly R and K are coproduct objects of* $\langle R_1, \ldots, R_k \rangle$ *and* $\langle K_1, \ldots, K_k \rangle$, *respectively. Moreover, l and r are uniquely determined, using the universal property of coproducts, by the families of arrows* $\{l_i\}_{i \in \underline{k}}$ *and* $\{r_i\}_{i \in \underline{k}}$, *respectively. Finally, for each* $i \in \underline{k}$, in^i *denotes the triple of injections* $\langle in^i_L : L_i \to L, in^i_K : K_i \to K, in^i_R : R_i \to R \rangle$. *The* empty *production is the (only) parallel production with* $k = 0$, *having the empty graph* $\emptyset$ *(initial object in TG-***Graph***) as left-hand side, right-hand side and interface, and it is denoted by* $\emptyset$.

We will often denote the parallel production of Figure 5.2.(a) simply as $q_1 + q_2 + \ldots + q_k : (L \xleftarrow{l} K \xrightarrow{r} R)$; note however that the "+" operator is not assumed to be commutative. We will also use the more compact drawing of Figure 5.2.(b) to depict the same parallel production. Furthermore, we will freely identify a production q of $\mathscr{G}$ with the parallel production $\langle (q, \langle id_L, id_K, id_R \rangle) \rangle$; thus, by default, productions will be parallel in the rest of the thesis.

[1] For each $n \in \mathbb{N}$, by $\underline{n}$ we denote the set $\{1, 2, \ldots, n\}$ (thus $\underline{0} = \emptyset$).

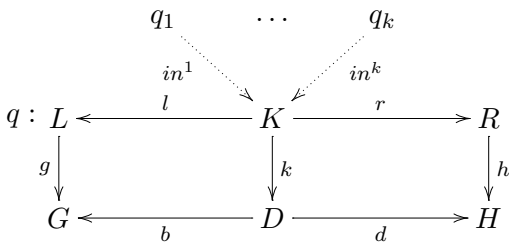


Figure 5.3: (Parallel) direct derivation as double-pushout construction.

The rewriting procedure involves two pushout diagrams in the category of graphs, hence the name of double-pushout approach.

DEFINITION 5.8 ((PARALLEL) DIRECT DERIVATION)
Given a typed graph G, a parallel production $q = q_1 + \ldots + q_k : (L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R)$, and a match *(i.e., a graph morphism) $g : L \rightarrow G$, a* (parallel) direct derivation δ from G to H using q (based on g) *exists if and only if the diagram in Figure 5.3 can be constructed, where both squares are required to be pushouts in TG-***Graph***. In this case, D is called the* context *graph, and we write $\delta : G \Rightarrow_q H$, or also $\delta : G \Rightarrow_{q,g} H$; only seldom we shall write $\delta : G \stackrel{\langle g,k,h,b,d \rangle}{\Longrightarrow_q} H$, indicating explicitly all the morphisms of the double-pushout. If $q = \emptyset$, i.e., if q is the empty production, then $G \Rightarrow_\emptyset H$ is called an* empty direct derivation.

EXAMPLE 5.9 (CLIENT-SERVER SYSTEMS)
As a running example we will use the simple typed graph grammar shown in Figure 5.4, which models the evolution of client-server systems (this is a little modification of an example from [CMR+97]). The typing morphisms from the involved graphs to the graph of types TG are not depicted explicitly, but they are encoded by attaching to each item its type, i.e., its image in TG, separated by a colon. We use natural numbers for nodes, and underlined numbers for edges. For example, the node 4 of the start graph G_0 is typed over the node C of TG.

A graph typed over TG represents a possible configuration containing servers and clients (denoted by nodes of types S and C, respectively), which can be in various states, as indicated by edges. A loop of type *job* on a client means that the client is performing some internal activity, while a loop of type *req* means that the client issued a request. An edge of type *busy* from a client to a server means that the server is processing a request issued by the client.

Production *REQ* models the issuing of a request by a client. After producing the request, the client continues its internal activity (*job*), while the request is served asynchronously; thus a request can be issued at any time, even if other requests are pending or if the client is being served by a server. Production *SER* connects a client

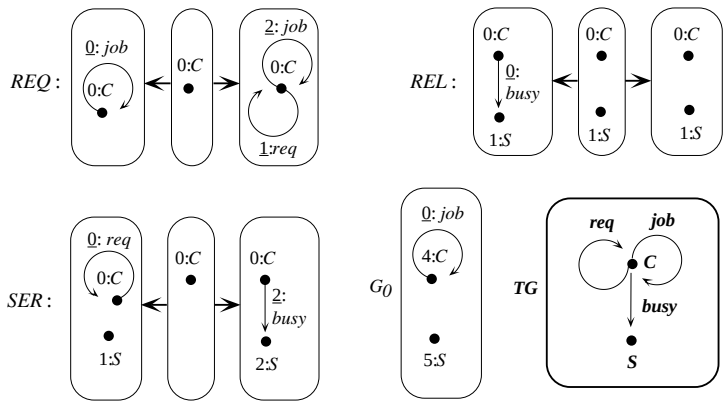


Figure 5.4: Productions, start graph and graph of types of the grammar $\mathscr{C}$-$\mathscr{S}$ modelling client-server systems.

that issued a request with a server through a *busy* edge, modelling the beginning of the service. Since the production deletes a node of type S and creates a new one, the *dangling condition* (see below) ensures that it will be applied only if the server has no incoming edges, i.e., if it is not busy. Production *REL* (for *release*) disconnects the client from the server (modelling the end of the service). Notice that in all rules the graph morphisms are inclusions.

The grammar is called $\mathscr{C}$-$\mathscr{S}$ (for *client-server*), and it is formally defined as $\mathscr{C}\text{-}\mathscr{S} = \langle TG, G_0, \{REQ,\ SER,\ REL\}, \pi \rangle$, where π maps the production names to the corresponding production spans depicted in Figure 5.4. □

To have an informal understanding of the notion of direct derivation, one should recall the interpretation of the pushout as gluing of objects. According to this interpretation, the rewriting procedure removes from the graph G the images via g of the items of the left-hand side which are not in the image of the interface, namely $g(L - l(K))$, producing in this way the graph D. Then the items in the right-hand side, which are not in the image of the interface, namely $R - r(K)$, are added to D, obtaining the final graph H. Thus the interface K (common part of L and R) specifies what is preserved. For what regards the *applicability* of a production to a given match, it is possible to prove that the situation in the category TG-**Graph** is exactly the same as in **Graph**, namely pushouts always exist, while for the existence of the pushout complement some conditions have to be imposed, called *gluing conditions* [Ehr87], which consist of two parts:

[Dangling condition]
No edge $e \in G - g(L)$ is incident to any node in $g(L - l(K))$;

[Identification condition]
There is no $x, y \in L$, $x \neq y$, such that $g(x) = g(y)$ and $y \notin l(K)$.

Nicely, the gluing conditions have a very intuitive interpretation: the dangling condition avoids the deletion of a node if some edge is still pointing to it, and thus it ensures the absence of dangling edges in D. The identification condition requires each item of G which is deleted by the application of q, to be the image of only one item of L. Among other things, this ensures that the application of a production cannot specify simultaneously both the preservation and the deletion of an item and furthermore, that a single item of G cannot be deleted "twice" by the application of q. Notice that, instead, the identification condition does not forbid the match to be non-injective on preserved items. Intuitively this means that preserved (read-only) resources can be used with multiplicity greater than one (see [CMR+97] for a broader discussion). Uniqueness of the pushout complement, up to isomorphism, follows from the injectivity of l.

REMARK 5.10
If $G \overset{\langle g,k,h,b,d\rangle}{\Rightarrow_\emptyset} H$ is an *empty direct derivation*, then morphisms g, k, and h are necessarily the only morphisms from the empty (typed) graph (since $\langle\emptyset,\emptyset\rangle$ is initial in TG-**Graph**), while b and d must be isomorphisms. Morphism $d \circ b^{-1} : G \to H$ is called the *isomorphism induced* by the empty direct derivation. Moreover, for any pair of isomorphic graphs $G \simeq H$, there is an empty direct derivation $G \overset{\langle\emptyset,\emptyset,\emptyset,b,d\rangle}{\Rightarrow_\emptyset} H$ for each triple $\langle D, b : D \to G, d : D \to H\rangle$, where b and d are isomorphisms. An empty direct derivation $G \overset{\langle\emptyset,\emptyset,\emptyset,b,d\rangle}{\Rightarrow_\emptyset} H$ will be also briefly denoted as $G \overset{\langle b,d\rangle}{\Rightarrow_\emptyset} H$. □

A parallel derivation can be seen as a sequence of single steps of the system, each one consisting of the concurrent execution of a set of independent basic actions of the system, analogously to step sequences of Petri nets.

DEFINITION 5.11 ((PARALLEL) DERIVATION)
A (parallel) derivation (over $\mathscr{G}$) *is either a graph G (called an* identity derivation*, and denoted by $G : G \Rightarrow^* G$), or a sequence of (parallel) direct derivations $\rho = \{G_{i-1} \Rightarrow_{q_i} G_i\}_{i\in\underline{n}}$ such that $q_i = q_{i1} + \ldots + q_{ik_i}$ is a (parallel) production over $\mathscr{G}$ for all $i \in \underline{n}$ (as in Figure 5.6). In the last case, the derivation is written $\rho : G_0 \Rightarrow^*_{\mathscr{G}} G_n$ or simply $\rho : G_0 \Rightarrow^* G_n$. If $\rho : G \Rightarrow^* H$ is a (possibly identity) derivation, then the graphs G and H, called the* source *and* target graphs *of ρ, are denoted by $\sigma(\rho)$ and $\tau(\rho)$, respectively. The* length *of a derivation ρ, denoted by $|\rho|$, is the number of direct derivations in ρ (hence $|\rho| = 0$ if ρ is an identity derivation). The* order *of ρ, denoted by $\#\rho$, is the total number of elementary productions used in ρ, i.e., $\#\rho = \sum_{r=1}^{n} k_r$; moreover, $prod(\rho) : \underline{\#\rho} \to P$ is the function returning for each j the name of the j-th production applied in ρ—formally, $prod(\rho)(j) = q_{is}$ if $j = \left(\sum_{r=1}^{i} k_r\right) + s$. The* sequential composition *of two derivations ρ and ρ' is defined if and only if $\tau(\rho) = \sigma(\rho')$; in this case it is denoted $\rho\,;\,\rho' : \sigma(\rho) \Rightarrow^* \tau(\rho')$, and it is the diagram obtained by identifying $\tau(\rho)$ with $\sigma(\rho')$ (thus if $\rho : G \Rightarrow^* H$, then $G\,;\,\rho = \rho = \rho\,;\,H$, where G and H are the identity derivations).*

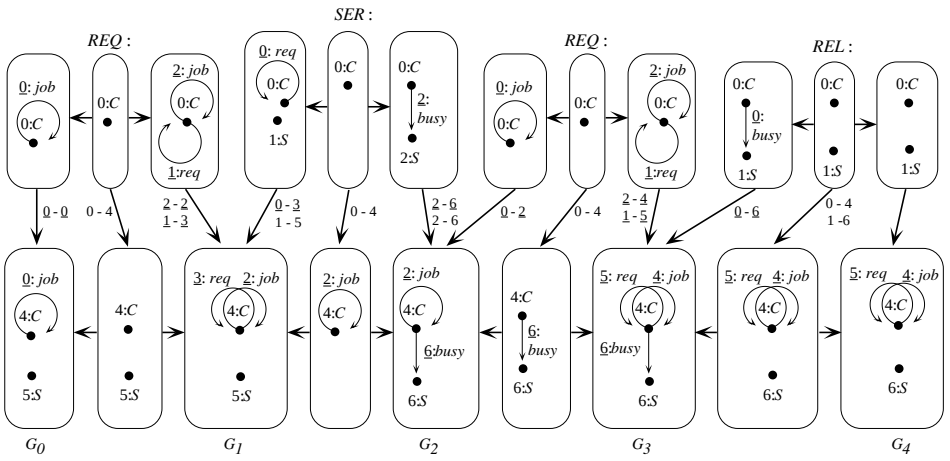


Figure 5.5: A derivation of grammar $\mathscr{C}$-$\mathscr{S}$ starting from graph G_0.

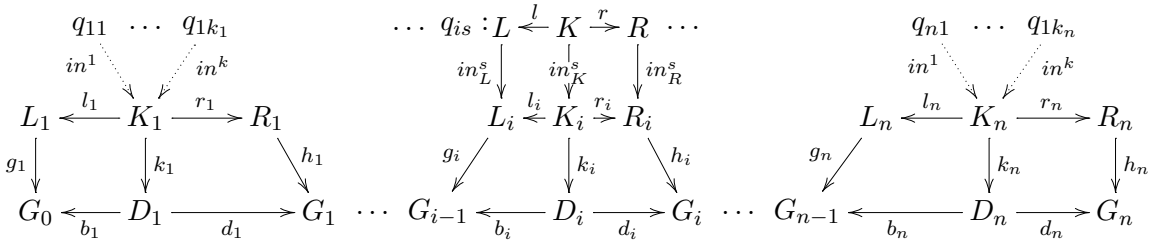


Figure 5.6: A (parallel) derivation, with explicit drawing of the s-th production of the i-th direct derivation.

EXAMPLE 5.12 (DERIVATION)
Figure 5.5 shows a derivation ρ using grammar $\mathscr{C}$-$\mathscr{S}$ and starting from the start graph G_0. The derivation models the situation where a request is issued by the client, and while it is handled by the server, a new request is issued. All the horizontal morphisms are inclusions, while the vertical ones are annotated by the relation on graph items they induce. □

5.1.1 Relation with Petri nets

The basic notions introduced so far allow us to give a more precise account of the relation between Petri nets and graph grammars in the double pushout approach.

Being Petri nets one of the most widely accepted models for the representation of concurrent and distributed systems, people working on the concurrency theory of graph grammars have been naturally led to compare their formalisms with nets. Therefore various encodings of nets into graph graph grammars have been proposed along the years, all allowing to have some correspondences between

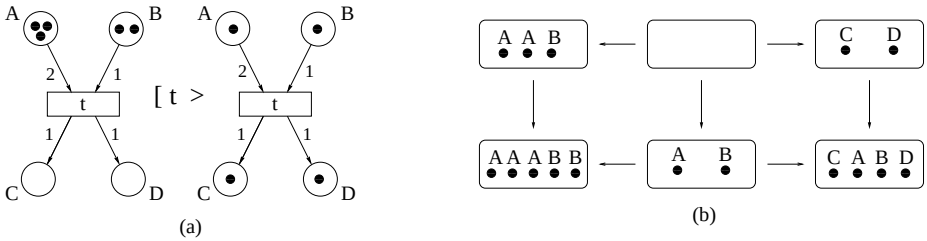


Figure 5.7: Firing of a transition and corresponding DPO derivation.

net-related notions and graph-grammars ones. Some encodings involving the DPO approach can be found in [Kre81, KW86, Sch93] (for a complete survey the reader can consult [Cor96]). All these papers represent a net as a grammar by explicitly encoding in the start graph the topological structure of the net as well as its initial marking.

Here we refer to a slightly simpler modelling (see, e.g., [Cor96]), which can help in understanding how the theory developed in the FIRST PART for contextual and inhibitor nets can be generalized to graph grammars. It is based on the simple observation that a Petri net is essentially a rewriting system on multisets, and that, given a set A, a multiset of A can be represented as a discrete graph typed over A. In this view a P/T Petri net can be seen as a graph grammar acting on discrete graphs typed over the set of places, the productions being (some encoding of) the net transitions: a marking is represented by a set of nodes (tokens) labelled by the place where they are, and, for example, the unique transition t of the net in Figure 5.7.(a) is represented by the graph production in the top row of Figure 5.7.(b): such production consumes nodes corresponding to two tokens in A and one token in B and produces new nodes corresponding to one token in C and one token in D. The interface is empty since nothing is explicitly preserved by a net transition. Notice that in this encoding the topological structure of the net is not represented at all: it is only recorded in the productions corresponding to the transitions.

It is easy to check that this representation satisfies the properties one would expect: a production can be applied to a given marking if and only if the corresponding transition is enabled, and the double pushout construction produces the same marking as the firing of the transition. For instance, the firing of transition t, leading from the marking $3A + 2B$ to the marking $A + B + C + D$ in Figure 5.7.(a) becomes the double pushout diagram of Figure 5.7.(b).

The considered encoding of nets into graph grammars enlightens the dimensions in which graph grammars properly extends nets. First, graph grammars allow for a more structured description of the state, that is a general, possibly non-discrete, graph. Furthermore they allow for productions where the interface graph may not be empty, thus specifying a "context" consisting of items that have to be present for the productions to be applied, but are not affected by the application. We already

observed that, due to their capability of expressing steps which "preserve" part of the state, contextual nets can be seen as an intermediate model between ordinary Petri nets and graph grammars. Indeed, the above encoding of ordinary Petri nets into graph grammars can be straightforwardly extended to P/T contextual nets. They are represented by graph grammars still acting on discrete graphs, but where productions are allowed to have a non-empty interface. Furthermore, we will see in the next chapter that the graphical structure of the state and the dangling condition which prevents the application of a production when it would leave some dangling edge, has a very natural interpretation in the setting of inhibitor nets.

To conclude, let us stress that a gap of abstraction exists between graph grammars and nets. The problem resides in the fact that graphs are more concrete than markings: the nodes of a graph, although carrying the same label, have a precise identity, while tokens in the same place of a net are indistinguishable. Formally, the exact counterpart of a marking (multiset) is an isomorphism class of discrete graphs. Therefore suitable equivalences have to be imposed on graphs and derivations if we want to obtain a precise correspondence with net computations.

5.2 Derivation trace semantics

Historically, the first truly concurrent semantics for graph transformation systems proposed in the literature has been the derivation trace semantics. It is based on the idea of defining suitable equivalences on concrete derivations, equating those derivations which should be considered undistinguishable according to the following two criteria:

- *irrelevance of representation details*, namely of the concrete identity of the items in the graphs involved in a derivation, and
- *true concurrency*, namely irrelevance of the order in which independent productions are applied in a derivation.

The corresponding equivalences, called respectively *abstraction equivalence* and *shift equivalence*, are presented below. Concatenable derivation traces are then defined as equivalence classes of concrete derivations with respect to the least equivalence containing both the abstraction and the shift equivalences. Due to an appropriate choice of the abstraction equivalence, the obvious notion of sequential composition of concrete derivations induces an operation of sequential composition at the abstract level. Thus, as suggested by their name, concatenable derivation traces can be sequentially composed and therefore they can be seen as arrows of a category. Such category, called here the *category of concatenable derivation traces*, coincides with the *abstract truly concurrent model of computation* of a grammar presented in [CMR+97], namely the most abstract model in a hierarchy of models of computation for a graph grammar.

5.2.1 Abstraction equivalence and abstract derivations

Almost invariably, two isomorphic graphs are considered as representing the *same* system state, which is determined only by the topological structure of the graph and by the typing. This is extremely natural in the algebraic approach to graph transformation, where the result of the rewriting procedure is defined in terms of categorical constructions and thus determined only up to isomorphism.[2] A natural solution to reason in terms of *abstract graphs* and *abstract derivations* consists of considering two derivations as equivalent if the corresponding diagrams are isomorphic. Unfortunately, if one wants to have a meaningful notion of sequential composition between abstract derivations this approach does not work. For an extensive treatment of this problem we refer the reader to [CEL+94b, CEL+94a]. Roughly speaking, the difficulty can be described as follows. Two isomorphic graphs, in general, are related by more than one isomorphism, but if we want to concatenate derivations keeping track of the flow of causality we must specify in some way how the items of two isomorphic graphs have to be identified. The problem is treated in [CEL+94b, CEL+94a], which propose a solution based on the choice of a uniquely determined isomorphism, named *standard isomorphism*, relating each pair of isomorphic graphs.

Here we adopt an equivalent, but slightly different solution which is inspired by the theory of Petri nets, and in particular by the notion of concatenable net process [DMM96], and which borrows a technique from [MSW96]. We choose for each class of isomorphic typed graphs a specific graph, called *canonical graph*, and we decorate the source and target graphs of a derivation with a pair of isomorphisms from the corresponding canonical graphs to such graphs. In such a way we are able to distinguish "equivalent"[3] elements in the source and target graphs of derivations and we can safely define their sequential composition. The advantage of our solution is that the notion of equivalence between derivations does not depend on representation details, i.e., on the real identity of the items of the graphs involved in the derivation. For instance, given a derivation, we can change uniformly the name of a node in all the involved graphs and relations, obtaining a new derivation which is equivalent to the original one. We refer to [BCE+99] for a more detailed comparison of the two approaches.

DEFINITION 5.13 (CANONICAL GRAPHS)
We denote by Can a fixed operation that associates to each (TG-typed) graph a so-called canonical graph*, satisfying the following properties:*

1. $Can(G) \simeq G$;
2. *if* $G \simeq G'$ *then* $Can(G) = Can(G')$.

[2] At the concrete level, the fact that the pushout and pushout complement constructions are defined only up to isomorphism generates an undesirable and scarcely intuitive *unbounded nondeterminism* for each production application.

[3] With "equivalent" we mean here two items related by an automorphism of the graph, that are, in absence of further information, indistinguishable.

The construction of the canonical graph can be performed by adapting to our slightly different framework the ideas of [MSW96] and a similar technique can be used to single out a class of standard isomorphisms in the sense of [CEL+94b, CEL+94a]. Working with finite graphs the constructions are effective.

Definition 5.14 (decorated derivation)
A decorated derivation $\psi : G_0 \Rightarrow^* G_n$ *is a triple* $\langle m, \rho, M \rangle$*, where* $\rho : G_0 \Rightarrow^* G_n$ *is a derivation, while* $m : Can(G_0) \to G_0$ *and* $M : Can(G_n) \to G_n$ *are isomorphisms. If* ρ *is an identity derivation then* ψ *is called* discrete*.*

In the following we will denote the components of a decorated derivation ψ by m_ψ, ρ_ψ and M_ψ. For a decorated derivation ψ, we write $\sigma(\psi)$, $\tau(\psi)$, $\#\psi$, $|\psi|$, $prod(\psi)$ to refer to the results of the same operations applied to the underlying derivation ρ_ψ.

Definition 5.15 (sequential composition)
Let ψ *and* ψ' *be two decorated derivations such that* $\tau(\psi) = \sigma(\psi')$ *and* $M_\psi = m_{\psi'}$*. Their* sequential composition*, denoted by* $\psi\,;\,\psi'$*, is defined as follows:*

$$\langle m_\psi, \rho_\psi\,;\,\rho_{\psi'}, M_{\psi'} \rangle.$$

One could have expected sequential composition of decorated derivations ψ and ψ' to be defined whenever $\tau(\psi) \simeq \sigma(\psi')$, regardless of the concrete identity of the items in the two graphs. We decided to adopt a more concrete notion of concatenation since it is technically simpler and it induces, like the more general one, the desired notion of sequential composition at the abstract level.

The abstraction equivalence identifies derivations that differ only for representation details. As announced it is a suitable refinement of the natural notion of diagram isomorphism.

Definition 5.16 (abstraction equivalence)
Let ψ *and* ψ' *be two decorated derivations, with* $\rho_\psi : G_0 \Rightarrow^* G_n$ *and* $\rho_{\psi'} : G'_0 \Rightarrow^* G'_{n'}$ *(whose* i^{th} *steps are depicted in the low rows of Figure 5.8). Suppose that* $q_i = q_{i1} + \ldots + q_{ik_i}$ *for each* $i \in \underline{n}$*, and* $q'_j = q'_{j1} + \ldots + q'_{jk'_j}$ *for each* $j \in \underline{n'}$*. Then they are* abstraction equivalent*, written* $\psi \equiv^{abs} \psi'$*, if*

1. $n = n'$*, i.e., they have the same length;*
2. *for each* $i \in \underline{n}$*,* $k_i = k'_i$ *and for all* $s \in \underline{k_i}$*,* $q_{is} = q'_{is}$*; i.e., the productions applied in parallel at each direct derivation are the same and they are composed in the same order; in particular* $\#\psi = \#\psi'$*;*
3. *there exists a family of isomorphisms*

$$\{\theta_{X_i} : X_i \to X'_i \mid X \in \{L, K, R, G, D\},\ i \in \underline{n}\} \cup \{\theta_{G_0}\}$$

between corresponding graphs appearing in the two derivations such that

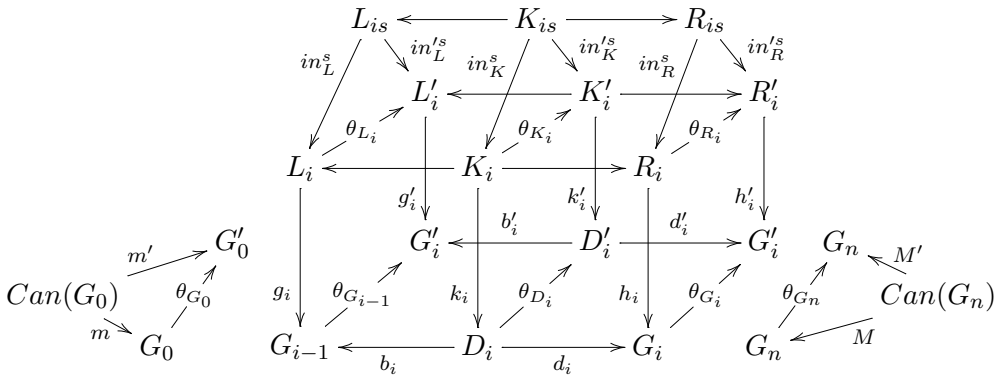


Figure 5.8: Abstraction equivalence of decorated derivations (the arrows in productions spans are not labelled).

(a) the isomorphisms relating the source and target graphs commute with the decorations, i.e., $\theta_{G_0} \circ m = m'$ and $\theta_{G_n} \circ M = M'$;

(b) the resulting diagram commutes (the middle part of Figure 5.8 represents the portion of the diagram relative to step i, indicating only the s^{th} of the k_i productions applied in parallel with the corresponding injections).[4]

Notice that two derivations are abstraction equivalent if, not only they have the same length and apply the same productions in the same order, but also, in a sense, productions are applied to "corresponding" items (condition (3)). In other words abstraction equivalence identifies two decorated derivations if one can be obtained from the other by uniformly renaming the items appearing in the involved graphs. Relation $\equiv^{abs}$ is clearly an equivalence relation. Equivalence classes of decorated derivations with respect to $\equiv^{abs}$ are called *abstract derivations* and are denoted by $[\psi]_{abs}$, where ψ is an element of the class.

It is easy to prove that if $\psi \equiv^{abs} \psi'$ and $\psi_1 \equiv^{abs} \psi'_1$ then, if defined, $\psi;\psi_1 \equiv^{abs} \psi';\psi'_1$. Therefore sequential composition of decorated derivations lifts to composition of abstract derivations.

DEFINITION 5.17 (CATEGORY OF ABSTRACT DERIVATIONS)
The category of abstract derivations *of a grammar $\mathscr{G}$, denoted by* $\mathbf{Abs}[\mathscr{G}]$*, has abstract graphs as objects, and abstract derivations as arrows. In particular, if $\psi : G \Rightarrow^* H$, then $[\psi]_{abs}$ is an arrow from $[G]$ to $[H]$. The identity arrow on $[G]$ is the abs-equivalence class of a discrete derivation $\langle i, G, i\rangle$, where $i : Can(G) \to G$ is any isomorphism, and the composition of arrows $[\psi]_{abs} : [G] \to [H]$ and*

[4]Notice that the isomorphisms θ_{X_i} for $X \in \{L, K, R\}$, relating corresponding parallel productions, are uniquely determined by the properties of coproducts.

$[\psi']_{abs} : [H] \to [X]$ *is defined as* $[\psi\,;\psi'']_{abs} : [G] \to [X]$*, where* $\psi'' \in [\psi']_{abs}$ *is such that the composition is defined.*

It is worth stressing that, whenever $\tau(\psi) \simeq \sigma(\psi')$, we can always rename the items in the graphs of ψ', in order to obtain a derivation ψ'', abs-equivalent to ψ' and composable with ψ, namely such that $\tau(\psi) = \sigma(\psi'')$ and $M_\psi = m_{\psi''}$. Basically, it suffices to substitute in the derivation ψ' each item x in $\sigma(\psi')$ with $M_\psi(m_{\psi'}^{-1}(x))$.

5.2.2 Shift equivalence and derivation traces

From a truly concurrent perspective two derivations should be considered as equivalent when they apply the same productions to the "same" subgraph of a certain graph, even if the order in which the productions are applied may be different. The basic idea of equating derivations which differ only for the order of independent production applications is formalized in the literature through the notion of *shift equivalence* [Kre77, Kre87, Ehr87]. The shift equivalence is based on the possibility of sequentializing a parallel direct derivation (*analysis* construction) and on the inverse construction (*synthesis* construction), which is possible only in the case of *sequential independence*. The union of the shift and abstraction equivalences will yield the *(concatenable) truly concurrent* equivalence, whose equivalence classes are the *(concatenable) derivation traces*.

Let us start by defining the key notion of sequential independence. Intuitively, two consecutive direct derivations $G \Rightarrow_{q'} X$ and $X \Rightarrow_{q''} H$, as in Figure 5.9, are sequentially independent if they may be swapped, i.e., if q'' can be applied to G, and q' to the resulting graph, without changing in an "essential way" the matches. Therefore q'' cannot delete anything that has been explicitly preserved by the application of q' at match g_1 and, moreover, it cannot use (neither consuming nor preserving it) any element generated by q'; this is ensured if the overlapping of R_1 and L_2 in X is included in the intersection of the images of the interface graphs K_1 and K_2 in X.

Definition 5.18 (sequential independence)
Consider a derivation $\delta_1;\delta_2$*, consisting of two direct derivations* $\delta_1 : G \Rightarrow_{q',g_1} X$ *and* $\delta_2 : X \Rightarrow_{q'',g_2} H$ *(as in Figure 5.9). The derivations* δ_1 *and* δ_2 *are called* sequentially independent *if* $g_2(L_2) \cap h_1(R_1) \subseteq g_2(l_2(K_2)) \cap h_1(r_1(K_1))$*; in words, if the images in* X *of the left-hand side of* q'' *and of the right-hand side of* q' *overlap only on items that are preserved by both derivation steps. In categorical terms, this condition can be expressed by requiring the existence of two arrows* $s : L_2 \to D_1$ *and* $u : R_1 \to D_2$ *such that* $d_1 \circ s = g_2$ *and* $b_2 \circ u = h_1$*.*

Notice that, differently from what happens for other formalisms, such as ordinary Petri nets or term rewriting, two rewriting steps δ_1 and δ_2 do not need to be applied at completely disjoint matches to be independent. The graphs to which δ_1 and δ_2 are applied can indeed overlap on something that is preserved by both rewriting

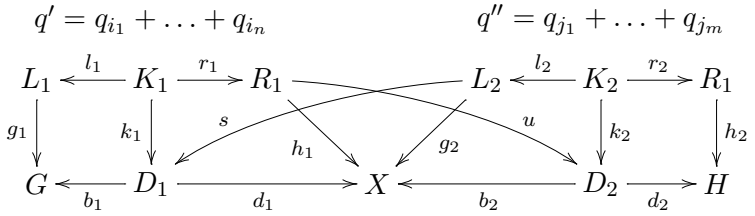


Figure 5.9: Sequential independent derivations.

steps. As in the case of contextual nets, according to the interpretation of preserved items as read-only resources, we can explain this fact by saying that graph rewriting allows for the *concurrent access to read resources*.

EXAMPLE 5.19 (SEQUENTIAL INDEPENDENCE)
Consider the derivation of Figure 5.5. The first two direct derivations are not sequential independent; in fact, the edge $\underline{3} : req$ of graph G_1 is in the image of both the right-hand side of the first production and the left-hand side of the second one, but it is in the context of neither the first nor the second direct derivation. On the contrary, in the same figure, both the derivations from G_1 to G_3 and those from G_2 to G_4 consists of two sequentially independent steps.

□

The next well-known result states that every parallel direct derivation can be sequentialized in an arbitrary way as the sequential application of the component productions, and, conversely, that every pair of sequentially independent direct derivations can be transformed into a parallel direct derivation. This result represents the basis of the theory of concurrency of the double pushout approach to graph rewriting.

THEOREM 5.20 (PARALLELISM THEOREM)
Given (possibly parallel) productions q' and q'', the following statements are equivalent (see Figure 5.10):

1. *There is a parallel direct derivation $G \Rightarrow_{q'+q''} H$*
2. *There are sequentially independent derivations $G \Rightarrow_{q'} H_1 \Rightarrow_{q''} H$.*
3. *There are sequentially independent derivations $G \Rightarrow_{q''} H_2 \Rightarrow_{q'} H$.* □

The proof of the theorem is given by providing two constructions. The first one, called *analysis*, given a parallel direct derivation $G \Rightarrow_{q'+q''} H$, transforms it into a derivation consisting of two steps $G \Rightarrow_{q'} X \Rightarrow_{q''} X$ (with same source and target graphs) which, moreover, is proved to be sequentially independent. The second one, called *synthesis*, applied to any sequentially independent two-step derivation,

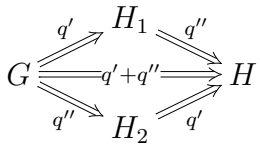


Figure 5.10: Local confluence of independent direct derivations.

transforms it into a parallel direct derivation between the same source and target graphs. These constructions, in general nondeterministic, are used to define suitable relations among derivations (see, e.g., [Kre77, EHKPP91, CMR+97]).

Unlike the original definition of analysis and synthesis, following [CEL+96b], we explicitly keep track of the permutation of the applied productions induced by the constructions. Therefore we first introduce some notation for permutations.

Definition 5.21 (permutation)
A permutation *on the set* $\underline{n} = \{1, 2, \ldots, n\}$ *is a bijective mapping* $\Pi : \underline{n} \to \underline{n}$. *The* identity permutation *on* $\underline{n}$ *is denoted by* Π^n_{id}. *The* composition *of two permutations* Π_1 *and* Π_2 *on* $\underline{n}$, *denoted by* $\Pi_1 \circ \Pi_2$, *is their composition as functions, while the* concatenation *of two permutations* Π_1 *on* $\underline{n_1}$ *and* Π_2 *on* $\underline{n_2}$, *denoted by* $\Pi_1 \mid \Pi_2$, *is the permutation on* $\underline{n_1 + n_2}$ *defined as*

$$\Pi_1 \mid \Pi_2(x) = \begin{cases} \Pi_1(x) & \text{if } 1 \leq x \leq n_1 \\ \Pi_2(x - n_1) + n_1 & \text{if } n_1 < x \leq n_2 \end{cases}$$

Concatenation and composition of permutations are clearly associative.

Proposition 5.22 (analysis and synthesis)
Let $\rho : G \Rightarrow_q H$ *be a parallel direct derivation using* $q = q_1 + \ldots + q_k : (L \stackrel{l}{\leftarrow} K \stackrel{r}{\rightarrow} R)$. *Then for each partition* $\langle I = \{i_1, \ldots, i_n\}, J = \{j_1, \ldots, j_m\}\rangle$ *of* $\underline{k}$ *(i.e.,* $I \cup J = \underline{k}$ *and* $I \cap J = \emptyset$*) there is a constructive way—in general nondeterministic—to obtain a sequential independent derivation* $\rho' : G \Rightarrow_{q'} X \Rightarrow_{q''} H$, *called an* analysis *of* ρ, *where* $q' = q_{i_1} + \ldots + q_{i_n}$, *and* $q'' = q_{j_1} + \ldots + q_{j_m}$ *(see Figure 5.9). If* ρ *and* ρ' *are as above, we shall write* $\rho \equiv^{an}_{\Pi} \rho'$, *where* Π *is the permutation on* $\underline{k}$ *defined as* $\Pi(i_x) = x$ *for* $x \in \underline{n}$, *and* $\Pi(j_x) = n + x$ *for* $x \in \underline{m}$.

Conversely, let $\rho : G \Rightarrow_{q'} X \Rightarrow_{q''} H$ *be a pair of sequentially independent derivations. Then there is a constructive way to obtain a parallel direct derivation* $\rho' = G \Rightarrow_{q'+q''} H$, *called a* synthesis *of* ρ. *In this case, we shall write* $\rho \equiv^{syn}_{\Pi} \rho'$, *where* $\Pi = \Pi^{\#\rho}_{id}$. □

Informally, two derivations are shift equivalent if one can be obtained from the other by repeatedly applying the analysis and synthesis constructions. The next definition emphasizes the fact that the sets of productions applied in two shift equivalent

derivations are related by a permutation which is constructed inductively starting from the permutations introduced by analysis and synthesis.

DEFINITION 5.23 (SHIFT EQUIVALENCE)
Derivations ρ and ρ' are shift equivalent via permutation Π, written $\rho \equiv^{sh}_{\Pi} \rho'$, if this can be deduced by the following inference rules:

$$(\mathsf{SH{-}id})\ \frac{}{\rho \equiv^{sh}_{\Pi^{\#\rho}_{id}} \rho} \qquad (\mathsf{SH{-}\emptyset})\ \frac{d \circ b^{-1} = id_G}{G \equiv^{sh}_{\emptyset} G \overset{\langle \emptyset,\emptyset,\emptyset,b,d\rangle}{\Rightarrow_{\emptyset}} G} \qquad (\mathsf{SH{-}an})\ \frac{\rho \equiv^{an}_{\Pi} \rho'}{\rho \equiv^{sh}_{\Pi} \rho'}$$

$$(\mathsf{SH{-}syn})\ \frac{\rho \equiv^{syn}_{\Pi} \rho'}{\rho \equiv^{sh}_{\Pi} \rho'} \qquad (\mathsf{SH{-}sym})\ \frac{\rho \equiv^{sh}_{\Pi} \rho'}{\rho' \equiv^{sh}_{\Pi^{-1}} \rho} \qquad (\mathsf{SH{-}trans})\ \frac{\rho \equiv^{sh}_{\Pi} \rho',\ \rho' \equiv^{sh}_{\Pi'} \rho''}{\rho \equiv^{sh}_{\Pi' \circ \Pi} \rho''}$$

$$(\mathsf{SH{-}comp})\ \frac{\rho_1 \equiv^{sh}_{\Pi_1} \rho'_1,\ \rho_2 \equiv^{sh}_{\Pi_2} \rho'_2,\ \tau(\rho_1) = \sigma(\rho_2)}{\rho_1 \,;\, \rho_2 \equiv^{sh}_{\Pi_1 | \Pi_2} \rho'_1 \,;\, \rho'_2}$$

Note that by $(\mathsf{SH} - \emptyset)$ an empty direct derivation is shift equivalent to the identity derivation G if and only if the induced isomorphism is the identity. The shift equivalence *is the equivalence relation $\equiv^{sh}$ defined as $\rho \equiv^{sh} \rho'$ iff $\rho \equiv^{sh}_{\Pi} \rho'$ for some permutation Π.*

It is worth stressing that the shift equivalence abstracts both from the order in which productions are composed inside a single direct parallel step and from the order in which independent productions are applied at *different* direct derivations.

EXAMPLE 5.24 (SHIFT EQUIVALENCE)
Figure 5.11 shows a derivation ρ' which is shift equivalent to derivation ρ of Figure 5.6. It is obtained by applying the synthesis construction to the sub-derivation of ρ from G_1 to G_3. □

Despite the unusual definition, it is easy to check that the shift equivalence just introduced is the same as in [Kre77, Ehr87, CEL+94a]. From the definitions of the shift equivalence and of the analysis and synthesis constructions, it follows that $\rho \equiv^{sh} \rho'$ implies that ρ and ρ' have the same order and the same source and target graphs (i.e., $\#\rho = \#\rho'$, $\sigma(\rho) = \sigma(\rho')$, and $\tau(\rho) = \tau(\rho')$; by the way, this guarantees that rules $(\mathsf{SH} - \mathsf{comp})$ and $(\mathsf{SH} - \mathsf{trans})$ are well-defined. The shift equivalence can be extended in a natural way to decorated derivations.

DEFINITION 5.25
The shift equivalence *on decorated derivations, denoted with the same symbol $\equiv^{sh}$, is defined by $\langle m, \rho, M\rangle \equiv^{sh} \langle m, \rho', M\rangle$ if $\rho \equiv^{sh} \rho'$.*

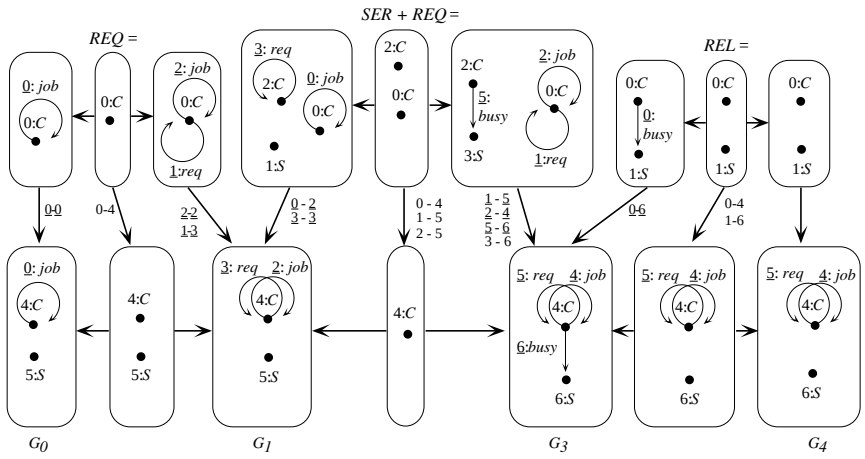


Figure 5.11: A derivation ρ' in grammar $\mathscr{C}$-$\mathscr{S}$, shift-equivalent to derivation ρ of Figure 5.6.

Equivalence $\equiv^{sh}$ does not subsume abstraction equivalence, since, for example, it cannot relate derivations starting from different but isomorphic graphs.

We introduce a further equivalence on decorated derivations, obtained simply as the union of $\equiv^{abs}$ and $\equiv^{sh}$. It is called *truly-concurrent* (or *tc-*) equivalence, since it equates all derivations which are not distinguishable from a true concurrency perspective, at an adequate degree of abstraction from representation details. A small variation of this equivalence is introduced as well, called *ctc-equivalence*, where the first "c" stays for "concatenable". Equivalence classes of (c)tc-equivalent decorated derivations are called *(concatenable) derivation traces*.

DEFINITION 5.26 (TRULY-CONCURRENT EQUIVALENCES AND TRACES)
Two decorated derivations ψ and ψ' are ctc-equivalent via permutation Π, written $\psi \equiv^c_\Pi \psi'$, if this can be deduced by the following inference rules:

$$(\mathsf{CTC{-}abs})\ \frac{\psi \equiv^{abs} \psi'}{\psi \equiv^c_{\Pi^{\#\psi}_{id}} \psi'} \qquad (\mathsf{CTC{-}sh})\ \frac{\psi \equiv^{sh}_\Pi \psi'}{\psi \equiv^c_\Pi \psi'} \qquad (\mathsf{CTC{-}trans})\ \frac{\psi \equiv^c_\Pi \psi' \quad \psi' \equiv^c_{\Pi'} \psi''}{\psi \equiv^c_{\Pi' \circ \Pi} \psi''}$$

Equivalence $\equiv^c$, defined as $\psi \equiv^c \psi'$ iff $\psi \equiv^c_\Pi \psi'$ for some permutation Π, is called the concatenable truly concurrent (ctc-) equivalence. *Equivalence classes of derivations with respect to $\equiv^c$ are denoted as $[\psi]_c$ and are called* concatenable derivation traces. *A* derivation trace *is an equivalence class of derivations with respect to the* truly-concurrent (tc-) equivalence $\equiv$ *defined by the following rules:*

$$(\mathsf{TC{-}ctc})\ \frac{\psi \equiv^c_\Pi \psi'}{\psi \equiv_\Pi \psi'} \qquad (\mathsf{TC{-}iso})\ \frac{\psi \equiv_\Pi \psi' \quad \alpha \text{ discrete decor. deriv. s.t. } \psi'\,;\alpha \text{ is defined}}{\psi \equiv_\Pi \psi'\,;\alpha}$$

Equivalently, the tc-equivalence could have been defined as $\psi \equiv_\Pi \psi'$ if and only if $\psi \equiv^c_\Pi \langle m_{\psi'}, \rho_{\psi'}, M' \rangle$, for some isomorphism $M' : Can(\tau(\psi')) \to \tau(\psi')$. Informally,

differently from ctc-equivalence, tc-equivalence does not take into account the decorations of the target graphs of derivations, that is their ending interface, and this is the reason why derivation traces are not concatenable.

Concatenable derivation traces, instead, are naturally equipped with an operation of sequential composition, inherited from concrete decorated derivations, which allows us to see them as arrows of a category having abstract graphs as objects. Such category is called the category of concatenable derivation traces (or the abstract truly concurrent model of computation) of the grammar.

DEFINITION 5.27 (CATEGORY OF CONCATENABLE DERIVATION TRACES)
The category of concatenable derivation traces *of a grammar $\mathscr{G}$, denoted by* $\mathbf{Tr}[\mathscr{G}]$*, is the category having abstract graphs as objects, and concatenable derivation traces as arrows. In particular, if $\psi : G \Rightarrow^*_{\mathscr{G}} H$ then $[\psi]_c$ is an arrow from $[G]$ to $[H]$. The identity arrow on $[G]$ is the ctc-equivalence class of a discrete derivation $\langle i, G, i\rangle$, where i is any isomorphism from $Can(G)$ to G. The composition of arrows $[\psi]_c : [G] \to [H]$ and $[\psi']_c : [H] \to [X]$ is defined as $[\psi\,;\,\psi'']_c : [G] \to [X]$, where $\psi'' \in [\psi']_c$ is a decorated derivation such that $\psi\,;\,\psi''$ is defined.*

Category $\mathbf{Tr}[\mathscr{G}]$ is well-defined because so is the sequential composition of arrows: in fact, if $\psi_1 \equiv^c \psi_2$ and $\psi'_1 \equiv^c \psi'_2$ then (if defined) $\psi_1\,;\,\psi'_1 \equiv^c \psi_2\,;\,\psi'_2$ (hence the attribution "concatenable"). Moreover, for abstract derivations, whenever $\tau(\psi) \simeq \sigma(\psi')$, it is always possible to concatenate the corresponding traces, namely one can always find a derivation $\psi'' \in [\psi']_c$ such that $\psi\,;\,\psi''$ is defined.

5.3 A category of typed graph grammars

Various notions of morphism for graph grammars have been introduced in the literature (see, e.g., [CEL+96a, HCEL96, Rib96, BC96]), most of them influenced by Winskel's notion of Petri net morphism [Win87a, Win87b] through the close relationship existing between typed graph grammars and P/T Petri nets.

Relying on the proposals in [CEL+96a, BC96], this section defines the category of graph grammars which will be used in the thesis. We explain how graph grammars morphisms arise as a generalization of Petri net morphisms, and we show that, as for Petri nets, graph grammars morphisms preserve the behaviour, namely they allow to "translate" each derivation of the source grammar into a derivation of the target grammar.

5.3.1 From multirelations to spans

Multisets and multirelations play an essential role in the definition of morphisms of (generalized) Petri nets. We next provide a categorical view of such concepts, by showing that a tight relationship exists between the category of multirelations and the category of spans over **Set**. These considerations will be helpful to understand

the notion of grammar morphism as a generalization of Petri net morphisms. The material is elaborated partly from [CEL+96a], where the first notion of grammar morphism have been introduced and partly from [BC96, BG00].

The category of spans

We start by reviewing the definition of the category of *semi-abstract spans* over a given category [BC96].

DEFINITION 5.28 (SPAN)
Let $\mathbf{C}$ *be a category. A* (concrete) span *in* $\mathbf{C}$ *is a pair of coinitial arrows* $f = \langle f^L, f^R \rangle$ *with* $f^L : x_f \to a$ *and* $f^R : x_f \to b$. *Objects* a *and* b *are called the* source *an the* target *of the span and we write* $f : a \leftrightarrow b$. *The span* f *will be sometimes written as* $\langle f^L, x_f, f^R \rangle$, *explicitly giving the object* x_f.

Consider the relation $\sim$ over the set of spans of a category, defined as follows: given two spans with the same source and target $f, f' : a \leftrightarrow b$, put $f \sim f'$ if there exists an isomorphism $k : x_f \to x_{f'}$ such that $f'^L \circ k = f^L$ and $f'^R \circ k = f^R$ (see Figure 5.12.(a)). It is immediate to see that $\sim$ is an equivalence relation, which intuitively abstracts out from the particular choice of the object x_f in a concrete span f. The $\sim$-equivalence class of a concrete span f will be denoted by $[f]$ and called a *semi-abstract span.*

DEFINITION 5.29 (COMPOSITION OF SPANS)
Let f_1 *and* f_2 *be spans in a category* $\mathbf{C}$. *A* composition *of* f_1 *and* f_2, *denoted by* $f_1; f_2$ *is a span* f *constructed as in Figure 5.12.(b) (i.e.,* $f^L = f_1^L \circ y$ *and* $f^R = f_2^R \circ z$), *where the square is a pullback.*

The composition of spans is defined only up to equivalence, since the pullback, if it exists, is unique only up to isomorphism. To obtain a categorical structure we must work with semi-abstract spans, in a category with pullbacks.

DEFINITION 5.30 (CATEGORY OF SPANS)
Let $\mathbf{C}$ *be a category with pullbacks. Then the category* $\mathbf{Span}(\mathbf{C})$ *has the same objects of* $\mathbf{C}$ *and semi-abstract spans on* $\mathbf{C}$ *as arrows. More precisely, a semi-abstract span* $[f]$ *is an arrow from the source to the target of* f. *The composition of two semi-abstract spans* $[f_1] : a \leftrightarrow b$ *and* $[f_2] : b \leftrightarrow c$ *is defined as* $[f_1; f_2]$, *where* $f_1; f_2$ *denotes any composition of the concrete spans* f_1 *and* f_2. *The identity on an object* a *is the equivalence class of the span* $\langle id_a, id_a \rangle$, *where* id_a *is the identity of* a *in* $\mathbf{C}$.

By exploiting the properties of pullbacks it can be shown that $\mathbf{Span}(\mathbf{C})$ is a well-defined category, namely that composition is associative and that identities behave correctly.

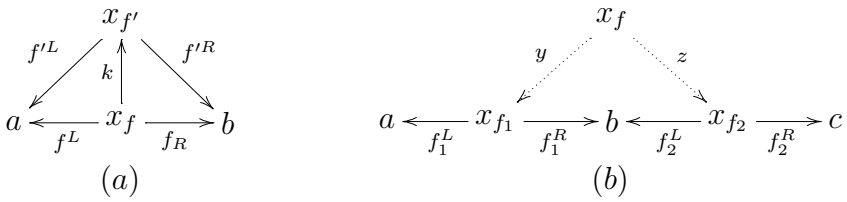


Figure 5.12: Equivalence and composition of spans.

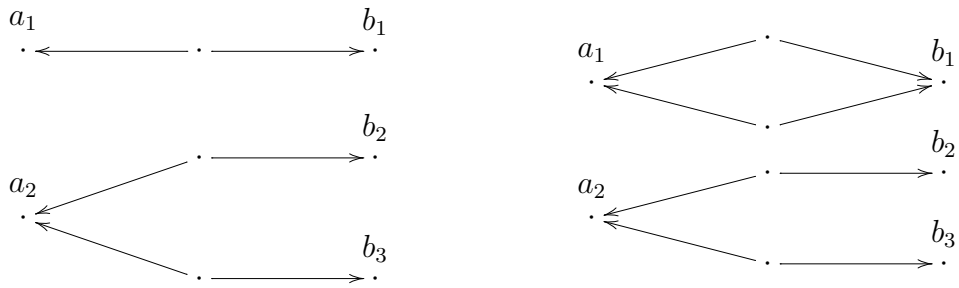


Figure 5.13: The spans f_1 (left diagram) and f_2 (right diagram) in **Set**.

Multirelations and Span(Set)

Let us restrict our attention to the category **Span**(**Set**) of spans over the category **Set** of sets and total functions. We will see that a deep connection exists between **Span**(**Set**) and the category **MSet** of sets and (finitary) multirelations.

Given two sets A and B, consider a semi-abstract span $[f] : A \leftrightarrow B$. We can give a graphical representation of a span by tracing an arrow between the elements $x \in X_f$ and $a \in A$ whenever $f^L(x) = a$ and, similarly, between $x \in X_f$ and $b \in B$ whenever $f^R(x) = b$. The left diagram in Figure 5.13 represents the span $[f_1] : A \leftrightarrow B$, where $A = \{a_1, a_2\}$, $B = \{b_1, b_2, b_3\}$, $X_f = \{x_1, x_2, x_3\}$ and $f_1^R = \{(x_1, a_1), (x_2, a_2), (x_3, a_2)\}$, $f_1^L = \{(x_1, b_1), (x_2, b_2), (x_3, b_3)\}$. The fact that the set X_f is fixed only up to isomorphism means that we can disregard the concrete identities of its elements.

Intuitively the span is completely characterized by the way in which elements of A are connected to elements of B. Furthermore observe that not only the existence of a path between two elements a and b is important, but also the number of such paths. For instance the spans f_1 and f_2 of Figure 5.13 are not equivalent since in f_1 the element a_1 is connected to b_1 via just one path, while in f_2 there are two paths.

The above discussion suggests that semi-abstract spans in **Set** can be seen as an alternative presentation of multirelations, in the sense that one can think of $[f] : A \leftrightarrow B$ as specifying for all pairs of elements $a \in A$ and $b \in B$, "how many times they are related". For example, the span $[f_1]$ can be identified with the multirelation

$(a_1, b_1) + (a_2, b_2) + (a_2, b_3)$, while $[f_2]$ can be seen as a representation of $2 \cdot (a_1, b_1) + (a_2, b_2) + (a_2, b_3)$.

This intuitive correspondence can be made very formal. The category **MSet** of sets and multirelations can be embedded into **Span**(**Set**). The embedding functor is the identity on objects, and maps any multirelation $F : A \to B$, which is a function $F : A \times B \to \mathbb{N}$, to the span $[\hat{F}]$ with

$$X_{\hat{F}} = \{((a, b), n) \mid (a, b) \in A \times B \ \wedge \ n < F(a, b)\}$$

and $\hat{F}^R((a, b), n)) = a$, $\hat{F}^L((a, b), n)) = b$, for all $((a, b), n) \in X_{\hat{F}}$.

PROPOSITION 5.31
There is an embedding of the category **MSet** *of (finitary) multirelations into the category* **Span**(**Set**) *of semi-abstract spans over* **Set**.

The two categories are not isomorphic essentially because **Span**(**Set**) extends **MSet** by adding also multirelations which are not finitary. Say that a span $[f] : A \leftrightarrow B$ is *finitary* if each element of A has a finite counterimage in X_f, namely $f^{R^{-1}}(a)$ is finite for all $a \in A$. Then it is possible to prove that the functor described above restricts to an isomorphism between **FSpan**(**Set**), the lluf subcategory (Definition A.4) of **Span**(**Set**) having finitary spans as arrows, and **MSet** [BG00].

Multisets and multirelation application

Let **1** denote the initial object in **Set**, namely the set with just one element. Observe that a multiset M of a set A can be seen as a multirelation of $\mathbf{1} \times A$, and therefore it can be represented as a semi-abstract span $[\hat{M}] : \mathbf{1} \leftrightarrow A$. Equivalently, since there is a unique arrow from $X_{\hat{M}}$ to **1**, the multiset can be identified with the isomorphism class of the mapping $\hat{M}^L : X_{\hat{M}} \to A$ (in the comma category $\langle \mathbf{Set} \downarrow A \rangle$). This observation confirms that (abstract) typed graphs are the natural generalization of multisets when moving from sets to graphs.

Furthermore, given a multirelation $F : A \to B$ and a multiset M of A, it is immediate to verify that the image of M through F, namely $\mu F(M)$ can be computed simply as the composition of the corresponding spans, i.e. as $\hat{M}; \hat{F}$, as depicted in the left part of Figure 5.14, where the square is a pullback. The arrow towards the initial object is dotted to stress that we can think of the construction as working on typed sets. Referring to the figure, we can say that the construction maps the "A-typed set" $\hat{M}^R : X_{\hat{M}} \to A$ to the the "B-typed set" $z; \hat{F}^R : X_{\hat{M}} \to A$

As an example, the right part of Figure 5.14 shows how the construction can be used to apply the multirelation $F_1 = 2 \cdot (a_1, b_1) + (a_2, b_2) + (a_2, b_3)$, represented by the span f_2 of Figure 5.13, to the multiset $a_1 + a_2$. The result is correctly $2 \cdot b_1 + b_2 + b_3$.

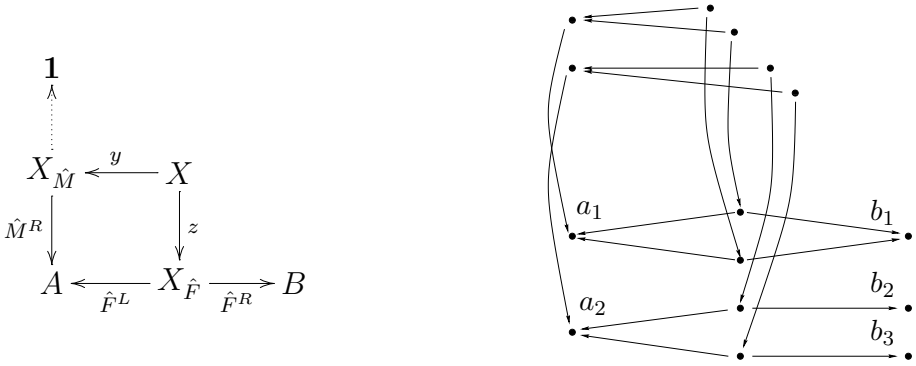


Figure 5.14: Computing the image of a multiset.

Relations and functions

Relations can be identified with special multirelations $R : A \to B$ where multiplicities are bounded by one (namely $R(a, b) \leq 1$ for all $a \in A$ and $b \in B$). The corresponding condition on a span $f : A \leftrightarrow B$ is the existence of at most one path between any two elements $a \in A$ and $b \in B$. The next definition provides a categorical formalization of this concept.

DEFINITION 5.32
Let $\mathbf{C}$ *be a category. A span* $f : a \leftrightarrow b$ *in* $\mathbf{C}$ *is called* relational *if* $\langle f^L, f^R \rangle : x_f \to a \times b$ *is mono.*

In other words $f : a \leftrightarrow b$ is relational if given any object c and pair of arrows $g, h : c \to x_f$, if $g; f^R = h; f^R$ and $g; f^L = h; f^L$ then $g = h$. It is easy to verify that if $f : A \leftrightarrow B$ is a span in **Set** then the above condition can be equivalently expressed as

$$\forall x, y \in X_f.\ f^R(x) \neq f^R(y) \ \vee \ f^L(x) \neq f^L(y)$$

which indeed corresponds exactly to the intuition of having at most one path between any two elements. For example, the span f_1 of Figure 5.13 is relational, while f_2 is not.

Observe that, in particular, a span $f : A \leftrightarrow B$ is relational when either its left or right component is injective. It is easy to realize that these kinds of span corresponds to the partial functions from A to B and backward. In fact, a partial function $g : A \to B$ can be identified with the span

$$A \hookleftarrow dom(g) \xrightarrow{g} B$$

and similarly a partial function $h : B \to A$ can be represented as the span

$$A \xleftarrow{h} dom(h) \hookrightarrow B$$

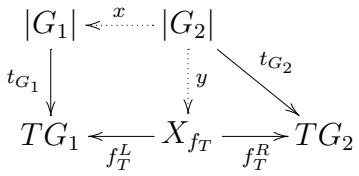
where unlabelled arrows are inclusions.

5.3.2 Graph grammar morphisms

Recall that a Petri net morphism [Win87a, Win87b] consists of two components, namely a multirelation between the sets of places and a partial function mapping transitions of the first net into transitions of the second one. Net morphisms are required to "preserve" the pre-set and post-set of transitions, in the sense that the pre- (post-)set of the image of a transition t must be the image of the pre- (post-)set of t.

Since the items of the graph of types of a grammar can be seen as a generalization of Petri net places, the first component of a grammar morphism from $\mathscr{G}_1$ to $\mathscr{G}_2$ will be a semi-abstract span between the type graphs of $\mathscr{G}_1$ and $\mathscr{G}_2$, which, as explained in the previous subsection, generalizes the notion of multirelation. The idea of using graph grammar morphisms based on spans has been first introduced in [CEL$^+$96a], while the introduction of semi-abstract spans is due to [BC96]. Different notions of morphism for graph grammars use a (partial) function from the type graph of $\mathscr{G}_1$ to the type graph of $\mathscr{G}_2$ [HCEL96] or in the converse direction [Rib96]. Since partial functions can be represented as (relational) spans, these notions can be seen as instances of the more general definition based on spans (with the advantage of allowing a simpler composition, not requiring the use of pullbacks).

Let $\mathscr{G}_1$ and $\mathscr{G}_2$ be two graph grammars and let $[f_T] : TG_1 \leftrightarrow TG_2$ be a semi-abstract span between the corresponding type graphs, namely an arrow in **Span**(**Graph**). By extending to graphs the construction presented in the previous section for sets (see Figure 5.14), $[f_T]$ allows us to relate TG_1-typed graphs to TG_2-typed graphs. Let G_1 be in TG_1-**Graph**. The graph G_1 is transformed, as depicted in the diagram below, by first taking a pullback (in **Graph**) of the arrows $f_T^L : X_{f_T} \to TG_1$ and $t_{G_1} : |G_1| \to TG_1$, and then typing the pullback object over TG_2 by using the right part of the span $f_T^R : X_{f_T} \to TG_2$.



The TG_2-typed graph $G_2 = \langle |G_2|, f_T^R \circ y \rangle$ obtained with this construction, later referred to as *pullback-retyping* construction induced by $[f_T]$, is determined only up to isomorphism. Sometimes we will write $f_T\{x, y\}(G_1, G_2)$ (or simply $f_T(G_1, G_2)$ if we are not interested in morphisms x and y) to express the fact that G_1 and G_2 are related in this way by the pullback-retyping construction induced by $[f_T]$.

We are now ready to define grammar morphisms. Besides the component specifying the relation between the type graphs, a morphism from $\mathscr{G}_1$ to $\mathscr{G}_2$ contains a (partial) mapping between production names. Furthermore a third component explicitly relates the (untyped) graphs underlying corresponding productions of the two grammars, as well as the graphs underlying the start graphs.

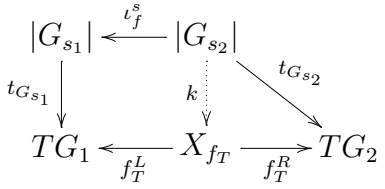
DEFINITION 5.33 (TYPED GRAPH GRAMMAR MORPHISM)
Let $\mathscr{G}_i = \langle TG_i, G_{s_i}, P_i, \pi_i \rangle$ $(i \in \{1,2\})$ be two graph grammars. A (typed graph grammar) morphism $f : \mathscr{G}_1 \to \mathscr{G}_2$ is a triple $\langle [f_T], f_P, \iota_f \rangle$ where

- *$[f_T] : TG_1 \to TG_2$ is a semi-abstract span in* **Graph**, *called the* type-span*;*
- *$f_P : P_1 \to P_2 \cup \{\emptyset\}$ is a total function, where $\emptyset$ is a new production name (not in P_2), with associated production $\emptyset \leftarrow \emptyset \rightarrow \emptyset$, referred to as the* empty production*;*
- *ι_f is a family $\{\iota_f(q_1) \mid q_1 \in P_1\} \cup \{\iota_f^s\}$ such that $\iota_f^s : |G_{s_2}| \to |G_{s_1}|$ and for each $q_1 \in P_1$, if $f_P(q_1) = q_2$, then $\iota_f(q_1)$ is a triple of morphisms*

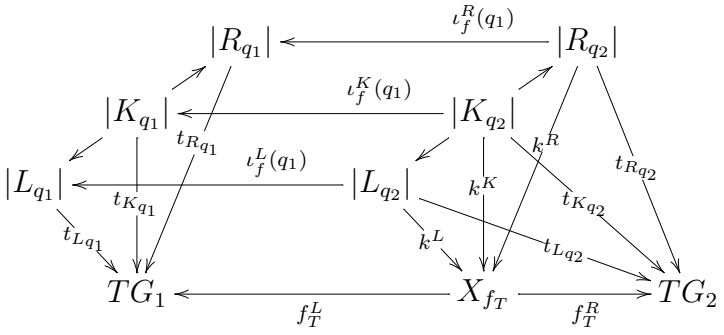
$$\langle \iota_f^L(q_1) : |L_{q_2}| \to |L_{q_1}|, \iota_f^K(q_1) : |K_{q_2}| \to |K_{q_1}|, \iota_f^R(q_1) : |R_{q_2}| \to |R_{q_1}| \rangle.$$

such that the following conditions are satisfied:

1. Preservation of the start graph.
   There exists a morphism k such that $f_T\{\iota_f^s, k\}(G_{s_1}, G_{s_2})$, namely such that the following diagram commutes (where the square is required to be a pullback).



2. Preservation of productions.
   For each $q_1 \in P_1$, with $q_2 = f_P(q_1)$, there exist morphisms k^L, k^K and k^R such that the diagram below commutes, and $f_T\{\iota_f^X(q_1), k^X\}(X_{q_1}, X_{q_2})$ for $X \in \{L, K, R\}$.



The grammar morphisms in [CEL+96a] rely on the assumption of having a fixed choice of pullbacks. Consequently the pullback-retyping construction is deterministic and morphisms are required to preserve the start graphs and the productions "on the nose", namely the construction applied to the start graph of $\mathscr{G}_1$ must result exactly in the start graph of $\mathscr{G}_2$. Similarly, each production in $\mathscr{G}_1$ must be mapped exactly to the corresponding production in $\mathscr{G}_2$. Notice that this requirement is very strict and it may imply the absence of a morphism between two grammars having start graph and productions which are the same up to isomorphism. Our notion of morphism is, in a sense, more liberal: we avoid a global choice of pullbacks and we fix "locally", for each morphism f, only part of the pullback diagrams, namely the morphisms in the family ι_f. The presence of such component in the morphism is intuitively motivated by the fact that graph grammars are more concrete than Petri nets. In fact, in a graph grammar the start graph and the productions are specified by means of *concrete* graphs, while the initial marking and the transitions of a Petri net are defined in terms of multisets which corresponds to abstract (discrete) graphs. Therefore when a grammar $\mathscr{G}_1$ simulates another grammar $\mathscr{G}_2$ it is important to specify not only that a production q_1 of $\mathscr{G}_1$ is mapped to a production q_2 of $\mathscr{G}_2$, but also the correspondence between the concrete items of the graphs in the two productions q_1 and q_2.

It is worth noticing that, for technical convenience, the partial mapping on production names is represented as a total mapping by enriching the target set with a distinguished point $\emptyset$, representing "undefinedness". In this way the condition asking the preservation of productions (condition (2)) faithfully rephrases the situation of net theory where the pre- and post-set of a transition on which the morphism is undefined are necessarily mapped to the empty multiset.

DEFINITION 5.34 (CATEGORY **GG**)
Graph grammars and graph grammar morphisms form a category **GG**.

The identities and the composition of morphisms in **GG** are defined in the obvious way. Given a grammar $\mathscr{G}$, the identity on $\mathscr{G}$ is the grammar morphism $\langle [id_{TG}], id_P, \iota \rangle$, where $[id_{TG}]$ is the identity span on TG, id_P is the identity function on P and all the components of ι are identities on the corresponding graphs.

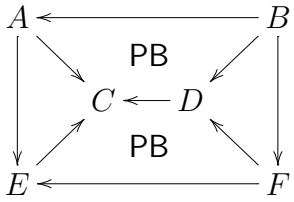
Given two morphisms $f_0 : \mathscr{G}_0 \to \mathscr{G}_1$ and $f_1 : \mathscr{G}_1 \to \mathscr{G}_2$, the composition $f_1 \circ f_0 : \mathscr{G}_0 \to \mathscr{G}_2$ is the morphism $\langle [f_{1T}] \circ [f_{0T}], f_{1P} \circ f_{0P}, \iota \rangle$, where for any $q_0 \in P_0$ and any $X \in \{L, K, R\}$, we have $\iota^X(q_0) = \iota^X_{f_0}(q_0) \circ \iota^X_{f_1}(f_{0P}(q_0))$

5.3.3 Preservation of the behaviour

As in [CEL+96a] it is possible to show that morphisms preserve the behaviour of graph grammars, in the sense that given a morphism $f : \mathscr{G}_1 \to \mathscr{G}_2$, for every derivation ρ_1 in $\mathscr{G}_1$ there is a corresponding derivation ρ_2 in $\mathscr{G}_2$, related to ρ_1 by the pullback-retyping construction induced by the type component $[f_T]$ of the morphism.

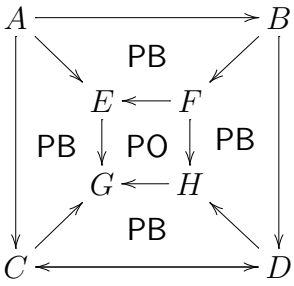
First, we need to review some results expressing properties of pullbacks and pushouts in **Graph**. The properties are stated for the category **Set** of sets and total functions, since the constructions of pullback and pushout can be lifted from **Set** to **Graph**, where limits and colimits are constructed "componentwise". A proof can be found in [CEL+96a].

PROPOSITION 5.35
Consider the following commuting diagram in **Set**



If the two internal squares, marked by PB *are pullbacks, then the outer square with vertices A, B, E and F is a pullback as well.*

PROPOSITION 5.36 (3-CUBE LEMMA)
Consider the following commuting diagram in **Set**



If the small internal squares, marked by PB *and* PO *are pullbacks and pushouts, respectively, then the outer square with vertices A, B, C and D is a pushout.*

We are now ready to prove that graph grammar morphisms preserve derivations. As a consequence of the partial arbitrariness in the choice of the pullback components, such correspondence is not "functional", but it establishes just a relation between concrete derivations of the source and target grammars of the morphism.

LEMMA 5.37
Let $f : \mathscr{G}_1 \to \mathscr{G}_2$ be a graph grammar morphism, and let $\delta_1 : G_1 \Rightarrow^*_{q_1} H_1$ be a direct derivation in $\mathscr{G}_1$. Then there exists a corresponding direct derivation $\delta_2 : G_2 \Rightarrow^*_{f_P(q_1)} H_2$ in $\mathscr{G}_2$, such that $f_T(G_1, G_2)$ and $f_T(H_1, H_2)$.

PROOF. The proof is extremely technical and follows the same outline as in [CEL+96a]. Here we give only a sketch, singling out some relevant passages.

Assume that the derivation δ_1 in $\mathscr{G}_1$ has the following shape:

$$q_1: \quad \begin{array}{ccccc} L_1 & \longleftarrow & K_1 & \longrightarrow & R_1 \\ {\scriptstyle g_1}\downarrow & & \downarrow{\scriptstyle k_1} & & \downarrow{\scriptstyle h_1} \\ G_1 & \xleftarrow[b_1]{} & D_1 & \xrightarrow[d_1]{} & H_1 \end{array}$$

The same derivation is depicted also in the left part of Figure 5.15. The right part of the top layer represents the production $q_2 = f_P(q_1)$. Observe that the figure is not complete. First, for each graph appearing in the top layer we should have indicated the corresponding typing morphism. Furthermore, by definition of grammar morphism, the two productions q_1 and q_2 are related by a pullback-retyping construction as expressed by condition (3) in Definition 5.33, and thus a morphism $\kappa^Y : |Y_2| \to X$ should appear for $Y \in \{L, K, R\}$. For the sake of clearness only the typing morphisms of L_1 and and the morphism $k^L : |L_2| \to X$ of the pullback-retyping construction are explicitly represented.

Consider any three graphs G_2, D_2 and H_2, obtained from G_2, D_2 and H_2, respectively, by applying the pullback-retyping construction. Such graphs, with the corresponding pullback-retyping diagrams are represented in the bottom part of Figure 5.15. Now, recall that the square with vertices $|G_1|$, $|G_2|$, TG_1, X is a pullback, and the square with vertices $|L_1|$, $|L_2|$, TG_1, X commutes. Hence the match $g_2 : |L_2| \to |G_2|$ is uniquely determined by the universal property of pullbacks. Similarly, observing that the square $|D_1|$, $|D_2|$, TG_1, X, commutes we uniquely determine a morphism $b_2 : |D_2| \to |G_2|$. With an analogous reasoning we can complete the whole diagram of Figure 5.15.

By using Proposition 5.35 it is not difficult to realize that all the squares with vertices

- $|L_1|$, $|L_2|$, $|G_1|$, $|G_2|$
- $|G_1|$, $|G_2|$, $|D_1|$, $|D_2|$
- $|D_1|$, $|D_2|$, $|K_1|$, $|K_2|$
- $|K_1|$, $|K_2|$, $|L_1|$, $|L_2|$

are pullbacks, while the square $|K_1|$, $|L_1|$, $|G_1|$, $|D_1|$ is a pushout, by construction. By the 3-cube lemma (Proposition 5.36) we deduce that the square $|K_2|$, $|L_2|$, $|G_2|$, $|D_2|$ is a pushout. Since, by symmetry also $|K_2|$, $|D_2|$, $|H_2|$, $|R_2|$ is a pushout, this concludes the construction of the desired direct derivation δ_2.

□

The result can straightforwardly be extended to general derivations involving an arbitrary number of (possibly parallel) direct derivation.

Recall that in [CEL+96a], where the more concrete notion of morphism based on a choice of pullbacks is adopted, the relation between derivations induced by a morphism is indeed a function. Furthermore the result above is extended to show that the concrete model of computation of the grammar (not considering abstraction equivalence) can be obtained via a functorial construction, establishing an adjunction between the category of graph grammars and the category of concrete derivations.

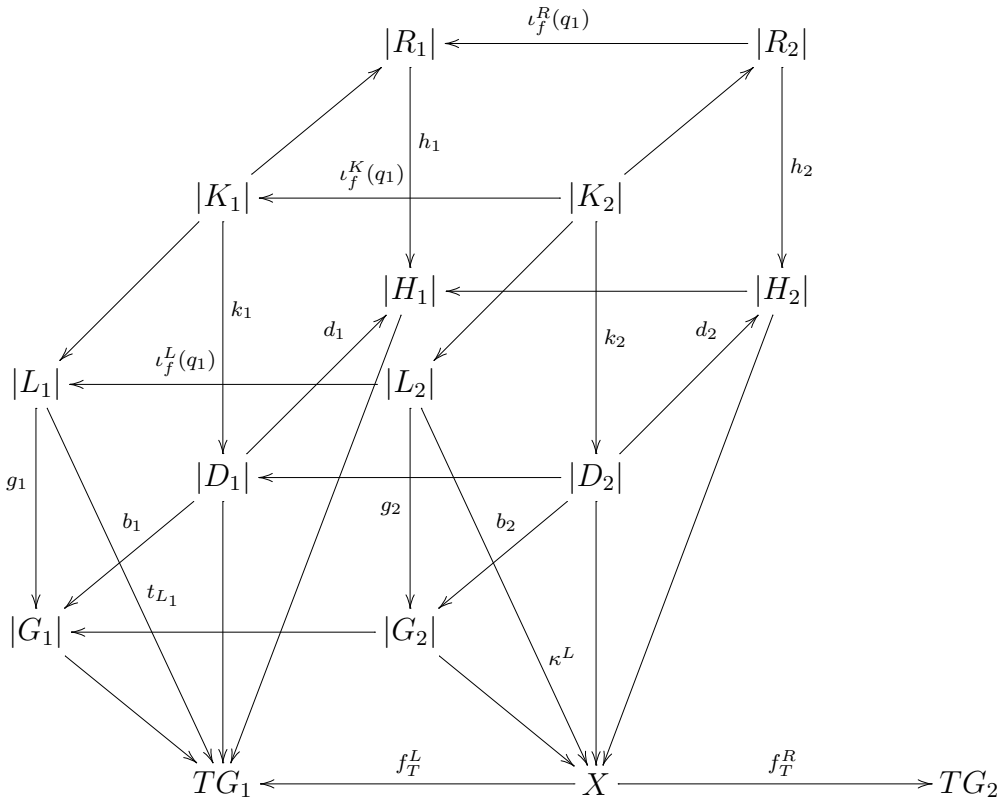


Figure 5.15: Graph grammars morphisms preserve derivations.

Chapter 6

Unfolding and Event Structure Semantics

This chapter introduces a truly concurrent semantics for DPO graph grammars based on a Winskel-style unfolding construction. The work developed in the First Part for contextual and inhibitor nets represents both an intuitive guide and a formal basis for the treatment of graph grammars. In fact, as in contextual nets, the possibility of specifying rewriting steps which preserve part of the state leads to *asymmetric conflicts* between productions. Furthermore the *dangling condition*, a part of the application condition which prevents the application of a production when it would leave dangling edges, has a natural encoding in the setting of inhibitor nets: the edges whose presence prevents the application of a production can be seen as inhibitor places for that production.

First, *nondeterministic occurrence grammars*, which generalize the (deterministic) occurrence grammars of [CMR96], are defined as safe grammars satisfying suitable acyclicity and well-foundedness conditions. As occurrence inhibitor nets are defined without considering the inhibitor arcs, the requirements on occurrence grammars disregard the constraints imposed by the dangling condition. Consequently not all the productions of an occurrence grammar are really executable. The possible deterministic computations of an occurrence grammar are captured by considering the *configurations* of the grammar which are later shown to be closely related to the configurations of the corresponding event structure.

An *unfolding construction* is proposed, which associates to each (consuming) graph grammar a nondeterministic occurrence grammar representing its behaviour. As for nets, the idea consists of starting from the start graph of the grammar, applying in all possible ways the productions of the grammar, and recording in the unfolding each occurrence of production and each new graph item generated by the rewriting process, both enriched with the corresponding causal history. Consistently with the notion of occurrence grammar, in the construction of the unfolding the productions are applied without considering the dangling condition.

The "object level" unfolding construction naturally works on the whole class of graph grammars essentially because graph grammars are strictly more concrete than Petri nets. Hence, exploiting the additional information given by the concrete identity of the items in a grammar, one can avoid the kind of confusion arising in Petri nets related to the presence of several causally indistinguishable items.

Then we face the problem of turning the unfolding construction into a functor establishing a coreflection between the category of graph grammars and that of occurrence grammars. We first restrict to those grammars where the start graph and the items produced by each production are injectively typed. Such grammars, by analogy with the corresponding subclass of Petri nets, are called *semi-weighted* grammars, and the corresponding full subcategory of **GG** is denoted by **SW-GG**. We show that indeed in this case the unfolding extends to a functor $\mathscr{U}_g : \mathbf{SW\text{-}GG} \to \mathbf{O\text{-}GG}$ which is right adjoint of the inclusion $\mathscr{I}_O : \mathbf{O\text{-}GG} \to \mathbf{SW\text{-}GG}$, and thus establishes a coreflection between the two categories.

It is also shown that the restriction to semi-weighted graph grammars is essential for the above categorical construction. However, suitably restricting graph grammars morphisms to still interesting subclasses (comprising, for instance, the morphisms of [Rib96, HCEL96]) it is possible to regain the categorical semantics for general, possibly non semi-weighted, grammars.

Finally, from the unfolding we can easily extract an event structure and a domain semantics. As suggested by the correspondence between occurrence inhibitor nets and graph grammars, *inhibitor event structures*, the extension of prime event structures introduced in CHAPTER 4, are expressive enough to represent the structure of graph grammar computations. Thus an IES can be naturally associated to each occurrence grammar via a functorial construction. Then, the results of CHAPTER 4 relating **IES** and **Dom** allow us to obtain a domain and prime event structure semantics.

$$\mathbf{SW\text{-}GG} \underset{\mathscr{U}_g}{\overset{\mathscr{I}_O}{\underset{\longrightarrow}{\overset{\longleftarrow}{\bot}}}} \mathbf{O\text{-}GG} \xrightarrow[\mathscr{E}_g]{} \mathbf{IES} \underset{\mathscr{L}_i}{\overset{\mathscr{P}_i}{\rightleftarrows}} \mathbf{Dom} \underset{\mathscr{P}}{\overset{\mathscr{L}}{\underset{\longrightarrow}{\overset{\longleftarrow}{\sim}}}} \mathbf{PES}$$

The notions of nondeterministic occurrence grammar and of grammar morphism suggest a notion of *nondeterministic graph process*, the prototypical example of nondeterministic process being the unfolding. Analogously to what happens in Petri net theory, a nondeterministic process of a grammar $\mathscr{G}$ consists of a (suitable) grammar morphism from an occurrence grammar to the grammar $\mathscr{G}$. Nicely, as we will see in the next chapter, our nondeterministic graph processes are a consistent generalizations of the graph processes of [CMR96, BCM98a], namely in the deterministic case they reduce to the same notion.

To conclude, it is worth stressing that an unfolding construction for a different approach to graph transformation, called *single-pushout* (SPO) approach, has been proposed by Ribeiro in her doctoral thesis [Rib96]. Although conceptually such construction is close to ours, we will see that, concretely, the differences between the

two settings, like the absence of the application condition in the SPO approach, a different notion of "enabling" allowing for the concurrent application of productions related by asymmetric conflict and a different choice of grammar morphisms, makes difficult a direct comparison.

The rest of the chapter is organized as follows. Section 6.1 introduces the notion of nondeterministic occurrence grammar and gives some more insights on the relation between graph grammars and inhibitor nets. Relying on the notion of occurrence grammar, Section 6.2 introduces nondeterministic graph processes. Section 6.3 describes the unfolding construction for graph grammars, whose categorical properties are then investigated in Section 6.4. Then Section 6.5 shows how an IES can be extracted from an occurrence grammar, thus providing, through the unfolding construction, an event structure semantics for graph grammars. Finally, in Section 6.6 we compare the unfolding construction in this chapter with that proposed in [Rib96] for the SPO approach.

6.1 Nondeterministic occurrence grammars

A first step towards the definition of nondeterministic occurrence grammars is a suitable notion of safeness for grammars [CMR96], generalizing that for P/T nets, which requires that each place contains at most one token in any reachable marking.

DEFINITION 6.1 ((STRONGLY) SAFE GRAMMAR)
A grammar $\mathscr{G} = \langle TG, G_s, P, \pi \rangle$ *is* (strongly) safe *if, for all* H *such that* $G_s \Rightarrow^* H$, H *has an injective typing morphism.*

The definition can be understood by thinking of nodes and edges of the type graph as a generalization of places in Petri nets. In this view the number of different items of a graph which are typed on a given item of the type graph corresponds to the number of tokens contained in a place, and thus the condition of safeness for a marking is generalized to typed graphs by the injectivity of the typing morphism.

Strongly safe graph grammars (hereinafter called just *safe grammars*) admit a natural net-like pictorial representation, where items of the type graph and productions play, respectively, the role of places and transitions of Petri nets. The basic observation is that typed graphs having an injective typing morphism can be safely identified with the corresponding subgraphs of the type graph (just thinking of injective morphisms as inclusions). Therefore, in particular, each graph $\langle |G|, t_G \rangle$ reachable in a safe grammar can be identified with the subgraph $t_G(|G|)$ of the type graph TG, and thus it can be represented by suitably decorating the nodes and edges of TG. Concretely, a node is drawn as a filled circle if it belongs to $t_G(|G|)$ and as an empty circle otherwise, while an edge is drawn as a continuous line if it is in $t_G(|G|)$ and as a dashed line otherwise (see Figure 6.1). This is analogous to the usual technique of representing the marking of a safe net by putting a token in each place which belongs to the marking.

With the above identification, in each derivation of a safe grammar starting from the start graph a production can be applied only to the subgraph of the type graph which is the image via the typing morphism of its left-hand side. Therefore according to its typing, we can think that a production *produces*, *preserves* and *consumes* items of the type graph. This is expressed by drawing productions as arrow-shaped boxes, connected to the consumed and produced resources by incoming and outcoming arrows, respectively, and to the preserved resources by undirected lines. Figure 6.1 presents two examples of safe grammars, with their pictorial representation. Notice that the typing morphisms for the start graph and the productions are represented by suitably labelling the involved graphs with items of the type graph.

Using a net-like language, we speak of *pre-set* ${}^\bullet q$, *context* $\underline{q}$ and *post-set* $q^\bullet$ of a production q. The notions of pre-set, post-set and context of a production have a clear interpretation only for safe grammars. However for technical reasons it is preferable to define them for general graph grammars.

DEFINITION 6.2 (PRE-SET, POST-SET, CONTEXT)
Let $\mathscr{G}$ be a graph grammar. For any $q \in P$ we define

$$ {}^\bullet q = t_{L_q}(|L_q| - l_q(|K_q|)) \qquad q^\bullet = t_{R_q}(|R_q| - r_q(|K_q|)) $$
$$ \underline{q} = t_{K_q}(|K_q|) $$

seen as sets of nodes and edges, and we say that q consumes, creates *and* preserves *items in ${}^\bullet q$, $q^\bullet$ and $\underline{q}$, respectively. Similarly for a node or an edge x in TG we write ${}^\bullet x$, $\underline{x}$ and $x^\bullet$ to denote the sets of productions which produce, preserve and consume x, respectively.*

For instance, for grammar $\mathscr{G}_2$ in Figure 6.1, the pre-set, context and post-set of production q_1 are ${}^\bullet q_1 = \{C\}$, $\underline{q_1} = \{B\}$ and $q_1{}^\bullet = \{A, L\}$, while for the node B, ${}^\bullet B = \emptyset$, $\underline{B} = \{q_1, q_2, q_3\}$ and $B^\bullet = \{q_4\}$.

Observe now that because of the dangling condition, a production q which consumes a node n can be applied only if there are no edges with source or target in n which remain dangling after the application of q. In other words, if $n \in {}^\bullet q$, $e \notin {}^\bullet q$ and $n \in \{s(e), t(e)\}$ then the application of q is *inhibited* by the presence of e. By analogy with inhibitor nets we introduce the *inhibitor set* of a production.

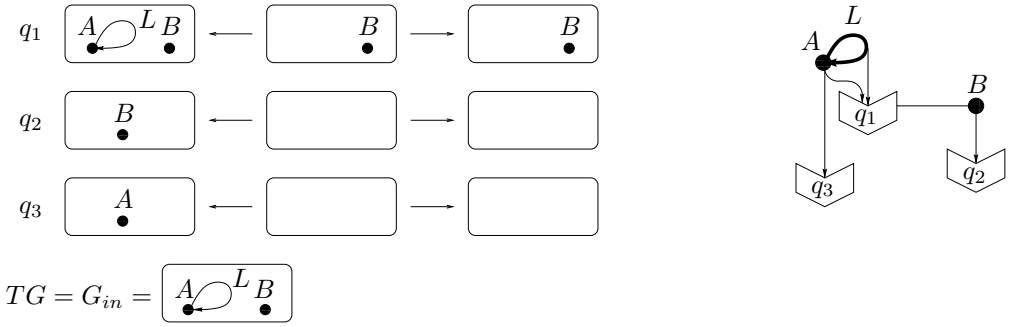
DEFINITION 6.3 (INHIBITOR SET)
Let $\mathscr{G}$ be a graph grammar. The inhibitor set *of a production $q \in P$ is defined by*

$$ {}^\circledcirc q = \{e \in E_{TG} \mid \exists n \in {}^\bullet q.\ n \in \{s(e), t(e)\} \ \wedge \ e \notin {}^\bullet q\} $$

Similarly, for an edge $e \in E_{TG}$ we define the inhibitor set of e as the set of productions inhibited by e, namely ${}^\circledcirc e = \{q \in P \mid e \in {}^\circledcirc q\}$.

For instance, in the grammar $\mathscr{G}_2$ of Figure 6.1 the inhibitor set of q_4 is ${}^\circledcirc q_4 = \{L\}$, while for the edge L we have ${}^\circledcirc L = \{q_4\}$.

Grammar $\mathscr{G}_1$



Grammar $\mathscr{G}_2$

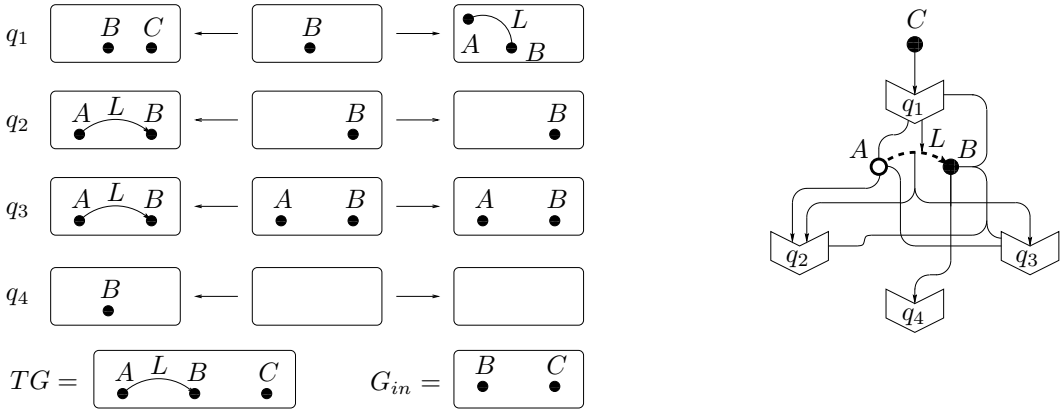


Figure 6.1: Two safe grammars and their net-like representation.

Note that with the above definition, a production q of a safe grammar $\mathscr{G}$ satisfies the dangling condition (when using its typing as match) in a subgraph G of the type graph TG if and only if ${}^{\circledcirc}q \cap G = \emptyset$.

The correspondence between safe grammars and inhibitor nets can be made more explicit by observing that we can associate to any safe grammar $\mathscr{G} = \langle G_s, TG, P, \pi \rangle$ an inhibitor net $N_{\mathscr{G}}$ having the items of TG as places, G_s as initial marking, P as set of transitions with pre-set, post-set, context and inhibitor set of each transition defined exactly as in the grammar. Figure 6.2 shows the inhibitor nets corresponding to the safe grammars $\mathscr{G}_1$ and $\mathscr{G}_2$ in Figure 6.1. It is possible to show that the grammar $\mathscr{G}$ and the net $N_{\mathscr{G}}$ have essentially the same behaviour in the sense that each derivation in $\mathscr{G}$ corresponds to a step sequence in $N_{\mathscr{G}}$ using the same productions and leading to the same state, and vice versa. Although this fact is not used in a formal way, the proposed translation can help in understanding how the work on contextual and inhibitor nets influences the treatment of graph grammars in this chapter. It is important to observe that the inhibitor net associated to a safe graph grammar by the described translation has a very particular shape, as expressed below.

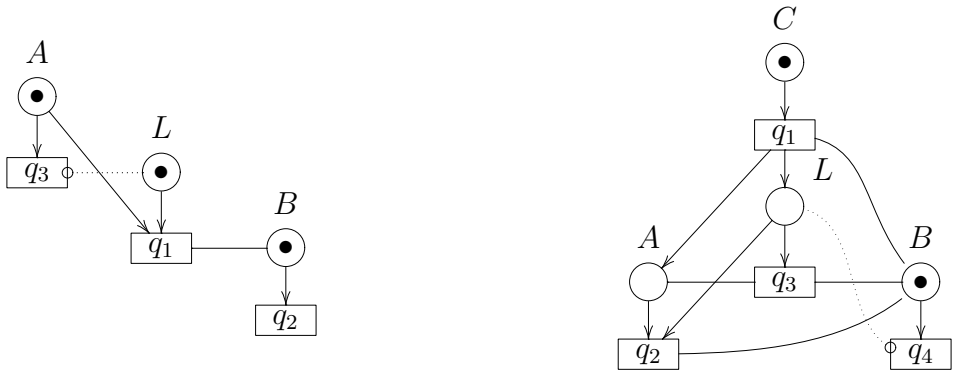


Figure 6.2: The inhibitor nets corresponding to the grammars $\mathscr{G}_1$ and $\mathscr{G}_2$ in Figure 6.1.

REMARK 6.4
Let $\mathscr{G}$ be a safe grammar and let $q, q' \in P$ be two productions of $\mathscr{G}$. Observe that whenever $q^\bullet \cap {}^\circledcirc q' \neq \emptyset$ then necessarily $q \nearrow q'$. In fact let $e \in q^\bullet \cap {}^\circledcirc q'$. By definition of ${}^\circledcirc q'$ the production q' consumes a node n which is the source or the target of e. Since, by definition of $q^\bullet$, the production q produces the edge e, it must produce or preserve the node n. Consequently $n \in (q^\bullet \cup \underline{q}) \cap {}^\bullet q'$ and thus $q \nearrow q'$. For similar reasons, if ${}^\circledcirc q' \cap {}^\bullet q'' \neq \emptyset$ then $q'' \nearrow q'$.

Hence, differently from what happens in general inhibitor nets, if a production q can inhibit a production q' then q cannot be applied after q'. For instance, consider the net $N_{\mathscr{G}_2}$ in the right part of Figure 6.2. The production q_1 can inhibit q_4 since it produces a token in L and indeed it must precede q_4, since $B \in \underline{q_1} \cap {}^\bullet q_4$. Similarly q_3 re-enables q_4 and $q_3 \nearrow q_4$. This means that, in a sense, (safe) graph grammars computations are simpler than (safe) inhibitor nets computations.

Following the approach adopted for inhibitor nets, the causal and asymmetric conflict relations for a graph grammar are defined without taking into account the inhibitor sets, namely disregarding the dangling condition.

DEFINITION 6.5 (CAUSAL RELATION)
The causal relation *of a grammar $\mathscr{G}$ is the binary relation $<$ over $Elem(\mathscr{G})$ defined as the least transitive relation satisfying: for any node or edge x in the type graph TG and for productions $q_1, q_2 \in P$*

1. *if $x \in {}^\bullet q_1$ then $x < q_1$;*
2. *if $x \in q_1{}^\bullet$ then $q_1 < x$;*
3. *if $q_1{}^\bullet \cap \underline{q_2} \neq \emptyset$ then $q_1 < q_2$.*

As usual $\leq$ denotes the reflexive closure of $<$. Moreover, for $x \in Elem(\mathscr{G})$ we write $\lfloor x \rfloor$ for the set of causes of x in P, namely $\{q \in P \mid q \leq x\}$.

DEFINITION 6.6 (ASYMMETRIC CONFLICT)
The asymmetric conflict relation *of a grammar $\mathscr{G}$ is the binary relation $\nearrow$ over the set P of productions, defined by:*

1. *if $\underline{q_1} \cap {}^\bullet q_2 \neq \emptyset$ then $q_1 \nearrow q_2$;*
2. *if ${}^\bullet q_1 \cap {}^\bullet q_2 \neq \emptyset$ and $q_1 \neq q_2$ then $q_1 \nearrow q_2$;*
3. *if $q_1 < q_2$ then $q_1 \nearrow q_2$.*

A *nondeterministic occurrence grammar* is an acyclic grammar which represents, in a branching structure, several possible computations starting from its start graph and using each production at most once. Again the dangling condition is not considered in the definition.

DEFINITION 6.7 ((NONDETERMINISTIC) OCCURRENCE GRAMMAR)
A (nondeterministic) occurrence grammar *is a graph grammar $\mathscr{O} = \langle TG, G_s, P, \pi \rangle$ such that*

1. *the causal relation $\leq$ is a partial order, and for any $q \in P$ the set $\lfloor q \rfloor$ is finite and the asymmetric conflict $\nearrow$ is acyclic on $\lfloor q \rfloor$;*
2. *the start graph G_s coincides with the set $Min(\mathscr{O})$ of the items of the type graph TG which are minimal with respect to causality $\leq$ (with the graphical structure inherited from TG and typed by the inclusion);*
3. *each edge or node x in TG is created by at most one production in P, namely $|{}^\bullet x| \leq 1$;*
4. *for each production $q : L_q \stackrel{l_q}{\leftarrow} K_q \stackrel{r_q}{\rightarrow} R_q$, the typing t_{L_q} is injective on the "consumed part" $|L_q| - l_q(|K_q|)$, and similarly t_{R_q} is injective on the "produced part" $|R_q| - r_q(|K_q|)$.*

We denote by **O-GG** *the full subcategory of* **GG** *having occurrence grammars as objects.*

Since the start graph of an occurrence grammar $\mathscr{O}$ is determined by $Min(\mathscr{O})$, we often do not mention it explicitly.

Intuitively, conditions (1)–(3) recast in the framework of graph grammars the analogous conditions of occurrence contextual nets. Condition (4), is closely related to safeness and requires that each production consumes and produces items with "multiplicity" one. Observe that, together with acyclicity of $\nearrow$, it disallows the presence of some productions which surely could never be applied, because they fail to satisfy the identification condition with respect to the typing morphism.

The next proposition shows that, by the defining conditions, each occurrence grammar is *safe*.

PROPOSITION 6.8 (OCCURRENCE AND SAFE GRAMMARS)
Each occurrence grammar $\mathscr{O}$ is safe.

PROOF. Let $\mathscr{O} = \langle TG, P, \pi \rangle$ be an occurrence grammar and let $\rho : G_s \Rightarrow^* G_n$ be a derivation in $\mathscr{O}$. Since the grammar is consuming, namely each production has a non empty pre-set, and causality $\leq$ is a partial order (there are no "cycles"), every derivation in the grammar starting from the start graph can apply each production at most once. Without any loss of generality we can assume that each direct derivation of ρ applies a single production.

We show by induction on the order n of the derivation that the graph G_n is injective. For $n = 0$ just recall that G_s is a subgraph of TG, typed by the inclusion. If $n > 0$, by inductive hypothesis, G_{n-1} is injective. Moreover the typing of q_n is injective on $|R_{q_n}| - r_{q_n}(|K_{q_n}|)$. This observation, together with the fact that the items of the start graph have empty pre-set (they are minimal with respect to causality) and each item of the type graph is produced by at most one production implies that the graph G_n, which is obtained from G_{n-1} by first "removing" $L_{q_n} - l_{q_n}(K_{q_n})$ and then "adding" $R_{q_n} - r_{q_n}(K_{q_n})$, is injective. □

Disregarding the dangling condition in the definition of occurrence grammar has as a consequence the fact that, analogously to what happens for inhibitor nets, we are not guaranteed that every production of an occurrence grammar is applicable at least in one derivation starting from the start graph. The restrictions to the behaviour imposed by the dangling condition are taken into account when defining the configurations of an occurrence grammar, which represent exactly, in a sense formalized later, all the possible deterministic runs of the grammar.

DEFINITION 6.9 (CONFIGURATION)
A configuration *of an occurrence grammar $\mathscr{O} = \langle TG, P, \pi \rangle$ is a subset $C \subseteq P$ such that*

1. *if $\nearrow_C$ denotes the restriction of the asymmetric conflict relation to C, then $(\nearrow_C)^*$ is a finitary partial order on C;*
2. *C is left-closed with respect to $\leq$, i.e. for all $q \in C$, $q' \in P$, $q' \leq q$ implies $q' \in C$;*
3. *for all $e \in TG$, if ${}^{\circledcirc}e \cap C \neq \emptyset$ and ${}^{\bullet}e \subseteq C$ then $e^{\bullet} \cap C \neq \emptyset$.*

If C satisfies conditions (1) and (2), then it is called a pre-configuration. *The set of all configurations of the grammar $\mathscr{O}$ is denoted by $Conf(\mathscr{O})$.*

Condition (1) ensures that in C there are no $\nearrow$-cycles and thus it excludes the possibility of having in C a subset of productions in conflict. Furthermore it guarantees that each production has to be preceded only by finitely many other productions in the computation represented by the configuration. Condition (2) requires the presence of all the causes of each production. Condition (3) considers the dangling condition: for any edge e in the type graph, if the configuration contains a production q inhibited by e and a production q' producing such an edge

then some production q'' consuming e must be present as well, otherwise, due to the dangling condition, q could not be executed. This requirement is better understood recalling that, as observed in Remark 6.4, in this situation, i.e., when $e \in q'^{\bullet} \cap {}^{\circledcirc}q \neq \emptyset$, necessarily $q' \nearrow q$, namely q' must be applied before q in the computation. Similar considerations apply if the edge e is present in the start graph, i.e., ${}^{\bullet}e = \emptyset$. For example the set of configurations of the grammar $\mathscr{G}_2$ in Figure 6.1 is $Conf(\mathscr{G}_2) = \{\emptyset, \{q_1\}, \{q_1, q_2\}, \{q_1, q_3\}, \{q_1, q_2, q_4\}, \{q_1, q_3, q_4\}, \{q_4\}\}$. The set $S = \{q_1, q_4\}$, is instead only a pre-configuration, since for the edge L we have $q_4 \in {}^{\circledcirc}L$, ${}^{\bullet}L = \{q_1\} \subseteq S$, but the intersection of S with $L^{\bullet} = \{q_2, q_3\}$ is empty.

The notion is reminiscent of that of configuration of an inhibitor event structure. Indeed we will prove later that the configurations of an occurrence grammar are exactly the configurations of the IES associated to the grammar. The fact that an occurrence grammar configuration does not include an explicit choice relation can be understood, for the moment, by recalling Remark 6.4 which implies that the productions in a configuration implicitly determine their order of application.

The claim that configurations represent all and only the deterministic runs of an occurrence grammar is formalized by the following result. We first need the notion of *reachable graph* associated to a configuration, which extends an analogous concept introduced in [CMR96], in the case of deterministic occurrence grammars.

DEFINITION 6.10 (REACHABLE GRAPHS)
Let $\mathscr{O} = \langle TG, P, \pi \rangle$ be an occurrence grammar. For any $C \subseteq P$, finite configuration of $\mathscr{O}$, the reachable set *associated to C is the set of nodes and edges $reach(C) \subseteq Items(TG)$ defined as*

$$reach(C) = (Min(\mathscr{O}) \cup \bigcup_{q \in C} q^{\bullet}) - \bigcup_{q \in C} {}^{\bullet}q.$$

PROPOSITION 6.11 (CONFIGURATIONS AND DERIVATIONS)
Let C be any configuration of an occurrence grammar $\mathscr{O}$. Then $reach(C)$ is a well-defined subgraph of the type graph TG and, moreover, $Min(\mathscr{O}) \Rightarrow^*_C reach(C)$ with a derivation which applies exactly once every production in C, in any order consistent with $(\nearrow_C)^*$. Vice versa for each derivation $Min(\mathscr{O}) \Rightarrow^*_S G$ in $\mathscr{O}$, the set of productions S it applies is a configuration and $G = reach(S)$.

PROOF. Let $C \in Conf(\mathscr{O})$ be any finite configuration of $\mathscr{O}$. The fact that $reach(C)$ is a well-defined subgraph of the type graph TG and $Min(\mathscr{O}) \Rightarrow^*_C reach(C)$ can be proved by induction on $|C|$. The base case in which $|C| = 0$, namely $C = \emptyset$, is trivial since, by definition, $Min(\mathscr{O}) = reach(\emptyset)$. If instead $|C| > 0$ consider any production $q \in C$, maximal with respect to $(\nearrow_C)^*$ (such a production exists since $(\nearrow_C)^*$ is a partial order and C is finite). It is easy to see that $C - \{q\}$ is a configuration and therefore, by inductive hypothesis,

$$Min(\mathscr{O}) \Rightarrow^*_{C-\{q\}} G$$

where $G = reach(C - \{q\})$. By point (1) in the definition of configuration, entailing that $\nearrow_C$ is acyclic, and by point (2) we immediately get that ${}^{\bullet}q \cup \underline{q} \subseteq G$. Furthermore, by point (3), ${}^{\circledcirc}q \cap G = \emptyset$ and thus q satisfies also the dangling condition. Recalling that q satisfies the identification

condition by definition of occurrence grammar, we deduce that q is applicable to G and thus $G \Rightarrow_q (G - {}^\bullet q) \cup q^\bullet$. Since in an occurrence grammar the post-sets of productions are all disjoint we conclude that $(G - {}^\bullet q) \cup q^\bullet = \mathit{reach}(C)$ and thus

$$Min(\mathscr{O}) \Rightarrow_C^* \mathit{reach}(C)$$

The second part of the proposition can be proved essentially by reversing the above steps. □

As an immediate consequence of the previous result, a production which does not satisfy the dangling condition in any graph reachable from the start graph (and thus which is never applicable) is not part of any configuration. For example, q_3 does not appear in the set of configurations of $\mathscr{G}_1$, $\mathit{Conf}(\mathscr{G}_1) = \{\emptyset, \{q_1\}, \{q_2\}, \{q_1, q_2\}\}$.

6.2 Nondeterministic graph processes

In the theory of Petri nets the notion of occurrence net is strictly related to that of process. A (non)deterministic net process is a (non)deterministic occurrence net with a suitable morphism to the original net. Similarly, nondeterministic occurrence grammars can be used to define a notion of *nondeterministic graph processes*, generalizing the deterministic graph processes of [CMR96, BCM98a]. Then, the unfolding of a grammar, as introduced in the next section, can be seen as a "complete" nondeterministic process of the grammar, expressing all the possible computations of the grammar.

A *nondeterministic graph process* is aimed at representing in a unique "branching" structure several possible computations of a grammar. The underlying occurrence grammar makes explicit the causal structure of such computations since each production can be applied at most once and each item of the type graph can be "filled" at most once. Via the morphism to the original grammar, productions and items of the type graph in the occurrence grammar can be thought of, respectively, as instances of applications of productions and instances of items generated in the original grammar by such applications. Actually, to allow for such an interpretation, some further restrictions have to be imposed on the process morphism. Recall that process morphisms in Petri net theory must map places to places (rather than to multisets of places) and must be total on transitions. Similarly, for graph process morphisms the left component of the type-span is required to be an isomorphism in such a way that the type-span can be thought of simply as a total graph morphism. Furthermore a process morphism cannot map a production to the empty production, a requirement corresponding to totality.

DEFINITION 6.12 (STRONG MORPHISM)
A grammar morphism $f : \mathscr{G}_1 \to \mathscr{G}_2$ is called strong *if*

1. $f_T^L : X_{f_T} \to TG_1$ *is an isomorphism;*
2. $f_P(q_1) \neq \emptyset$, *for any* $q_1 \in P_1$.

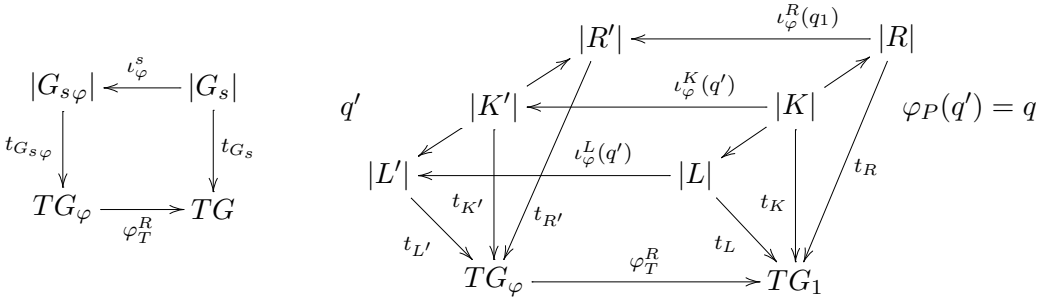


Figure 6.3: Graph processes.

In the following, without any loss of generality, we will always choose as concrete representative of the type-span of a strong grammar morphism f, a span f_T such that the left component f^L_T is the identity id_{TG_1}.

DEFINITION 6.13 (GRAPH PROCESS)
Let $\mathscr{G}$ be a graph grammar. A (marked) graph process *of $\mathscr{G}$ is a strong grammar morphism $\varphi : \mathscr{O}_\varphi \to \mathscr{G}$, where $\mathscr{O}_\varphi$ is an occurrence grammar. We will denote by TG_φ, G_{s_φ}, P_φ and π_φ the components of the occurrence grammar $\mathscr{O}_\varphi$ underlying a process φ.*

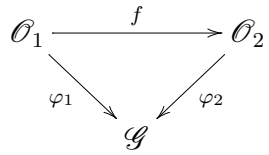
Alternatively, if $\mathbf{GG}^*$ indicates the subcategory of $\mathbf{GG}$ having the same objects and strong grammar morphisms as arrows, then the category of processes of a grammar $\mathscr{G}$ can be simply defined as the comma category $\langle \mathbf{O\text{-}GG} \downarrow \mathscr{G} \rangle$ in $\mathbf{GG}^*$.

It is not difficult to verify that, if $f : \mathscr{G}_1 \to \mathscr{G}_2$ is a strong morphism then, by condition (1) in the definition of grammar morphism (Definition 5.33), $\iota^s_f : |G_{s_2}| \to |G_{s_1}|$ is an isomorphism. Similarly, by condition (2), for each production $q_1 \in P_1$, $\iota_f(q_1)$ is a triple of isomorphisms, namely each production of $\mathscr{G}_1$ is mapped to a production of $\mathscr{G}_2$ with associated isomorphic (untyped) span. Furthermore the pullback-retyping construction induced by f_T becomes extremely simple: when applied to graph G_1 typed over TG_1 it produces (up to isomorphism) the graph $\langle |G_1|, t_{TG_1}; f^R_T \rangle$ obtained by composing the typing morphism of G_1 with f^R_T.

In this situation the conditions on a process φ requiring the preservation of the start graph and of the productions spans, can be expressed by simply asking the commutativity of the diagrams in Figure 6.3, the right one for any production $q' \in P_\varphi$. Moreover, since the left-component of φ_T is assumed to be the identity, to lighten the notation we will sometimes omit the superscript R when denoting the right component φ^R_T.

To understand when two graph processes φ_1 and φ_2 are isomorphic as objects of $\langle \mathbf{O\text{-}GG} \downarrow \mathscr{G} \rangle$ in $\mathbf{GG}^*$, first consider a generic morphism $f : \varphi_1 \to \varphi_2$ in such a category. According to the definition of comma category, $f : \mathscr{O}_1 \to \mathscr{O}_2$ is a grammar

morphism such that $\varphi_2 \circ f = \varphi_1$, as shown below



Since φ_1 and φ_2 are strong morphisms, it immediately follows that the left component of the type span f_T must be an isomorphism (although f is not necessarily strong since f_P may map some productions to the empty one). Moreover the component ι_f of the morphism must be a collection of isomorphisms, which are completely determined by the ι components of φ_1 and φ_2. Exploiting these observations we conclude the following characterization of the isomorphism between processes which essentially states that the isomorphism is completely determined by the right component of the type span and by the production component.

PROPOSITION 6.14 (ISOMORPHISM OF GRAPH PROCESSES)
Let φ_1 and φ_2 be two processes of the grammar $\mathscr{G}$. Then φ_1 and φ_2 are isomorphic if and only if there exists a pair $\langle f_T, f_P \rangle$, where

1. *$f_T : \langle TG_{\varphi_1}, \varphi_{1T} \rangle \to \langle TG_{\varphi_2}, \varphi_{2T} \rangle$ is an isomorphism (of TG-typed graphs);*
2. *$f_P : P_{\varphi_1} \to P_{\varphi_2}$ is a bijection such that $\varphi_{1P} = \varphi_{2P} \circ f_P$;*
3. *the left diagram in Figure 6.4 commutes;*
4. *for each $q_1 : (L_1 \leftarrow K_1 \rightarrow R_1)$ in P_{φ_1}, $q_2 = fp(q_1) : (L_2 \leftarrow K_2 \rightarrow R_2)$ in P_{φ_2}, if $q = \varphi_{1P}(q_1) = \varphi_{2P}(q_2) : (L \leftarrow K \rightarrow R)$ in P, the right diagram in Figure 6.4 commutes.*

To indicate that φ_1 and φ_2 are isomorphic *we write $\varphi_1 \cong \varphi_2$.*

In the sequel when speaking of an isomorphism of processes we will always refer to the pair $\langle f_T, f_P \rangle$ rather than to the entire morphism.

In the next chapter we will restrict to deterministic processes and define an operation of sequential composition on them. As in the case of nets, to have a meaningful notion of composition, one must consider "unmarked" processes, starting from any graph instead that from the start graph of the grammar.

DEFINITION 6.15 (UNMARKED GRAPH PROCESS)
An unmarked strong grammar morphism is a grammar morphism satisfying all the conditions of Definition 6.12, but the preservation of the start graph.

An unmarked graph process *of a graph grammar $\mathscr{G}$ is an unmarked strong grammar morphism $\varphi : \mathscr{O}_\varphi \to \mathscr{G}$, where $\mathscr{O}_\varphi$ is an occurrence grammar.*

All the definitions and results introduced in this section can be easily adapted to unmarked processes simply by forgetting the conditions on the start graph.

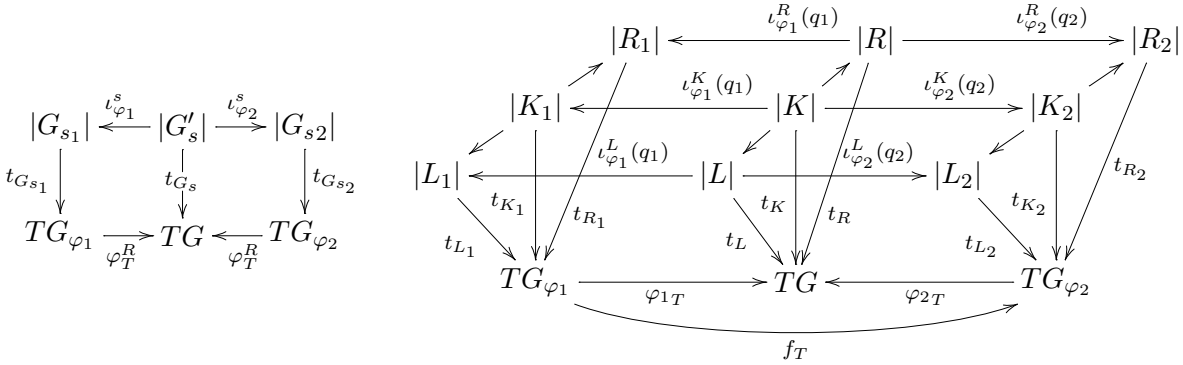


Figure 6.4: Graph process isomorphism.

6.3 Unfolding construction

This section introduces the unfolding construction which, when applied to a consuming grammar $\mathscr{G}$, produces a nondeterministic occurrence grammar $\mathscr{U}_g(\mathscr{G})$ describing the behaviour of $\mathscr{G}$. The unfolding is equipped with a strong grammar morphism $\varphi_{\mathscr{G}}$ to the original grammar $\mathscr{G}$, making it a process of $\mathscr{G}$.

The unfolding is constructed by starting from the start graph of the grammar, then applying in all possible ways its productions, and recording in the unfolding each occurrence of production and each new graph item generated in the rewriting process, both enriched with the corresponding causal history. According to the discussion in the previous section, in the unfolding procedure, productions are applied without considering the dangling condition. Moreover we adopt a notion of concurrency which is "approximated", again in the sense that it does not take care of the precedences between productions induced by the dangling condition. In the analogy between graph grammars and inhibitor nets, this corresponds to applying the unfolding construction to the contextual net obtained by forgetting the inhibitor arcs. Recall that in the case of inhibitor nets, the net obtained by unfolding the underlying contextual net is finally "enriched" by inserting again the inhibitor arcs. Since for a graph grammar the inhibitor set of a production is implicitly given by the typing of the production, the result of the unfolding construction needs not to be further modified.

DEFINITION 6.16 (QUASI-CONCURRENT GRAPH)
Let $\mathscr{O} = \langle TG, P, \pi \rangle$ be an occurrence grammar. A subgraph G of TG is called quasi-concurrent *if*

1. $\bigcup_{x \in G} \lfloor x \rfloor$ *is a pre-configuration;*
2. $\neg(x < y)$ *for all* $x, y \in G$.

Another basic ingredient for the unfolding is the gluing operation. It can be interpreted as a "partial application" of a rule to a given match, in the sense that it generates the new items as specified by the production (i.e., items of right-hand side not in the interface), but items that should have been deleted are not affected: intuitively, this is because such items may still be used by another production in the nondeterministic unfolding.

DEFINITION 6.17 (GLUING)
Let q be a TG-typed production, G a TG-typed graph and $m : L_q \to G$ a graph morphism. We define, for any symbol $*$, the gluing of G and R_q along K_q, according to m and marked by $*$, *denoted by $glue_*(q, m, G)$, as the graph $\langle N, E, s, t\rangle$, where:*

$$N = N_G \cup m_*(N_{R_q}) \qquad E = E_G \cup m_*(E_{R_q})$$

with m_* defined by: $m_*(x) = m(x)$ if $x \in K_q$ and $m_*(x) = \langle x, *\rangle$ otherwise. The source and target functions s and t, and the typing are inherited from G and R_q.

The gluing operation keeps unchanged the identity of the items already in G, and records in each newly added item from R_q the given symbol $*$. Notice that the gluing, as just defined, is a concrete deterministic definition of the pushout of the arrows $G \overset{m}{\leftarrow} L_q \overset{l_q}{\hookleftarrow} K_q$ and $K_q \overset{r_q}{\hookrightarrow} R_q$.

As described below, the unfolding of a graph grammar is obtained as the limit of a chain of occurrence grammars, which approximate the unfolding up to a certain causal depth. The next definition formally introduces the notion of depth.

DEFINITION 6.18 (DEPTH)
Let $\mathscr{O} = \langle TG, P, \pi\rangle$ be an occurrence grammar. The function depth : $Elem(\mathscr{O}) \to \mathbb{N}$ is defined inductively as follows:

$$\begin{array}{ll} depth(x) = 0 & \text{for } x \in |G_s| = Min(\mathscr{O}); \\ depth(q) = \max\{depth(x) \mid x \in {}^\bullet q \cup \underline{q}\} + 1 & \text{for } q \in P; \\ depth(x) = depth(q) & \text{for } x \in q^\bullet. \end{array}$$

It is not difficult to prove that *depth* is a well-defined total function, since infinite descending chains of causality are disallowed in occurrence grammars. Moreover, given an occurrence grammar $\mathscr{O}$, the grammar containing only the items of *depth* less or equal to n, denoted by $\mathscr{O}^{[n]}$, is a well-defined occurrence grammar. As expected the following result holds.

PROPOSITION 6.19
An occurrence grammar $\mathscr{O}$ is the (componentwise) union of its subgrammars $\mathscr{O}^{[n]}$, of depth n.

Moreover it is not difficult to see that if $g : \mathscr{O} \to \mathscr{G}$ is a grammar morphism, then for any $n \in \mathbb{N}$, g restricts to a morphism $g^{[n]} : \mathscr{O}^{[n]} \to \mathscr{G}$. In particular, if $TG^{[n]}$ denotes the type graph of $\mathscr{O}^{[n]}$, then the type-span of $g^{[n]}$ will be the equivalence class of

$$TG^{[n]} \xleftarrow[g_T^{L[n]}]{} X^{[n]} \xrightarrow[g_T^{R[n]}]{} TG_{\mathscr{G}}$$

where $X^{[n]} = \{x \in X_g \mid g_T^L(x) \in TG^{[n]}\}$. Vice versa each morphism $g : \mathscr{O} \to \mathscr{G}$ is uniquely determined by its truncations at finite depths.

We are now ready to present the unfolding construction.

DEFINITION 6.20 (UNFOLDING)
Let $\mathscr{G} = \langle TG, G_s, P, \pi\rangle$ be a (consuming) graph grammar. We inductively define, for each n, an occurrence grammar $\mathscr{U}_g(\mathscr{G})^{[n]} = \langle TG^{[n]}, P^{[n]}, \pi^{[n]}\rangle$ and a morphism $\varphi^{[n]} = \langle {\varphi_T}^{[n]}, {\varphi_P}^{[n]}, \iota^{[n]}\rangle : \mathscr{U}_g(\mathscr{G})^{[n]} \to \mathscr{G}$. Then the unfolding *$\mathscr{U}_g(\mathscr{G})$ and the* folding morphism *$\varphi_{\mathscr{G}} : \mathscr{U}_g(\mathscr{G}) \to \mathscr{G}$ are the occurrence grammar and strong grammar morphism defined as the componentwise union of $\mathscr{U}_g(\mathscr{G})^{[n]}$ and $\varphi^{[n]}$, respectively.*

Since each morphism $\varphi^{[n]}$ is strong, assuming that the left component of the type-span ${\varphi_T}^{[n]}$ is the identity on $TG^{[n]}$, we only need to define the right component $\varphi_T^{R[n]} : TG^{[n]} \to TG$, which, by the way, makes $\langle TG^{[n]}, \varphi_T^{R[n]}\rangle$ a TG-typed graph.

($\mathbf{n = 0}$) The components of the grammar $\mathscr{U}_g(\mathscr{G})^{[0]}$ are $TG^{[0]} = |G_s|$, $P^{[0]} = \pi^{[0]} = \emptyset$, while morphism $\varphi^{[0]} : \mathscr{U}_g(\mathscr{G})^{[0]} \to \mathscr{G}$ is defined by $\varphi_T^{R[0]} = t_{G_s}$, ${\varphi_P}^{[0]} = \emptyset$, and ${\iota^{[0]}}^s = id_{|G_s|}$.

($\mathbf{n \to n+1}$) Given $\mathscr{U}_g(\mathscr{G})^{[n]}$, the occurrence grammar $\mathscr{U}_g(\mathscr{G})^{[n+1]}$ is obtained by extending it with all the possible production applications to quasi-concurrent subgraphs of the type graph of $\mathscr{U}_g(\mathscr{G})^{[n]}$. More precisely, let $M^{[n]}$ be the set of pairs $\langle q, m\rangle$ such that $q \in P$ is a production in $\mathscr{G}$ and $m : L_q \to \langle TG^{[n]}, \varphi_T^{R[n]}\rangle$ is a match satisfying the identification condition, with $m(|L_q|)$ quasi-concurrent *subgraph of $TG^{[n]}$. Then $\mathscr{U}_g(\mathscr{G})^{[n+1]}$ is the occurrence grammar resulting after performing the following steps for each $\langle q, m\rangle \in M^{[n]}$.*

- *Add to $P^{[n]}$ the pair $\langle q, m\rangle$ as a new production name and extend ${\varphi_P}^{[n]}$ so that ${\varphi_P}^{[n]}(\langle q, m\rangle) = q$. Intuitively, $\langle q, m\rangle$ represents an occurrence of q, where the match m is needed to record the "history".*

- *Extend the type graph $TG^{[n]}$ by adding to it a copy of each item generated by the application q, marked by $\langle q, m\rangle$ (in order to keep trace of the history). The morphism $\varphi_T^{R[n]}$ is extended consequently. More formally, the TG-typed graph $\langle TG^{[n]}, \varphi_T^{R[n]}\rangle$ is replaced by $glue_{\langle q,m\rangle}(q, m, \langle TG^{[n]}, \varphi_T^{R[n]}\rangle)$.*

- *The production $\pi^{[n]}(\langle q, m\rangle)$ has the same untyped span of $\pi(q)$ and the morphisms $\iota^{[n]}(\langle q, m\rangle)$ are identities, that is $\iota(\langle q, m\rangle) = \langle id_{|L_q|}, id_{|K_q|}, id_{|R_q|}\rangle$. The typing of the left-hand side and of the interface is determined by m, and each item x of the right-hand side which is not in the interface is typed over the corresponding new item $\langle x, \langle q, m\rangle\rangle$ of the type graph.*

It is not difficult to verify that for each n, $\mathscr{U}_g(\mathscr{G})^{[n]}$ is a (finite depth) nondeterministic occurrence grammar, and $\mathscr{U}_g(\mathscr{G})^{[n]} \subseteq \mathscr{U}_g(\mathscr{G})^{[n+1]}$, componentwise. Therefore $\mathscr{U}_g(\mathscr{G})$ is a well-defined occurrence grammar. Similarly for each $n \in \mathbb{N}$ we have that $\varphi^{[n]}$ is a well-defined morphism from $\mathscr{U}_g(\mathscr{G})^{[n]}$ to $\mathscr{G}$, which is the restriction to $\mathscr{U}_g(\mathscr{G})^{[n]}$ of $\varphi^{[n+1]}$. This induces a unique morphism $\varphi_{\mathscr{G}} : \mathscr{U}_g(\mathscr{G}) \to \mathscr{G}$.

The deterministic gluing construction ensures that, at each step, the order in which productions are applied does not influence the final result of the step. Moreover if a production is applied twice at the same match (even if in different steps), the generated items are always the same and thus they appear only once in the unfolding.

It is possible to show that the unfolding construction applied to an occurrence grammar yields a grammar which is isomorphic to the original one. This is essentially a consequence of the fact that for each production q of an occurrence grammar the (image via the typing morphism of the) left-hand side of q is always quasi-concurrent and the typing morphism t_{L_q} satisfies the identification condition, as it can easily derived from Definition 6.1.

6.4 The unfolding as a universal construction

The unfolding construction has been defined, up to now, only at "object level". This section faces the problem of characterizing the unfolding as a coreflection between suitable categories of graph grammars and of occurrence grammars. As in the case of contextual and inhibitor nets, we first restrict to a full subcategory **SW-GG** of **GG** where objects satisfy conditions analogous to those defining semi-weighted P/T Petri nets. Then we show that the unfolding construction can be extended to a functor $\mathscr{U}_g$: **SW-GG** $\to$ **O-GG** that is right adjoint to the inclusion functor $\mathscr{I}_O$: **O-GG** $\to$ **SW-GG**.

The restriction to the semi-weighted case is essential for the above categorical construction when one uses general morphisms. However, suitably restricting graph grammars morphisms to still interesting subclasses (comprising, for instance, the morphisms of [Rib96, HCEL96]) it is possible to regain the categorical semantics for general, possibly non semi-weighted, grammars.

6.4.1 Unfolding of semi-weighted graph grammars

A graph grammar is semi-weighted if the start graph is injective and the right-hand side of each production is injective when restricted to produced items (namely, to the items which are not in the interface). It is possible to show that, if we encode a Petri net N as a grammar $\mathscr{G}_N$, according to the translation sketched in Section 5.1.1, then N is a semi-weighted net if and only if $\mathscr{G}_N$ is a semi-weighted grammar.

DEFINITION 6.21 (SEMI-WEIGHTED GRAMMARS)
A TG-typed production $L \leftarrow K \rightarrow R$ is called semi-weighted *if t_R is injective on the "produced part" of R, namely on $|R| - r(|K|)$. A grammar $\mathscr{G}$ is called* semi-weighted *if the start graph G_s is injectively typed and for any $q \in P$ the production $\pi(q)$ is semi-weighted. We denote by* **SW-GG** *the full subcategory of* **GG** *having semi-weighted grammars as objects.*

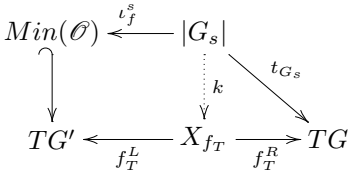
The coreflection result strongly relies on the technical property which is stated in the next lemma. It is important to notice that this is a key point where the restriction to semi-weighted grammars plays a role, since, as we will see, the lemma fails to hold for arbitrary grammars.

LEMMA 6.22
Let $\mathscr{G} = \langle TG, G_s, P, \pi \rangle$ be a semi-weighted grammar, let $\mathscr{O} = \langle TG', G'_s, P', \pi' \rangle$ be an occurrence grammar and let $f : \mathscr{O} \rightarrow \mathscr{G}$ be a grammar morphism. Then the type span $[f_T]$ of the morphism is relational.

PROOF. Recall from Section 5.3 (Definition 5.32) that the span $f_T : TG' \leftrightarrow TG$ is relational if $\langle f_T^L, f_T^R \rangle : X_{f_T} \rightarrow TG \times TG'$ is mono. In turn, in the categories **Set** and **Graph** this amounts to say

$$\forall x, y \in X_{f_T}.\ f_T^R(x) \neq f_T^R(y) \ \vee \ f_T^L(x) \neq f_T^L(y)$$

We proceed by contraposition. Consider $x, y \in X_{f_T}$ such that $f_T^L(x) = f_T^L(y) = z'$ and $f_T^R(x) = f_T^R(y) = z$. Since $\mathscr{O}$ is an occurrence grammar, necessarily z' is in the start graph or in the post-set of some production. Let us assume that $z' \in Min(\mathscr{O})$. By definition of grammar morphism, there exists a morphism $k : |G_s| \rightarrow X_{f_T}$ such that the following diagram commutes and the square is a pullback, where the unlabelled arrow is an inclusion:



The fact that $f_T^L(x) = f_T^L(y) = z$ and $z \in Min(\mathscr{O})$ implies that there are $x'', y'' \in |G_s|$ such that $k(x'') = x$ and $k(y'') = y$. Recalling that the triangle on the right commutes we have

$$t_{G_s}(x'') = f_T^L(k(x'')) = f_T^L(x) = f_T^L(y) = f_T^L(k(y'')) = t_{G_s}(y'')$$

Since the graph G_s is injectively typed, we conclude that $x'' = y''$, and thus $x = k(x'') = k(y'') = y$ that is what we wanted to prove. A similar reasoning applies if the item z belong the post-set of some production, since the grammar is semi-weighted and thus productions are injectively typed on the produced part. □

Observe that, as an immediate consequence of the above lemma, if $f : \mathscr{O} \rightarrow \mathscr{G}$ is a grammar morphism, where $\mathscr{G}$ is a semi-weighted grammar and $\mathscr{O}$ is an occurrence grammar, then the morphism k, such that $f_T\{\iota_f^s, k\}(G_s, G_{s'})$ (see Definition 5.33, condition (1)) is uniquely determined. Similarly, for each $q \in P$, with $q' = f_P(q)$, the morphisms k^L, k^K and k^R such that $f_T\{\iota_f^X(q), k^X\}(X_q, X_{q'})$ for $X \in \{L, K, R\}$ (see Definition 5.33, condition (2)) are uniquely determined.

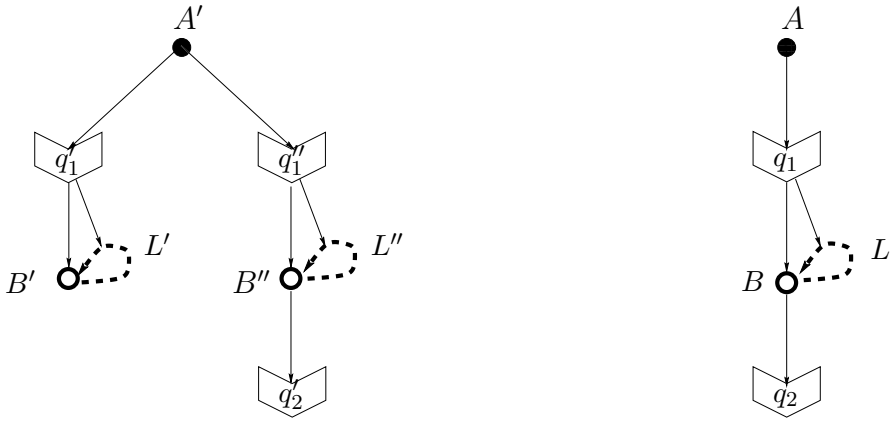


Figure 6.5: The construction mapping safe grammars into i-nets is not functorial.

Some considerations on morphisms between occurrence grammars are now in order. Let $f : \mathscr{O}_1 \to \mathscr{O}_2$ be a grammar morphism, where $\mathscr{O}_1$ and $\mathscr{O}_2$ are occurrence grammars. Since, by Lemma 6.22, $[f_T]$ is relational we will sometimes confuse it with the corresponding relation between the items of TG_1 and TG_2, namely with

$$\{(x_1, x_2) \mid \exists x \in X_{f_T}.\ f_T^L(x) = x_1 \ \wedge \ f_T^R(x) = x_2\}$$

and we will write $f_T(x_1, x_2)$ to mean that there exists $x \in X_{f_T}$ such that $f_T^L(x) = x_1$ and $f_T^R(x) = x_2$. By definition of grammar morphism such relation preserves the start graph and pre-set, post-set and context of productions of $\mathscr{O}_1$.

Incidentally, observe that, instead, condition (2.d) in the definition of i-net morphism (Definition 4.33) in general does not hold for grammar morphisms. Therefore the construction, described in Section 6.1, which associates to each safe grammar an inhibitor net is not functorial. An example of occurrence grammar morphism not meeting the mentioned condition is reported in Figure 6.5. The morphism relates items (production names or graph items) of the first grammar to items of the second grammar with the same name, (possibly) without "primes". It is not difficult to see that this is a legal grammar morphism, but the inhibitor set of q_2 is not reflected since $L \in {}^{\circledcirc}q_2$, L' and L'' are mapped to L, but ${}^{\circledcirc}q_2'$ contains only L'' (while to satisfy condition (2.d) it should have been ${}^{\circledcirc}q_2' = \{L', L''\}$). The fact that graph grammar morphisms are more liberal than inhibitor net morphisms is intuitively justified by the particular properties of the inhibitor set of productions in graph grammars observed in Remark 6.4.

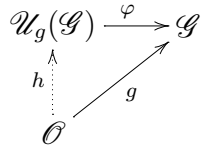
An important property of morphisms between occurrence grammars, which will play a central role in the proof of the coreflection, is the fact that they "preserve" quasi-concurrency. Furthermore quasi-concurrent items cannot be identified by a morphism. This lemma can be proved along the same lines of an analogous result which hold for morphisms of contextual Petri nets. In fact, the notion of quasi-concurrency disregards the dangling condition, taking into account only causality

and asymmetric conflict, two kind of dependencies whose treatment is basically the same for grammars and contextual nets.

Lemma 6.23 (preservation of concurrency)
Let $\mathscr{O}_1$ and $\mathscr{O}_2$ be occurrence grammars, let $f : \mathscr{O}_1 \to \mathscr{O}_2$ be a grammar morphism, and consider, for $i \in \{1, 2\}$, a TG_i-typed graph G_i. If $f_T(G_1, G_2)$ and $t_{G_1}(|G_1|)$ is a quasi-concurrent subgraph of TG_1 then $t_{G_2}(|G_2|)$ is a quasi-concurrent subgraph of TG_2. Furthermore for all $x, y \in t_{G_1}(|G_1|)$ and $z \in t_{G_2}(|G_2|)$, if $f_T(x, z)$ and $f_T(y, z)$ then $x = y$.

Occurrence grammars are particular semi-weighted grammars, thus we can consider the inclusion functor $\mathscr{I}_O : \mathbf{O\text{-}GG} \to \mathbf{SW\text{-}GG}$. The next theorem shows that the unfolding of a grammar $\mathscr{U}_g(\mathscr{G})$ and the folding morphism $\varphi_{\mathscr{G}}$ are cofree over $\mathscr{G}$. Therefore $\mathscr{U}_g$ extends to a functor that is right adjoint of $\mathscr{I}_O$ and thus establishes a coreflection between **SW-GG** and **O-GG**.

Theorem 6.24 (coreflection between **SW-GG** and **O-GG**)
Let $\mathscr{G}$ be a semi-weighted grammar, let $\mathscr{U}_g(\mathscr{G})$ be its unfolding and let $\varphi : \mathscr{U}_g(\mathscr{G}) \to \mathscr{G}$ be the folding morphism as in Definition 6.20. Then for any occurrence grammar $\mathscr{O}$ and for any morphism $g : \mathscr{O} \to \mathscr{G}$ there exists a unique morphism $h : \mathscr{O} \to \mathscr{U}_g(\mathscr{G})$ such that the following diagram commutes:

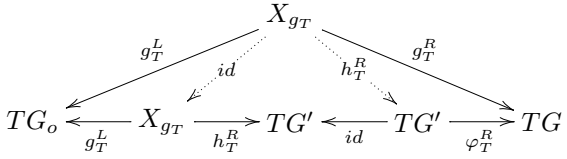


Therefore $\mathscr{I}_O \dashv \mathscr{U}_g$.

Proof (Sketch). To avoid a cumbersome notation, let us fix the names of the components of the various grammars. Let $\mathscr{G} = \langle TG, G_s, P, \pi \rangle$, $\mathscr{U}_g(\mathscr{G}) = \langle TG', G'_s, P', \pi' \rangle$, and let $\mathscr{O} = \langle TG_o, G_{s_o}, P_o, \pi_o \rangle$.

According to Definition 5.33, a morphism $h : \mathscr{O} \to \mathscr{U}_g(\mathscr{G})$ is determined by a semi-abstract span $[h_T] : TG_o \to TG'$, a function $h_P : P_o \to P'$, and a family of morphisms $\iota_h = \{\iota_h(q_o) \mid q_o \in P_o\} \cup \{\iota_h^s\}$ satisfying suitable requirements.

As a first step, we show that both the left component of $[h_T]$ and the family ι_h are uniquely determined by the condition $\varphi \circ h = g$ and by the properties of the folding morphism φ. In fact, let in the following diagram $\langle g_T^L, X_{g_T}, g_T^R \rangle : TG_o \to TG$ be an arbitrary but fixed representative of $[g_T]$, and let $\langle id, TG', \varphi_T^R \rangle : TG' \to TG$ be a representative of $[\varphi_T]$ (where, being φ strong, we can choose the identity of TG' as left component).



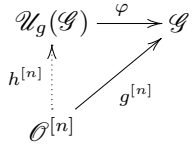
Then it is easily shown that for any semi-abstract span $[h_T] : TG_o \to TG'$ such that $[\varphi_T] \circ [h_T] = [g_T]$ we can choose a representative of the form $\langle g_T^L, X_{g_T}, h_T^R \rangle$ for some h_T^R, because the inner square

becomes a pullback. This shows that, without any loss of generality, we can assume that the left components of $[h_T]$ and $[g_T]$ coincide.

As far as the family of morphisms ι_h is concerned, recall that morphism $\iota^s_\varphi : |G_s| \to |G'_s|$ is the identity by Definition 6.20; thus, since $\varphi \circ h = g$ implies $\iota^s_h \circ \iota^s_\varphi = \iota^s_g$, we deduce that $\iota^s_h = \iota^s_g$. The same holds for the components of $\iota_h(q_o)$ for any production $q_o \in P_o$, by observing that $\iota_h(q_o) \circ \iota_\varphi(h_P(q_o)) = \iota_g(q_o)$ must hold, and that $\iota_\varphi(q')$ is a triple of identities for each $q' \in P'$.

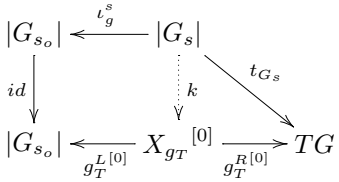
Existence

We will show inductively that for each $n \in \mathbb{N}$ we can find a morphism $h^{[n]}$ such that the diagram

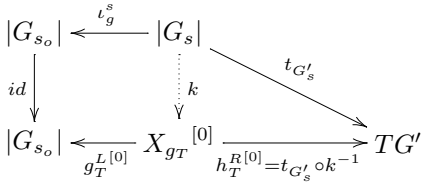


commutes, and such that $h^{[n+1]}$ extends $h^{[n]}$. Then the morphism h we are looking for will be the componentwise union of the chain of morphisms $\{h^{[n]}\}_{n\in\mathbb{N}}$.

($\mathbf{k = 0}$) By definition, the occurrence grammar $\mathscr{O}^{[0]}$ consists of the start graph of $\mathscr{O}$ only, typed identically on the type graph, with no productions, i.e., $\mathscr{O}^{[0]} = \langle |G_{s_o}|, \emptyset, \emptyset \rangle$. By the considerations above, to determine morphism $h^{[0]} : \mathscr{O}^{[0]} \to \mathscr{U}_g(\mathscr{G})$ we only have to provide the right component $h_T^{R[0]} : {X_{g_T}}^{[0]} \to TG'$ of $[{h_T}^{[0]}]$. Moreover, to be a well defined grammar morphism, $h^{[0]}$ must preserve the start graph. By condition (1) of Definition 5.33 applied to $g^{[0]} : \mathscr{O}^{[0]} \to \mathscr{G}$, there is a morphism $k : |G_s| \to {X_{g_T}}^{[0]}$ such that the diagram below commutes, and the square is a pullback. Furthermore, by the pullback properties k is an isomorphism, and, being $\mathscr{G}$ a semi-weighted grammar, by Lemma 6.22, k is uniquely determined.

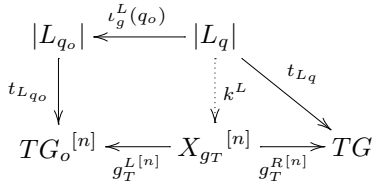


Now we define $h_T^{R[0]} = t_{G_s} \circ k^{-1}$, completing the definition of $h^{[0]}$. The next diagram shows that $h^{[0]}$ satisfies the requirement of preservation of the start graph. The fact that $g^{[0]} = \varphi \circ h^{[0]}$ easily follows by construction.

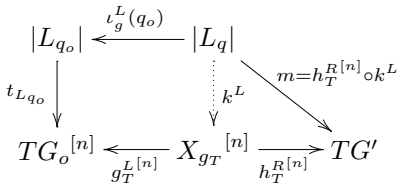


($\mathbf{n \to n{+}1}$) We have to define morphism $h^{[n+1]} : \mathscr{O}^{[n+1]} \to \mathscr{U}_g(\mathscr{G})$ by extending $h^{[n]}$ to the items of $TG_o \cup P_o$ of depth equal to $n+1$. Without any loss of generality we assume that there is just one production q_o in $\mathscr{O}^{[n+1]}$ with $depth(q_o) = n+1$ (the general case can be carried out in a completely analogous way). To ensure $\varphi_P \circ {h_P}^{[n+1]} = {g_P}^{[n+1]}$, the production q_o must be mapped to a production q' in $\mathscr{U}_g(\mathscr{G})$, which is an occurrence of the production $q = g_P(q_o)$ of $\mathscr{G}$. In other words, q' will be $\langle q, m \rangle$, with $m : L_q \to \langle TG', \varphi_T^R \rangle$ a match satisfying suitable conditions.

The defining conditions of grammar morphisms, applied to $g^{[n]} : \mathscr{O}^{[n]} \to \mathscr{G}$, ensure the existence of a morphism k^L, such that the diagram below commutes, where the square is a pullback.



Moreover, being $\mathscr{G}$ a semi-weighted grammar, by Lemma 6.22 g_T is relational and thus the arrow k^L is uniquely determined. By Definition 6.18, $depth(x) \leq n$ for all $x \in t_{L_{q_o}}(|L_{q_o}|) = {}^\bullet q_o \cup \underline{q_o}$, and thus $h^{[n]}$ is defined on the pre-set and on the context of q_o. Therefore we can construct the following diagram.



Notice that $m = h_T^{R[n]} \circ k^L$ can be seen as a TG-typed graph morphism from L_q to $\langle TG', \varphi_T^R \rangle$. In fact, it satisfies $\varphi_T^R \circ m = \varphi_T^R \circ h_T^{R[n]} \circ k^L = g_T^{R[n]} \circ k^L = t_{L_q}$. Moreover, recalling that $h_T{}^{[n]} = \langle g_T^{L\,[n]}, h_T^{R[n]} \rangle$, by the diagram above we have that $h_T{}^{[n]}(L_{q_o}, \langle |L_q|, m \rangle)$. Since by definition of occurrence grammar $t_{L_{q_o}}(|L_{q_o}|)$ is a quasi-concurrent subgraph of TG_o, by using Lemma 6.23, we can conclude that $m(|L_q|)$ is a quasi-concurrent subgraph of TG'. Let us prove that, in addition, the mapping m satisfies the identification condition. First observe that for $x, y \in |L_q|$

$$m(x) = m(y) \quad \Rightarrow \quad k^L(x) = k^L(y). \tag{$\dagger$}$$

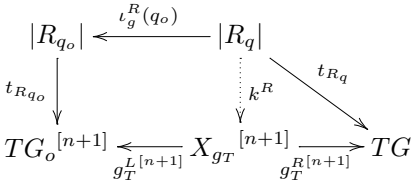
In fact, assume that $m(x) = m(y)$, let $x' = k^L(x)$ and $y' = k^L(y)$ and suppose $x' \neq y'$. From the fact that $m(x) = m(y)$ we deduce $h_T^{R[n]}(x') = h_T^{R[n]}(y')$, and therefore, since $h^{[n]}$ is relational, $g_T^{L[n]}(x') \neq g_T^{L[n]}(y')$. Now observe that, by commutativity of the square in the diagram above, $g_T^{L[n]}(x')$, $g_T^{L[n]}(y') \in t_{L_{q_o}}(|L_{q_o}|)$ and moreover $h_T{}^{[n]}(g_T^{L[n]}(x'), z)$, $h_T{}^{[n]}(g_T^{L[n]}(y'), z)$, where $z = m(x) = m(y)$. But according to Lemma 6.23 this would imply that $t_{L_{q_o}}(|L_{q_o}|)$ is not quasi-concurrent, contradicting the definition of occurrence grammar. Hence, as desired, it must be $k^L(x) = k^L(y)$. We can now conclude that m satisfies the identification condition, namely that for $x, y \in |L_q|$

$$m(x) = m(y) \quad \Rightarrow \quad x, y \in |K_q|.$$

In fact, suppose that $m(x) = m(y)$. By ($\dagger$) above we have that $k^L(x) = k^L(y)$, hence, by general pullback properties, $\iota_g^L(q_0)(x) \neq \iota_g^L(q_0)(y)$ and, by commutativity of the square in the diagram above, $t_{L_{q_o}}(\iota_g^L(q_0)(x)) = t_{L_{q_o}}(\iota_g^L(q_0)(y))$. Recalling that $t_{L_{q_o}}$ satisfies the identification condition we get that $\iota_g^L(q_0)(x)$ and $\iota_g^L(q_0)(y)$ must be in $|K_{q_o}|$, and thus $x, y \in |K_q|$.

Since $m : L_q \to \langle TG', \varphi_T^R \rangle$ is a match satisfying the identification condition and $m(|L_q|)$ is quasi-concurrent, by definition of unfolding $q' = \langle q, m \rangle$ is a production name in P'. Then the production component $h_P{}^{[n+1]}$ of the morphism $h^{[n]}$ can be defined by extending $h_P{}^{[n]}$ with $h_P{}^{[n+1]}(q_o) = q'$. The diagram above shows that, with this extension, the left-hand side of the production is preserved. Now, it can be seen that there is a unique way of extending the type-span

$h_T{}^{[n]}$ to take into account also the right-hand side of production q'. In fact, consider the diagram below expressing, for morphism $g^{[n+1]}$, the preservation of the right-hand side of q_o.



To complete the definition of $h^{[n]}$ we must define the right component $h_T^{R\,[n+1]} : X_{g_T}{}^{[n+1]} \to TG'$, extending $h_T^{R\,[n]}$ on the items which are in $X = X_{g_T}{}^{[n+1]} - X_{g_T}{}^{[n]}$. Now one can verify that k^R establishes an isomorphism between X and $|R_q| - r_q(|K_q|)$. Then the condition requiring that $h_T{}^{[n+1]}$ preserves the right-hand side of q_o forces us to define, for each $x \in X$, $h_T^{R\,[n+1]}(x) = t_{L_{q'}}(k^{R^{-1}}(x))$.

The fact that $g^{[n]} = \varphi \circ h^{[n]}$ easily follows by construction.

Uniqueness
Uniqueness essentially follows from the fact that at each step we are forced to define the morphism h as we have done to ensure commutativity. □

6.4.2 Unfolding of general grammars

A natural question regards the possibility of extending the universal characterization of the unfolding construction to the whole category **GG** of graph grammars. It should be noticed that the proof of the uniqueness of the morphism h in Theorem 6.24 strongly relies on Lemma 6.22 which in turn requires the grammar $\mathscr{G}$ to be semi-weighted. Unfortunately the problem does not reside in our proof technique: the cofreeness of the unfolding of $\mathscr{U}_g(\mathscr{G})$ and of the folding morphism $\varphi_{\mathscr{G}}$ over $\mathscr{G}$ may really fail to hold if the grammar $\mathscr{G}$ is not semi-weighted.

For instance, consider grammars $\mathscr{G}_1$ and $\mathscr{G}_2$ in Figure 6.6, where typed graphs are represented by decorating their items with pairs "concrete identity:type". The grammar $\mathscr{G}_2$ is not semi-weighted since the start graph is not injectively typed, while $\mathscr{G}_1$ is clearly an occurrence grammar. The unfolding $\mathscr{U}_g(\mathscr{G}_2)$ of the grammar $\mathscr{G}_2$, according to Definition 6.20, is defined as follows. The start graph and type graph of $\mathscr{U}_g(\mathscr{G}_2)$ coincide with $|G_{s_2}|$. Furthermore, $\mathscr{U}_g(\mathscr{G}_2)$ contains two productions $q_2' = \langle q_2, m' \rangle$ and $q_2'' = \langle q_2, m'' \rangle$, which are two occurrences of q_2 corresponding to the two possible different matches $m', m'' : L_{q_2} \to G_{s_2}$ (the identity and the swap).

Observe that there exists a morphism $g : \mathscr{G}_1 \to \mathscr{G}_2$ which is not relational, i.e., the property in Lemma 6.22 fails to hold. The component g_P on productions is defined by $g_P(q_1) = q_2$, while the type span g_T is defined as follows: X_{g_T} is a discrete graph with two nodes x and y, $g_T^L(x) = g_T^L(y) = A$ and $g_T^L(x) = g_T^L(y) = B$ (see the bottom row of the diagram in Figure 6.6). Consider the pullback-retyping diagram in Figure 6.6, expressing the preservation of the start graph for morphism g (condition (1) of Definition 5.33). Notice that there are two possible different morphisms k and k'

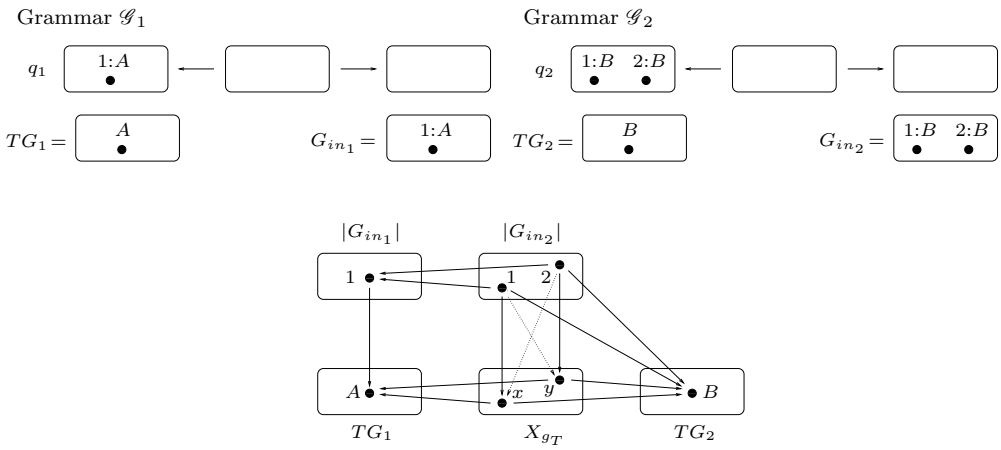
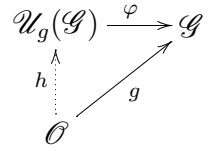


Figure 6.6: The grammars $\mathscr{G}_1$ and $\mathscr{G}_2$, and the pullback-retyping diagram for their start graphs.

from $|G_{s_2}|$ to X_{g_T} (represented via plain and dotted arrows, respectively) making the diagram commutes and the square a pullback. Now, it is not difficult to see that, correspondingly, we can construct two different morphisms $h_i : \mathscr{G}_1 \to \mathscr{U}_g(\mathscr{G}_2)$ ($i \in \{1, 2\}$), such that $\varphi_{\mathscr{G}_2} \circ h_i = g$, the first one mapping production q_1 into q_2' and the second one mapping q_1 into q_2''. An immediate consequence of this fact is the impossibility of extending $\mathscr{U}_g$ on morphisms, in order to obtain a functor which is right adjoint to the inclusion $\mathscr{I} : \mathbf{O\text{-}GG} \to \mathbf{GG}$.

The above considerations, besides giving a negative result, also suggest a way to overcome the problem. An inspection of the proof of Theorem 6.24 reveals that the only difficulty which prevents us to extend the result is the non uniqueness of the morphisms k, k^L, k^K and k^R in the pullback-retyping constructions. In other words, if we consider any morphism $g : \mathscr{O} \to \mathscr{G}$ such that $[g_T]$ is *relational* then we can prove, as in Theorem 6.24, the existence of a unique morphism $h : \mathscr{O} \to \mathscr{U}_g(\mathscr{G})$ making the following diagram commute:



Hence the coreflection result can be regained by limiting our attention to a (non full) subcategory $\widehat{\mathbf{GG}}$ of $\mathbf{GG}$, where objects are general graph grammars, but all morphisms have a relational span as type component. Then, the only thing to prove is that the unique morphism h constructed in the proof of Theorem 6.24 is indeed an arrow in $\widehat{\mathbf{GG}}$.

The naïve solution of taking *all* relational morphisms as arrows of $\widehat{\mathbf{GG}}$ does not work because they are not closed under composition. A possible appropriate choice is instead given by the category $\mathbf{GG}^R$, where the arrows are grammar morphisms f such that the left component f_T^L of the type span is mono.

DEFINITION 6.25 (CATEGORY $\mathbf{GG}^R$)
We denote by $\mathbf{GG}^R$ the lluf subcategory of $\mathbf{GG}$, where for any arrow f the left component f_T^L of the type span is mono. Furthermore we denote by $\mathbf{O}$-$\mathbf{GG}^R$ the full subcategory of $\mathbf{GG}^R$ having occurrence grammars as objects.

By the properties of pullbacks, the arrows in $\mathbf{GG}^R$ are closed under composition and thus $\mathbf{GG}^R$ is a well-defined subcategory of $\mathbf{GG}$.

THEOREM 6.26 (UNFOLDING AS COREFLECTION - REPRISE)
The unfolding construction can be turned into a functor $\mathscr{U}_g^R : \mathbf{GG}^R \to \mathbf{O}\text{-}\mathbf{GG}^R$, having the inclusion $\mathscr{I}_O^R : \mathbf{O}\text{-}\mathbf{GG}^R \to \mathbf{GG}^R$ as left adjoint, establishing a coreflection between the two categories.

PROOF. By the considerations above, the only thing to prove is that the morphism h constructed as in the proof of Theorem 6.24 is an arrow in $\mathbf{O}$-$\mathbf{GG}^R$. But this is obvious since, by construction $h_T^L = g_T^L$ and thus h_T^L is mono. □

Alternatively, the result can be proved for the subcategory $\mathbf{GG}^L$ of $\mathbf{GG}$ where arrows are grammar morphisms having the *right* component of the type span which is mono. Clearly this is a well defined subcategory, while proving that the morphism constructed in the proof of Theorem 6.24 is an arrow in $\mathbf{O}$-$\mathbf{GG}^L$ requires some additional effort.

Observe that although not completely general, the above results regard a remarkable class of grammar morphisms. In particular they comprise the morphisms adopted in [HCEL96, Rib96] where the type component of an arrow from $\mathscr{G}_1$ to $\mathscr{G}_2$ is a partial graph morphism from TG_1 to TG_2, and from TG_2 to TG_1, respectively.

Finally, it is worth remarking that strong grammar morphisms are arrows in $\mathbf{GG}^R$, a fact that, in the next chapter, will allow us to fruitfully use Theorem 6.26 to establish a relation between the unfolding and the deterministic process semantics of a grammar (see Section 7.4).

6.5 From occurrence grammars to event structures

Starting from the semantics of graph grammars given in terms of occurrence grammars, the aim of this section is to provide a more abstract semantics based on (suitable kind of) event structures and domains, following the guidelines traced in the FIRST PART for contextual and inhibitor nets. Due to the similarity between grammars and inhibitor nets, it comes as no surprise that inhibitor event structures,

the generalization of Winskel's event structures introduced in CHAPTER 4, are expressive enough to encode the causal structure of grammar computations. The IES associated to an occurrence grammar is obtained by forgetting the state, and remembering the productions of the grammar and the relationship among them. Recalling the encoding of safe grammars into i-nets, the formal definition is a straightforward generalization of the construction taking an occurrence i-net into the corresponding IES. By using the results in CHAPTER 4 the IES semantics can be finally "translated" into a domain and PES semantics.

The configurations of the IES associated to an occurrence grammar $\mathscr{O}$ can be shown to coincide with the configurations of the grammar $\mathscr{O}$, as defined in Section 6.1. Furthermore the extension ordering on the configurations of such IES can be characterized by using only the relations of causality and asymmetric conflict defined directly on the grammar, thus providing a simpler characterization of the domain semantics.

6.5.1 An IES semantics for graph grammars

We next introduce the DE-relation naturally associated to an occurrence grammar $\mathscr{O}$, which is is used to define first a pre-IES and then an IES for the grammar $\mathscr{O}$.

DEFINITION 6.27 (PRE-IES FOR AN OCCURRENCE GRAMMAR)
*Let $\mathscr{O} = \langle TG, P, \pi \rangle$ be an occurrence grammar. The pre-*IES *associated to $\mathscr{O}$ is defined as $I^p_{\mathscr{O}} = \langle P, \mapsto^p_{\mathscr{O}} \rangle$, with $\mapsto_{\mathscr{O}} \subseteq \mathbf{2}^P_1 \times P \times \mathbf{2}^P$, given by: for $q, q' \in P$ and $x \in Items(TG)$*

- *if $q^\bullet \cap ({}^\bullet q' \cup \underline{q'}) \neq \emptyset$ then $\mapsto^p_{\mathscr{O}}(\emptyset, q', \{q\})$*
- *if $({}^\bullet q \cup \underline{q}) \cap {}^\bullet q' \neq \emptyset$ then $\mapsto^p_{\mathscr{O}}(\{q'\}, q, \emptyset)$;*
- *if $x \in {}^{\circledcirc}q$ (and thus x is an edge in the type graph TG) then $\mapsto^p_{\mathscr{O}}({}^\bullet s, q, s^\bullet)$.*

As for inhibitor nets, $I^p_{\mathscr{O}}$ is a pre-IES satisfying also condition (1) of the definition of IES. Therefore, as discussed in Proposition 4.5, it can be "saturated" in order to obtain an IES.

DEFINITION 6.28 (IES FOR AN OCCURRENCE GRAMMAR)
The IES *associated to an occurrence grammar $\mathscr{O}$, denoted by $I_{\mathscr{O}} = \langle P, \mapsto_{\mathscr{O}} \rangle$, is defined as $\overline{I^p_{\mathscr{O}}}$.*

Recall that the causality, asymmetric conflict and conflict relations of $I_{\mathscr{O}}$ and $I^p_{\mathscr{O}}$ coincide. They will be denoted by $<^i_{\mathscr{O}}$, $\nearrow^i_{\mathscr{O}}$ and $\#^i_{\mathscr{O}}$, respectively. The superscript "i" is used to distinguish these relations from the relations $<_{\mathscr{O}}$ and $\nearrow_{\mathscr{O}}$, associated directly to the occurrence grammar in Section 6.1.

We now prove that the construction which maps an occurrence grammar into the corresponding IES $I_{\mathscr{O}}$ can be turn into a functor. This is not completely obvious, since, as observed in Section 6.4, grammar morphisms are more liberal than i-net

morphisms because, in general, they do not reflect the inhibitor set of productions. In the proof we will exploit the fact that, as explained in Section 6.4, the type component of a morphism between occurrence grammars can be thought of as a relation, preserving the start graph as well as the pre-set, post-set and context of productions.

PROPOSITION 6.29
Let $\mathscr{O}_0$ and $\mathscr{O}_1$ be occurrence grammars and let $h : \mathscr{O}_0 \to \mathscr{O}_1$ be a grammar morphism. Then $h_P : I_{\mathscr{O}_0} \to I_{\mathscr{O}_1}$ is an IES *morphism.*

PROOF (SKETCH). For $k \in \{0,1\}$, let $<_k$, $\nearrow_k$ and $\#_k$ be the relations of causality, asymmetric conflict and conflict in the pre-IES $I^p_k = \langle P_k, \mapsto^p_k \rangle$ associated to the grammar $\mathscr{O}_k$. As in the case of i-nets, we show that $h_P : I^p_0 \to I^p_1$ satisfies conditions (1)-(4) in the hypotheses of Lemma 4.9 and thus that h_P is an IES morphism between the corresponding "saturated" IES's.

Relying on the analogy between occurrence grammar and i-nets morphisms, most of the properties can be proved exactly as for i-nets. We will treat explicitly only only some cases which involve the inhibitor set of productions, since, as observed before, grammar morphisms impose on the inhibitor set of productions requirements which are weaker than those of i-nets morphisms.

1. $h_P(q_0) = h_P(q'_0) \ \wedge \ q_0 \neq q'_0 \quad \Rightarrow \quad q_0 \#_0 q'_0$.
   Treated as for i-nets.

2. $\mapsto^p_1(\emptyset, h_P(q_0), A_1) \quad \Rightarrow \quad \exists A_0 \subseteq h_P^{-1}(A_1).\ A_0 <_0 q_0$.
   Let us assume $\mapsto^p_1(\emptyset, h_P(q_0), A_1)$. By definition of $\mapsto^p_1$ one of the following holds:

   (a) $A_1 = \{q_1\}$ and $q_1{}^\bullet \cap {}^\bullet h(q_0) \neq \emptyset$

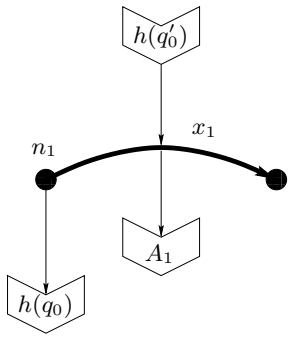
   (b) $A_1 = \{q_1\}$ and $q_1{}^\bullet \cap \underline{h(q'_0)} \neq \emptyset$

   (c) $\exists x_1 \in {}^{\circledcirc}h(q_0).\ {}^\bullet x_1 = \emptyset \ \wedge \ x_1{}^\bullet = A_1$,

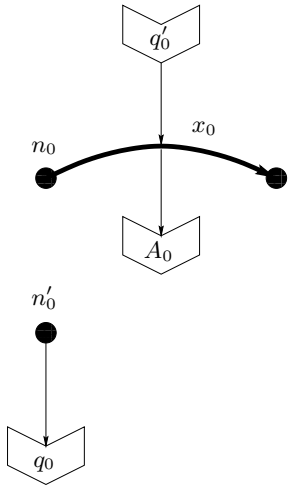
   The first two cases are treated as for i-nets. In case (c), observe that x_1 is necessarily an edge in the start graph of $\mathscr{O}_1$, with source or target in a node n_1 which must be in the start graph of $\mathscr{O}_1$ as well. By the properties of occurrence grammar morphisms, there are a unique node n_0 and a unique edge x_0 such that $h_T(n_0, n_1)$ and $h_T(x_0, x_1)$. Then $n_0 \in {}^\bullet q_0$, $x_0{}^\bullet <_0 t_0$ and $x_0{}^\bullet \subseteq h_P^{-1}(A_1)$.

3. $\mapsto^p_1(h_P(q'_0), h_P(q_0), \emptyset) \quad \Rightarrow \quad q_0 \nearrow_0 q'_0$.
   Treated as for i-nets.

4. $\mapsto^p_1(\{h_P(q'_0)\}, h_P(q_0), A_1) \wedge A_1 \neq \emptyset \quad \Rightarrow \quad \exists a_0 \subseteq \{q'_0\}.\ \exists A_0 \subseteq h_P^{-1}(A_1).\ \mapsto_0(a_0, q_0, A_0)$.
   Assume $\mapsto^p_1(\{h_P(q'_0)\}, h_P(q_0), A_1)$ and $A_1 \neq \emptyset$. Thus, by definition of $\mapsto^p_1$ there must be an edge $x_1 \in {}^{\circledcirc}h_P(q_0) \cap h_P(q'_0)^\bullet$ such that $A_1 = {}^\bullet x_1$. The figure below gives a graphical representation of the situation, in which the set of productions A_1 is depicted as a single box.



By properties of occurrence grammar morphisms, we can find in the post-set of q_0' an edge x_0 with source or target in the node n_0, and in the pre-set of q_0 a node n_0' such that the type component h_T relates n_0 and n_0' to n_1, while x_0 is related to x_1.



Now, two possibilities arise: if $n_0 = n_0'$ then we immediately conclude that $\vdash_0^p(\{q_0'\}, q_0, x_0^\bullet)$ and clearly $x_0^\bullet \subseteq h_P^{-1}(A_1)$.

If instead $n_0 \neq n_0'$, first observe that n_0 and n_0' are not in the start graph. In fact, suppose that n_0 is in the start graph. Then, by definition of grammar morphism, also n_1 is in the start graph and therefore one can easily conclude that also n_0' is in the start graph. Hence $\{n_0, n_0'\}$ is quasi concurrent, but $h_T(n_0, n_1)$ and $h_T(n_0', n_1)$, which is a contradiction by Lemma 6.23.

Now, considering $q_0'' \in {}^\bullet n_0$ and $q_0''' \in {}^\bullet n_0'$, we have $q_0'' \#_0 q_0'''$. Since q_0' produces the edge x_0, necessarily $n_0 \in q_0'^\bullet \cup \underline{q_0'}$ and therefore $q_0'' \leq_0 q_0'$. Furthermore, $q_0''' \leq_0 q_0$ and therefore $q_0 \#_0 q_0'$, which implies $q_0 \overline{\nearrow}_0 q_0'$. Recalling how a pre-IES is saturated to give rise to an IES (see Proposition 4.5) we finally conclude $\vdash_0(\{q_0'\}, q_0, \emptyset)$. □

By the above proposition the functor which maps each occurrence grammar to the corresponding IES defined as in Definition 6.28 and each grammar morphism to its production component is well-defined.

DEFINITION 6.30 (FROM OCCURRENCE GRAMMARS TO IES'S)
Let $\mathcal{E}_g$: **O-GG** $\rightarrow$ **IES** *be the functor defined as:*

- $\mathcal{E}_g(\mathcal{O}) = I_{\mathcal{O}}$, *for each occurrence grammar* $\mathcal{O}$;
- $\mathcal{E}_g(h : \mathcal{O}_0 \rightarrow \mathcal{O}_1) = h_P$, *for each morphism* $h : \mathcal{O}_0 \rightarrow \mathcal{O}_1$.

Now, by using the functor $\mathcal{L}_i$: **IES** $\rightarrow$ **Dom** defined in CHAPTER 4, mapping each IES into its domain of configurations we can obtain a domain and then a PES semantics for grammars.

6.5.2 A simpler characterization of the domain semantics

In Section 6.1 we have introduced the configurations of nondeterministic occurrence grammars as a mean to capture the possible deterministic computations of such grammars. We next show that they coincide with the configurations of the associated inhibitor event structure, thus allowing for a simpler characterization of the corresponding domain.

We first observe a property of the IES associated to an occurrence grammar, which generalizes the observation in Remark 6.4. Recall that in general IES's when, for instance, $\vdash\!\circ(\{q'\}, q, \{q''\})$, if the three events q, q', q'' appear in the same computation then there are two possible orders of execution, namely $q; q'; q''$ and $q'; q''; q$. Instead, in the case of occurrence grammars $\mathcal{O}$, if $\vdash\!\circ_{\mathcal{O}}(\{q'\}, q, \{q''\})$ then by definition of $\vdash\!\circ_{\mathcal{O}}$, $q'^{\bullet} \cap {}^{\circ}q \neq \emptyset$. Therefore, as observed in Remark 6.4, necessarily q' must precede q, i.e., $q' \nearrow q$, since q' produces or preserves a node which is consumed by q. Hence, in this case, the only possible order of execution is $q'; q''; q$. As a consequence, for each configuration C the associated choice relation $\hookrightarrow_C$ is uniquely determined by the events of the configuration and coincides with the asymmetric conflict.

PROPOSITION 6.31
Let $\mathcal{O} = \langle TG, P, \pi \rangle$ *be an occurrence grammar and let* $\mathcal{E}_i(\mathcal{O}) = \langle P, \vdash\!\circ_{\mathcal{O}} \rangle$ *be the corresponding* IES. *Then*

- $\vdash\!\circ_{\mathcal{O}}(\{q'\}, q, A) \quad \Rightarrow \quad q' \nearrow^i q \ \wedge \ \forall q'' \in A.\ q'' \nearrow^i q$;
- $\langle C, \hookrightarrow \rangle \in Conf(I) \quad \Rightarrow \quad \hookrightarrow = \nearrow^i_C$.

It is easy to realize that the relations $<_{\mathcal{O}}$ and $\nearrow_{\mathcal{O}}$, which do not take into account the dependencies induced by the dangling condition, are included in the corresponding relations $<^i_{\mathcal{O}}$ and $\nearrow^i_{\mathcal{O}}$. The above proposition entails that the transitive closures of the two asymmetric conflict relations coincide, namely

$$(\nearrow_{\mathcal{O}})^* = (\nearrow^i_{\mathcal{O}})^*$$

and thus, when restricted to a configuration, they express essentially the same precedences. This observation allows one to prove that the configurations of an occurrence grammar are exactly the configurations of the associated IES. Furthermore

the extension order on the IES's configurations can be characterized by using only the asymmetric conflict relation $\nearrow_{\mathscr{O}}$ associated to the grammar. Besides providing a simpler characterization of the domain semantics of a grammar, this result enforces our confidence in the appropriateness of the IES associated to an occurrence grammar, since, as we already observed, the configurations of an occurrence grammar correctly represents all the different possible deterministic computations of the grammar.

Proposition 6.32 (domain for an occurrence grammar)
Let $\mathscr{O}$ be an occurrence grammar and let $\mathscr{E}_g(O)$ be the corresponding IES. *Then $C \in Conf(\mathscr{O})$ iff $C \in \mathscr{E}_g(O)$. Moreover the order relation over the prime algebraic domain $\mathscr{L}_i(\mathscr{E}_g(O))$ can be characterized as*

$$C \sqsubseteq C' \quad \textit{iff} \quad C \subseteq C' \textit{ and } \forall q \in C.\ \forall q' \in C'.\ q' \nearrow_{\mathscr{O}} q \ \Rightarrow\ q' \in C.$$

Notice that, formally, the order on configuration is the same as for contextual nets: a configuration C cannot be extended with a production inhibited by some of the productions already present in C. However recall from Definition 6.9, that the asymmetric conflict by itself, without any additional information on the dangling condition, is not sufficient to define which subsets of events are configurations.

Hereafter we will always use the above characterization of the domain semantics of a grammar. Consequently we will never refer to the asymmetric conflict, conflict and causality relations of the IES $I_{\mathscr{O}}$, using instead the relations defined directly on the grammar $\mathscr{O}$.

Recall that the PES associated to the domain $\mathscr{L}_i(\mathscr{E}_g(\mathscr{O}))$, namely $\mathscr{P}(\mathscr{L}_i(\mathscr{E}_g(\mathscr{O}))))$, consists of the set of prime elements of the domain (with the induced partial order as causality and the inconsistency relation as conflict), i.e., the unique (up to isomorphisms) PES having $\mathscr{L}_i(\mathscr{E}_g(\mathscr{O}))$ as domain of configurations.

As already seen for contextual and inhibitor nets, there is not a one to one correspondence between events of $\mathscr{P}(\mathscr{L}_i(\mathscr{E}_g(\mathscr{O})))$ and productions in $\mathscr{O}$: a different event is generated for any possible "history" of each production in $\mathscr{O}$. This phenomenon of "duplication of events" is related to the fact that the PES represents the dependencies arising between productions in graph grammar computations by means of causality and symmetric conflict. A situation of asymmetric conflict like $q_1 \nearrow q_2$ in grammar $\mathscr{G}_1$ of Figure 6.1, is encoded in the PES by the insertion of a single event e_1 corresponding to q_1, and two "copies" e_2' ad e_2'' of q_2, the first one in conflict with e_1 and the second one caused by e_1 (see Figure 6.7.(a)). For what concerns the dangling condition, consider the grammar $\mathscr{G}_2$ in Figure 6.1. In this case three conflicting events are generated corresponding to q_4: e_4 representing the execution of q_4 from the start graph, which inhibits all other productions, and e_4', e_4'' representing the execution of q_4 after q_2 and q_3, respectively.

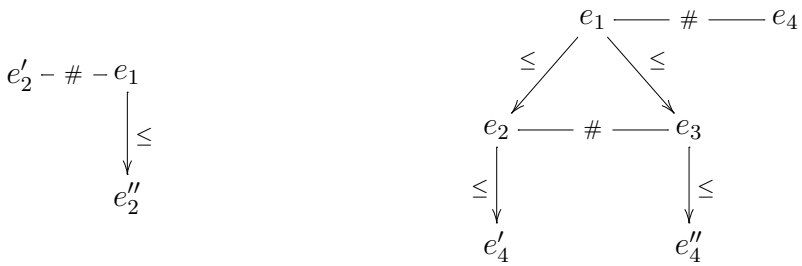


Figure 6.7: Encoding asymmetric conflict and dangling condition in prime event structures.

6.6 A related approach: unfolding semantics of SPO graph grammars

An unfolding construction for graph grammars in the *single-pushout* (SPO) approach has been proposed by Ribeiro in her doctoral thesis [Rib96]. She defines an unfolding functor from a category of SPO graph grammars to a category of (abstract) *occurrence grammars*, showing that it is a right adjoint to a suitable *folding functor*. In this section we discuss the relation between the construction introduced in this chapter and the one in [Rib96]. Although the two constructions rely on similar basic ideas, concretely, the differences between the two settings makes hard a formal direct comparison. Therefore we will mainly try to point out some conceptual differences and analogies.

First of all in the SPO approach there are no application conditions. Without getting into technical details, if a production specifies both the preservation and the deletion of an item, then such an item is removed by the application of the production. Furthermore, an edge which should remain dangling after the application of a production because its source (or target) node is deleted, is automatically deleted as well. For the presence of these "side-effects" the pre-set of a production, intended as the set of items which are "consumed" by the production, depends also on the considered match, and the application of a production may remove items which are not mentioned explicitly in the production itself. Consequently, the notion of causality for SPO graph grammars is less intuitive if compared with the DPO case, and the correspondence between the theory of SPO grammars and the theory of Petri nets becomes looser. For instance, if a production q_1 produces and edge e, and q_2 deletes e as side effect, we may wonder whether q_2 should causally depend on q_1 or not. If we assume that causality is induced by the flow of data in the system, the answer should be that $q_1 \leq q_2$ since q_2 consumes something which is produces by q_1. On the other hand, the application of q_1 is not necessary to make q_2 applicable, since q_2 does not explicitly require the presence of resources generated by q_1. Thus, if $q_1 \leq q_2$ means that the application of q_1 precedes q_2 in each computation where q_2

is applied, then there should be no causal relationship between q_1 and q_2.

Although the above question is not completely trivial, the more reasonable choice seems the latter. In this case, the notions of causality and of asymmetric conflict (called *weak conflict* in [Rib96]) can be defined disregarding the mentioned side-effects. This is basically due to the fact that when a production uses (consumes or preserves) an edge, it must use necessarily the corresponding source and target nodes, and therefore dependencies related to side-effects can be detected by looking only at explicitly used items.

A relevant difference with our work resides in the notion of *occurrence grammars*. The thesis [Rib96] introduces the so-called *doubly typed occurrence grammars*, where the type graph is itself typed over another graph of types. Intuitively, a doubly typed occurrence grammar is a sort of process where only the "type component" of the process morphism is specified. This essentially means that each production of such a grammar can be thought of as an instance of a production of another grammar, for which the match is already specified. Because of the double typing, the category of occurrence grammars considered in [Rib96] is not a subcategory of the category of graph grammars, and the left adjoint of the unfolding functor is not the inclusion, but a folding functor which basically forgets the intermediate typing.

The grammar which is obtained from a doubly-typed occurrence grammar by forgetting the intermediate typing satisfies conditions which are similar to those imposed on our occurrence grammars. However, again the theory necessarily diverges because of the absence of the application condition in the SPO approach. When a single rule requires both the preservation and the deletion of an item, according to our definition it should be in asymmetric conflict with itself, and thus never applicable. As mentioned above, instead in the SPO approach such kind of rule can be applied and the "conflictual requirement" is resolved in favor of deletion.

The final difference which must be taken into account regards the kind of morphisms on graph grammars. In [Rib96] the type component of a morphism from a grammar $\mathscr{G}_1$ to $\mathscr{G}_2$ is a partial function from TG_2 to TG_1. Since such partial function can be seen as a span where the right component is injective, in this respect the morphisms of [Rib96] are less general than ours. On the other hand, to characterize the parallel composition of graph grammars as a categorical limit, the image under a morphism of a production in $\mathscr{G}_1$ is required only to be a sub-production of the corresponding production in $\mathscr{G}_2$. When moving to occurrence grammars, also the morphisms of [Rib96] must preserve the components of a production. However preservation is required only up to isomorphism and this seems the reason why, at a first glance, the unfolding cannot be turned into a functor. The problem is solved by considering a category of abstract occurrence grammars, where objects are isomorphisms classes of grammars. Then the unfolding can be expressed as a functor which establishes an adjunction between the category of graph grammars and such category of abstract occurrence grammars.

Summing up, many analogies exist between the two constructions, but the subtle differences deeply connected with the diversity of the SPO and DPO approaches,

makes us skeptical about the possibility of establishing a formal link between the two results. A possibility still not explored could be to resort to the formulation of the DPO approach in term of SPO approach (with application conditions) to see if a common theory can be developed, although in the past such an approach has revealed some limits.

Chapter 7

Concatenable Graph Processes

This chapter introduces a semantics for DPO graph grammars based on *concatenable graph processes*. Conceptually, concatenable graph processes are very close to derivation traces in that they are aimed at representing the deterministic (truly concurrent) computations of a grammar. However, differently from traces they provide an explicit characterization of the events occurring in computations and of the dependency relationships among them.

Relying on the theory developed in the previous chapter, first a *deterministic graph process* is naturally defined as a special graph process such that the whole set of productions of the underlying occurrence grammar is a configuration, and thus all the productions of the process can be applied in a single computation. Given a deterministic process $\varphi : \mathscr{O} \to \mathscr{G}$, the morphism φ can be used to map derivations in the underlying occurrence grammar $\mathscr{O}$ to corresponding derivations in $\mathscr{G}$. The basic property of a graph process is that the derivations in $\mathscr{G}$ which are in the range of such mapping constitute a full class of *shift*-equivalent derivations. Therefore the process can be regarded as an abstract representation of such a class and plays a rôle similar to a *canonical derivation* [Kre77]. Although obtained by following a different path, the above notion of deterministic process can be seen easily to coincide with the previous proposals in [CMR96, BCM98a].

Graph processes are not naturally endowed with a notion of sequential composition, essentially because of the same problem described for abstract derivations: if the target graph of a process is isomorphic to the source graph of a second process, the naïve idea of composing the two processes by gluing the two graphs according to an isomorphism does not work. In fact, in general we can find several distinct isomorphisms relating two graphs, which may induce sequential compositions of the two processes which substantially differ from the point of view of causality. Using the same technique adopted for derivations, *concatenable graph processes* are defined as graph processes enriched with the additional information (a decoration of the source and target graphs) needed to concatenate them.

Then the claim that deterministic processes and derivation traces give conceptually equivalent descriptions of the system is formalized by showing that the category

$\mathbf{Tr}[\mathscr{G}]$ of concatenable (parallel) derivation traces (Definition 5.27) is isomorphic to the category of concatenable (abstract) processes $\mathbf{CP}[\mathscr{G}]$.

Finally, we show that the unfolding and the deterministic process semantics can be reconciled by following the approach described in the INTRODUCTION. As already done for generalized nets, we prove that the domain associated to a grammar via the unfolding construction can be characterized as the collection of deterministic processes starting from the initial graph, endowed with a kind of prefix order.

The rest of the chapter is structured as follows. Section 7.1 introduces deterministic graph processes as special nondeterministic processes, giving evidence of their relation with the previous notions in the literature. Section 7.2 presents the category $\mathbf{CP}[\mathscr{G}]$ of concatenable processes for a grammar $\mathscr{G}$. The equivalence between the concatenable trace semantics (introduced in Section 5.2) and the concatenable process semantics is proved in Section 7.3. Finally, Section 7.4 provides a connection between the unfolding and the deterministic process semantics of a grammar.

7.1 Deterministic graph processes

A nondeterministic graph process (Definition 6.13) represents a class of derivations of a grammar $\mathscr{G}$ by means of a nondeterministic occurrence grammar $\mathscr{O}$. A (special kind of) grammar morphism from $\mathscr{O}$ to the given grammar $\mathscr{G}$ allows one to map computations in $\mathscr{O}$ to computations in $\mathscr{G}$.

To obtain a *deterministic* graph process, namely an appropriate representative of a *single* concurrent computation in $\mathscr{G}$ we must restrict our attention to occurrence grammars where all productions can be applied in a single computation. By Proposition 6.11 we know that this is the case when the whole set of productions of $\mathscr{O}$ is a configuration of $\mathscr{O}$ itself.

DEFINITION 7.1 (DETERMINISTIC OCCURRENCE GRAMMAR)
An occurrence grammar $\mathscr{O} = \langle TG, P, \pi \rangle$ is called deterministic *if the set P of its productions is a configuration of $\mathscr{O}$. For a deterministic occurrence grammar $\mathscr{O}$ we denote by $Max(\mathscr{O})$ the set of items of the type graph TG which are maximal with respect to causality.*

Recalling the notion of configuration of an occurrence grammar (Definition 6.9), we see that an occurrence grammar is deterministic if the transitive closure of the asymmetric conflict relation is a finitary partial order. Furthermore whenever a production consumes a node in the type graph, then productions removing the edges with source or target in that node must be present as well. Observe that the first condition implies the absence of forward conflicts, namely each item of the type graph can be consumed by at most one production, i.e., $|x^\bullet| \leq 1$ for each item x of the type graph (while in general general occurrence grammars only backward conflicts are forbidden).

In the case of finite grammars the above conditions can be expressed in a simpler way. Since there are only finitely many productions, the first condition amounts to the acyclicity of asymmetric conflict. Assuming the first condition, and thus the absence of forward conflicts, the second condition can be expressed by simply requiring $Max(\mathscr{O})$ to be a well-defined graph.

PROPOSITION 7.2 (FINITE DETERMINISTIC OCCURRENCE GRAMMAR)
A finite occurrence grammar $\mathscr{O}$ is deterministic iff

1. *$\nearrow_{\mathscr{O}}$ is acyclic;*
2. *$Max(\mathscr{O})$ is a well-defined subgraph of the type graph.*

PROOF. Let $\mathscr{O}$ be an occurrence grammar satisfying (1) and (2) above. We must show that its set of productions P is a configuration of $\mathscr{O}$. The fact that $(\nearrow_{\mathscr{O}})^*$ is a finitary partial order immediately follows from the acyclicity of $\nearrow_{\mathscr{O}}$ and the fact that P is finite. Trivially P is closed with respect to causality. It remains to prove the last condition in the definition of configuration, namely that given any edge $e \in TG$, if ${}^{\circledcirc}e \cap P \neq \emptyset$ and ${}^{\bullet}e \subseteq P$ then $e^{\bullet} \cap P \neq \emptyset$. Suppose that $q \in {}^{\circledcirc}e \cap P$ for some edge e (clearly ${}^{\bullet}e \subseteq P$ since P contains all productions of $\mathscr{O}$). Hence there is a node $n \in {}^{\bullet}q$ which is the source or target of e. Therefore $n \leq q$ and thus $n \notin Max(\mathscr{O})$. Since $Max(\mathscr{O})$ is a well-defined graph, necessarily $e \notin Max(\mathscr{O})$ and thus there must exist $q' \in P$ such that $e \in {}^{\bullet}q'$, or equivalently $q' \in e^{\bullet}$.

The converse implication is proved in an analogous way. □

The above proposition, besides giving an alternative simpler characterization of deterministic occurrence grammars, shows that they coincide with the occurrence grammars of [CMR96, BCM98a, BCE+99]. It is worth remarking that in such papers, which are concerned only with deterministic computations, the causality relation is used to represent the order of execution within the specific concurrent computation, and as such it plays the role which is played here by the transitive closure of the asymmetric conflict relation. We apologize if the reader may get a bit confused by this change in terminology, but we hope she will agree that in this general context the terminology we adopted is more appropriate.

Let $\mathscr{O}$ be a deterministic occurrence grammar. Recall that, by Proposition 6.11, for any finite configuration C of $\mathscr{O}$, there is a derivation $Min(\mathscr{O}) \Rightarrow_C^* reach(C)$ applying all the productions of C in any order compatible with the asymmetric conflict. In particular, if the grammar is finite, we can take as C the full set P of productions of $\mathscr{O}$. It is easy to verify that $reach(P) = Max(\mathscr{O})$ and therefore

$$Min(\mathscr{O}) \Rightarrow_P^* Max(\mathscr{O})$$

using all productions in P exactly once, in any order consistent with $\nearrow_{\mathscr{O}}$.

This helps in understanding why a deterministic graph process of a grammar $\mathscr{G}$, that we are going to define simply as a graph process having a deterministic underlying occurrence grammar, can be seen as representative of a set of derivations of $\mathscr{G}$ where independent steps may be switched.

Observe that a deterministic occurrence grammar is a fully fledged graph grammar and thus one may "execute" it by considering any possible derivation beginning

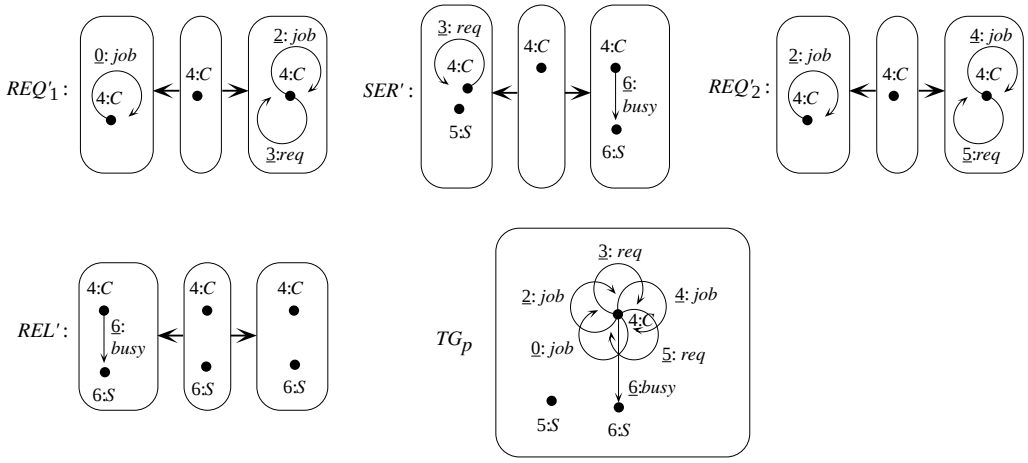


Figure 7.1: A graph process of the grammar $\mathscr{C}$-$\mathscr{S}$.

from the start graph $Min(\mathscr{O})$. However, the discussion above stresses that, if we are interested in using a deterministic occurrence grammar to represent a deterministic computation, it is natural to consider only those derivations which are consistent with the asymmetric conflict.

DEFINITION 7.3 (DETERMINISTIC GRAPH PROCESS)
*Let $\mathscr{G}$ be a graph grammar. A graph process $\varphi : \mathscr{O}_\varphi \to \mathscr{G}$ of $\mathscr{G}$ is called (*finite*)* deterministic *if the the occurrence grammar $\mathscr{O}_\varphi$ is (finite) deterministic.*

As for general nondeterministic processes, we can assume that the left component of the type span φ_T is the identity; furthermore to simplify the notation we will simply write φ_T to denote the right component φ_T^R of the type span. We will denote by $Min(\varphi)$ and $Max(\varphi)$ the subgraphs $Min(\mathscr{O}_\varphi)$ and $Max(\mathscr{O}_\varphi)$ of TG_φ. Moreover, by analogy with the case of nets, the graphs $Min(\mathscr{O}_\varphi)$ and $Max(\mathscr{O}_\varphi)$ typed over $TG_\mathscr{G}$ by (the suitable restrictions of) φ_T will be denoted by ${}^\bullet\varphi$ and $\varphi^\bullet$ respectively, and called the *source* and *target* graphs of the process.

EXAMPLE 7.4 (DETERMINISTIC PROCESS)
Figure 7.1 shows a deterministic process for grammar $\mathscr{C}$-$\mathscr{S}$ of Example 5.9. The typing morphisms from the productions of the process to TG_φ are inclusions, and the start graph is the subgraph of TG_φ containing the items $5 : S$, $\underline{0} : job$ and $4 : C$ (thus exactly the start graph G_0 of $\mathscr{C}$-$\mathscr{S}$), because these are the only items which are not generated by any production.

The morphism $\varphi_T : TG_\varphi \to TG$ is represented, as usual, by labelling the items of TG_φ with their image in TG, and the mapping φ_P from the productions of the process to those of $\mathscr{C}$-$\mathscr{S}$ is the obvious one. □

7.2 Concatenable graph processes

Since deterministic processes represent (concurrent) computations and express explicitly the dependencies between single rewriting steps, it is very natural to require a notion of processes composition "consistent" with such dependencies. To define such notion we have to face the same problem described for traces, namely given two processes φ_1 and φ_2 such that ${\varphi_1}^\bullet \simeq {}^\bullet\varphi_2$, the naïve idea of composing the two processes by gluing the target of φ_1 with the source of φ_2 does not work, since there might be several different isomorphisms between the graphs ${\varphi_1}^\bullet$ and ${}^\bullet\varphi_2$, which may lead to different results. The problem is solved as for traces, by decorating the source ${}^\bullet\varphi$ and the target $\varphi^\bullet$ of each process φ. A *concatenable graph process* is defined as a graph process equipped with two isomorphisms relating its source and target graphs to the corresponding canonical graphs. The two isomorphisms allow us to define a deterministic operation of sequential composition of processes consistent with causal dependencies, which is then exploited to define a category $\mathbf{CP}[\mathscr{G}]$ having abstract graphs as objects and (abstract) concatenable processes as arrows. Clearly the notion of sequential composition is meaningful only for unmarked processes whose source is a generic graph (see Definition 6.15).

Definition 7.5 (concatenable graph process)
Let $\mathscr{G} = \langle TG, G_s, P, \pi \rangle$ be a typed graph grammar. A concatenable graph process *for $\mathscr{G}$ is a triple*

$$\gamma = \langle m, \varphi, M \rangle$$

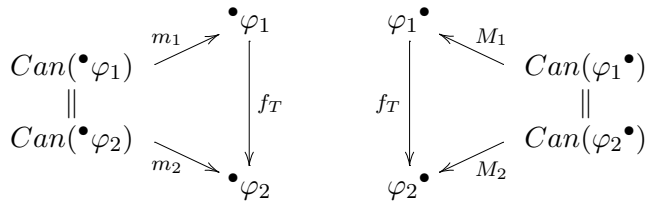
where φ is an unmarked deterministic process of $\mathscr{G}$ and $m : Can({}^\bullet\varphi) \to {}^\bullet\varphi$, $M : Can(\varphi^\bullet) \to \varphi^\bullet$ are isomorphisms (of TG-typed graphs).

It should be noted that the two isomorphisms to the source and target graphs of a process play the same role of the ordering on minimal and maximal places of *concatenable processes* in Petri net theory [DMM96]. From this point of view, *concatenable* graph processes are related to deterministic graph processes in the same way as the concatenable processes of [DMM96] are related to the classical Goltz-Reisig processes for P/T nets [GR83].

The notion of isomorphism between concatenable processes is the natural generalization of that of graph process isomorphism (see Proposition 6.14). In this case the mapping between the type graphs of the two processes is also required to be "consistent" with the decorations.

Definition 7.6 (isomorphism of concatenable processes)
Let $\gamma_1 = \langle m_1, \varphi_1, M_1 \rangle$ and $\gamma_2 = \langle m_2, \varphi_2, M_2 \rangle$ be two concatenable processes of a grammar $\mathscr{G}$. An isomorphism *between γ_1 and γ_2 is an isomorphism of processes*

$\langle f_T, f_P \rangle : \varphi_1 \to \varphi_2$ *such that the following diagrams commute:*



where $f_T : {}^\bullet\varphi_1 \to {}^\bullet\varphi_2$ *and* $f_T : \varphi_1{}^\bullet \to \varphi_2{}^\bullet$ *denote the restrictions of* f_T *to the corresponding graphs. If there exists an isomorphism* $f : \gamma_1 \to \gamma_2$ *we say that* γ_1 *and* γ_2 *are* isomorphic *and we write* $\gamma_1 \cong \gamma_2$.

DEFINITION 7.7 (ABSTRACT CONCATENABLE PROCESS)
An abstract concatenable process *is an isomorphism class of concatenable processes. It is denoted by* $[\gamma]$*, where* γ *is a member of that class.*

As usual, a particular role is played by processes based on grammars with empty set of productions.

DEFINITION 7.8 (DISCRETE PROCESS)
A discrete concatenable process *is a concatenable process* $\gamma = \langle m, \varphi, M \rangle$ *such that the corresponding occurrence grammar* $\mathscr{O}_\varphi$ *has an empty set of productions. In this case* ${}^\bullet\varphi = \varphi^\bullet = \langle TG_\varphi, \varphi_T \rangle$ *and the concatenable process is denoted by* $Sym_{\mathscr{G}}(m, G, M)$*, where* $G = {}^\bullet\varphi = \varphi^\bullet$.

Notice that two discrete concatenable processes $Sym_{\mathscr{G}}(m_j, G_j, M_j)$ $(j \in \{1, 2\})$ of a grammar $\mathscr{G}$ are isomorphic if and only if $m_1^{-1} \circ M_1 = m_2^{-1} \circ M_2$. Therefore, an abstract discrete concatenable process $[Sym_{\mathscr{G}}(m, G, M)]$ can be characterized as:

$$\{Sym_{\mathscr{G}}(m', G', M') \mid G \simeq G' \ \wedge \ m'^{-1} \circ M' = m^{-1} \circ M\}.$$

The isomorphism $m^{-1} \circ M$ is called the *automorphism on* $Can(G)$ *induced by the (abstract) discrete concatenable process.*

Given two concatenable processes γ_1 and γ_2 such that the target graph of the first one and the source graph of the second one, as well as the corresponding decorations, coincide, we can concatenate them by gluing the type graphs along the common part.

DEFINITION 7.9 (SEQUENTIAL COMPOSITION)
Let $\mathscr{G} = \langle TG, G_s, P, \pi \rangle$ *be a typed graph grammar and let* $\gamma_1 = \langle m_1, \varphi_1, M_1 \rangle$ *and* $\gamma_2 = \langle m_2, \varphi_2, M_2 \rangle$ *be two concatenable processes for* $\mathscr{G}$*, such that* $\varphi_1{}^\bullet = {}^\bullet\varphi_2$ *and* $M_1 = m_2$*. Suppose moreover that the type graphs of* γ_1 *and* γ_2 *overlap only on* $Max(\varphi_1) = Min(\varphi_2)$ *and suppose also* P_{φ_1} *and* P_{φ_2} *disjoint. Then the* concrete sequential composition *of* γ_1 *and* γ_2*, denoted by* $\gamma_1 ; \gamma_2$*, is defined as*

$$\gamma = \langle m_1, \varphi, M_2 \rangle,$$

where φ is the (componentwise) union of the processes φ_1 and φ_2. More precisely the type graph TG_φ is

$$TG_\varphi = \langle N_{TG_{\varphi_1}} \cup N_{TG_{\varphi_2}}, E_{TG_{\varphi_1}} \cup E_{TG_{\varphi_2}}, s_1 \cup s_2, t_1 \cup t_2 \rangle,$$

where s_i and t_i are the source and target functions of the graph TG_{φ_i} for $i \in \{1,2\}$, and analogously the typing morphism $\varphi_T = \varphi_{1T} \cup \varphi_{2T}$. Similarly $P_\varphi = P_{\varphi_1} \cup P_{\varphi_2}$, $\pi_\varphi = \pi_{\varphi_1} \cup \pi_{\varphi_2}$, $\varphi_P = \varphi_{1P} \cup \varphi_{2P}$ and $\iota_\varphi = \iota_{\varphi_1} \cup \iota_{\varphi_2}$. Finally, the start graph is $G'_s = {}^\bullet\varphi_1$.

It is not difficult to check that the sequential composition $\gamma_1 ; \gamma_2$ is a well-defined concatenable process. In particular observe that TG_{φ_1} and TG_{φ_2} overlap only on $Max(\varphi_1) = Min(\varphi_2)$ and therefore the asymmetric conflict relation $\nearrow_\varphi$ on $\mathscr{O}_\varphi$ can be expressed as the union of the asymmetric conflict relations of the two processes with a "connection relation" r_c, suitably relating productions of γ_1 which use items in $\varphi_1{}^\bullet$ with productions of γ_2 using corresponding items in ${}^\bullet\varphi_2$. Formally, $\nearrow_\varphi = \nearrow_{\varphi_1} \cup \nearrow_{\varphi_2} \cup r_c$, where relation $r_c \subseteq P_{\varphi_1} \times P_{\varphi_2}$ is defined by

$$r_c = \{\langle q_1, q_2 \rangle \mid \exists x \in Max(\varphi_1) = Min(\varphi_2).\ x \in (q_1{}^\bullet \cap ({}^\bullet q_2 \cup \underline{q_2})) \cup (\underline{q_1} \cap {}^\bullet q_2)\}.$$

The above characterization makes clear that the relation $\nearrow_\varphi$ is acyclic. Now, a routine checking allows us to conclude that $\mathscr{O}_\varphi$ satisfies conditions (1)-(4) of the definition of occurrence grammar (Definition 6.7). Furthermore the fact that $\mathscr{O}_\varphi$ is deterministic follows immediately from Proposition 7.2, since $\mathscr{O}_\varphi$ is finite, asymmetric conflict is acyclic and $Max(\mathscr{O}_\varphi) = Max(\mathscr{O}_{\varphi_2})$ is a well-defined graph.

The reader could be surprised and somehow displeased by the restrictiveness of the conditions which have to be satisfied in order to be able to compose two concrete processes. To understand our restrictions one should keep in mind that, as in the case of nets, the notion of sequential composition on concrete processes is not interesting in itself, but it is just introduced to be lifted to a meaningful notion of sequential composition on abstract processes. The given definition fulfills this aim since, as in the case of derivations, processes can be considered up to renaming of the items in their components. Thus, if $\varphi_1{}^\bullet \simeq {}^\bullet\varphi_2$, we can always rename the items of γ_2 to make the sequential composition defined. Hence we found it better to avoid a technically more involved (although more general) definition, leading to a nondeterministic result.

PROPOSITION 7.10
Let $\mathscr{G} = \langle TG, G_s, P, \pi \rangle$ be a typed graph grammar and let $\gamma_1 \cong \gamma'_1$ and $\gamma_2 \cong \gamma'_2$ be concatenable processes for $\mathscr{G}$. Then (if defined) $\gamma_1 ; \gamma_2 \cong \gamma'_1 ; \gamma'_2$.

PROOF. Just notice that if $f_j = \langle f_{j_T}, f_{j_P} \rangle : \gamma_j \to \gamma'_j$ $(j = 1, 2)$ are concatenable process isomorphisms, then the isomorphism between $\gamma_1 ; \gamma_2$ and $\gamma'_1 ; \gamma'_2$ can be obtained as $\langle f_{1_T} \cup f_{2_T}, f_{1_P} \cup f_{2_P} \rangle$. □

The previous proposition entails that we can lift the concrete operation of sequential composition of concatenable processes to abstract processes, by defining

$$[\gamma_1]\,;\,[\gamma_2] = [\gamma_1'\,;\,\gamma_2']$$

for $\gamma_1' \in [\gamma_1]$ and $\gamma_2' \in [\gamma_2]$ such that the concrete composition is defined.

DEFINITION 7.11 (CATEGORY OF CONCATENABLE PROCESSES)
Let $\mathscr{G} = \langle TG, G_s, P, \pi\rangle$ be a typed graph grammar. We denote by $\mathbf{CP}[\mathscr{G}]$ the category of (abstract) concatenable processes *having abstract graphs typed over TG as objects and abstract concatenable processes as arrows. An abstract concatenable process $[\langle m, \varphi, M\rangle]$ is an arrow from $[{}^\bullet\varphi]$ to $[\varphi^\bullet]$. The identity on an abstract graph $[G]$ is the discrete concatenable process $[Sym_{\mathscr{G}}(i, G, i)]$ (where $i : Can(G) \to G$ is any isomorphism), whose induced automorphism is the identity.*

A routine checking proves that the operation of sequential composition on concatenable processes is associative and that $[Sym_{\mathscr{G}}(i, G, i)]$ satisfies the identity axioms.

7.3 Relating derivation traces and processes

Although based on the same fundamental ideas, namely abstraction from representation details and true concurrency, processes and derivation traces have concretely a rather different nature. Derivation traces provide a semantics for grammars where the independence of events is represented implicitly by collecting in the same trace derivations in which the events appear in different orders. Processes, instead, provide a partial order semantics which represents explicitly the events and their relationships. In this section we show that there exists a close relationship between the trace and the process semantics introduced in the last two sections. More precisely we prove that the category $\mathbf{Tr}[\mathscr{G}]$ of concatenable (parallel) derivation traces (Definition 5.27) is isomorphic to the category of concatenable (abstract) processes $\mathbf{CP}[\mathscr{G}]$.

7.3.1 Characterization of the ctc-equivalence

The isomorphism result uses a characterization of the ctc-equivalence on decorated derivations which essentially expresses the invariant of a derivation trace. Roughly speaking, such characterization formalizes the intuition that two derivations are (c)tc-equivalent whenever it is possible to establish a correspondence between the productions that they apply and between the graph items in the two derivations, in such a way that "corresponding" productions consume and produce "corresponding" graph items. The correspondence between the graph items has to be compatible with the decorations on the source (and target) graphs.

The characterization is adapted from [CEL+96b], where it has been used for the definition of an event structure semantics for graph grammars. The only slight difference is that in the paper [CEL+96b] a different approach, based on standard isomorphisms (instead of canonical graphs), is followed to deal with the issue of trace concatenation.

The basic notions used in the characterization result presented below are those of consistent four-tuple and five-tuple.

DEFINITION 7.12 (CONSISTENT FOUR-TUPLES AND FIVE-TUPLES)
Let $\rho : G_0 \Rightarrow^* G_n$ be the derivation depicted in Figure 5.6 and let $\approx_\rho$ be the smallest equivalence relation on $\bigcup_{i=0}^{n} Items(G_i)$ such that

$$x \approx_\rho y \quad \text{if} \quad \begin{array}{l} \exists r \in \underline{n} \,.\, x \in Items(G_{r-1}) \wedge y \in Items(G_r) \wedge \\ \wedge\, \exists z \in Items(D_r) \,.\, b_r(z) = x \wedge d_r(z) = y. \end{array}$$

Denote by $Items(\rho)$ the set of equivalence classes of $\approx_\rho$, and by $[x]_\rho$ the class containing item x.[1] *For a decorated derivation ψ, we will often write $Items(\psi)$ to denote $Items(\rho_\psi)$.*

A four-tuple $\langle \rho, h_\sigma, f, \rho' \rangle$ is called consistent *if ρ and ρ' are derivations, $h_\sigma : \sigma(\rho) \to \sigma(\rho')$ is a graph isomorphism between their source graphs, $f : \underline{\#\rho} \to \underline{\#\rho'}$ is an injective function such that $prod(\rho) = prod(\rho') \circ f$, and there exists a total function $\xi : Items(\rho) \to Items(\rho')$ satisfying the following conditions:*

- *$\forall x \in Items(\sigma(\rho)) \,.\, \xi([x]_\rho) = [h_\sigma(x)]_{\rho'}$, i.e., ξ must be consistent with isomorphism h_σ;*
- *for each $j \in \underline{\#\rho}$, let i and s be determined by $j = \left(\sum_{r=1}^{i} k_r\right) + s$ (i.e., the j-th production of ρ is the s-th production of its i-th parallel direct derivation), and similarly let s' and i' satisfy $f(j) = \left(\sum_{r=1}^{i'} k'_r\right) + s'$. Then for each item x "consumed" by production $prod(\rho)(j) : \left(L \xleftarrow{l} K \xrightarrow{r} R\right)$, i.e., $x \in L - l(K)$, it must hold $\xi([g_i(in_L^s(x))]_\rho) = [g'_{i'}(in_L^{s'}(x))]_{\rho'}$. In words, ξ must relate the items consumed by corresponding production applications (according to f);*
- *a similar condition must hold for the items "created" by corresponding production applications. Using the above notations, for each $x \in R - r(K)$, $\xi([h_i(in_R^s(x))]_\rho) = [h'_{i'}(in_R^{s'}(x))]_{\rho'}$.*

Similarly, say that the five-tuple $\langle \rho, h_\sigma, f, h_\tau, \rho' \rangle$ is consistent if the "underlying" four-tuple $\langle \rho, h_\sigma, f, \rho' \rangle$ is consistent, f is a bijection, $h_\tau : \tau(\rho) \to \tau(\rho')$ is an isomorphism relating the target graphs, and the function ξ is an isomorphism that is consistent with h_τ as well (i.e., for each item $x \in \tau(\rho)$, $\xi([x]_\rho) = [h_\tau(x)]_{\rho'}$).

[1] Without loss of generality we assume here that the sets of items of the involved graphs are pairwise disjoint.

The next theorem provides a characterization of (c)tc-equivalence in terms of consistent four- (five-)tuples. We first recall some easy but useful composition properties of consistent five-tuples.

PROPOSITION 7.13

1. *If* $\langle \rho_1, h_\sigma, f, h, \rho_1' \rangle$ *and* $\langle \rho_2, h, f', h_\tau', \rho_2 \rangle$ *are consistent five-tuples, such that* $\rho_1 ; \rho_2$ *and* $\rho_1' ; \rho_2'$ *are defined, then* $\langle \rho_1 ; \rho_2, h_\sigma, f | f', h_\tau', \rho_1' ; \rho_2' \rangle$ *is consistent as well.*

2. *If* $\langle \rho, h_\sigma, f, h_\tau, \rho' \rangle$ *and* $\langle \rho', h_\sigma', f', h_\tau', \rho'' \rangle$ *are consistent five-tuples then the five-tuple* $\langle \rho, h_\sigma' \circ h_\sigma, f' \circ f, h_\tau' \circ h_\tau, \rho'' \rangle$ *is consistent as well.* □

THEOREM 7.14 (CHARACTERIZATION OF CTC- AND TC-EQUIVALENCE)
Let $\mathscr{G}$ *be any graph grammar (also non-consuming) and let* ψ, ψ' *be decorated derivations of* $\mathscr{G}$. *Then*

1. ψ *and* ψ' *are ctc-equivalent if and only if there is a permutation* Π *such that the five-tuple* $\langle \rho_\psi, m_{\psi'} \circ m_\psi^{-1}, \Pi, M_{\psi'} \circ M_\psi^{-1}, \rho_{\psi'} \rangle$ *is consistent;*

2. *similarly,* ψ *and* ψ' *are tc-equivalent if and only if there is a permutation* Π *such that the four-tuple* $\langle \rho_\psi, m_{\psi'} \circ m_\psi^{-1}, \Pi, \rho_{\psi'} \rangle$ *is consistent.*

PROOF. **Only if part of (1)**:
We show by induction that if two derivations are ctc-equivalent, i.e., $\psi \equiv_\Pi^c \psi'$ for some permutation Π, then the five-tuple $\langle \rho_\psi, m_{\psi'} \circ m_\psi^{-1}, \Pi, M_{\psi'} \circ M_\psi^{-1}, \rho_{\psi'} \rangle$ is consistent. By Definition 5.26, we have to consider the following cases:

(CTC − abs) If $\psi \equiv^{abs} \psi'$ then consistency of $\langle \rho_\psi, m_{\psi'} \circ m_\psi^{-1}, \Pi_{id}^{\#\psi}, M_{\psi'} \circ M_\psi^{-1}, \rho_{\psi'} \rangle$ follows directly by the conditions of Definition 5.16.

(CTC − sh) Since $\psi \equiv_\Pi^{sh} \psi'$ we have $m_\psi = m_{\psi'}$, $M_\psi = M_{\psi'}$ and $\rho_\psi \equiv_\Pi^{sh} \rho_{\psi'}$. Therefore this case reduces to the proof that $\rho \equiv_\Pi^{sh} \rho'$ implies the consistency of $\langle \rho, id_{\sigma(\rho)}, \Pi, id_{\tau(\rho)}, \rho' \rangle$. According to the rules of Definition 5.23, introducing shift equivalence on parallel derivations, we distinguish various cases:

(SH − id) The five-tuple $\langle \rho, id_{\sigma(\rho)}, \Pi_{id}^{\#\rho}, id_{\tau(\rho)}, \rho \rangle$ is trivially consistent.

(SH − ∅) The five-tuple $\langle G, id_G, \emptyset, id_G, \rho : G \Rightarrow_\emptyset G \rangle$ is consistent, since the isomorphism induced by ρ is id_G.

(SH − an) If $\rho \equiv_\Pi^{an} \rho'$, then the consistency of $\langle \rho, id_{\sigma(\rho)}, \Pi, id_{\tau(\rho)}, \rho' \rangle$ can be grasped from the drawing of ρ' in Figure 5.9. Consider for example production q_{j1} in q'': any item x it consumes in graph X must be in the same $\approx_{\rho'}$-equivalence class of an item of G (by the existence of morphism s), which is exactly the item consumed by the j_1-th production of ρ. To prove this formally one would need an explicit analysis construction.

(SH − syn) If $\rho \equiv_\Pi^{syn} \rho'$, the consistency of $\langle \rho, id_{\sigma(\rho)}, \Pi, id_{\tau(\rho)}, \rho' \rangle$ follows as in the previous case.

(SH − sym) The consistency of $\langle \rho', id_{\sigma(\rho')}, \Pi^{-1}, id_{\tau(\rho')}, \rho \rangle$ follows immediately from the consistency of $\langle \rho, id_{\sigma(\rho)}, \Pi, id_{\tau(\rho)}, \rho' \rangle$, since all mappings relating ρ and ρ' are invertible.

($\mathsf{SH}-\mathsf{comp}$) By the induction hypothesis $\langle \rho_1, id_{\sigma(\rho_1)}, \Pi_1, id_{\tau(\rho_1)}, \rho_1' \rangle$ and $\langle \rho_2, id_{\sigma(\rho_2)}, \Pi_2, id_{\tau(\rho_2)}, \rho_2' \rangle$ are consistent. Therefore, since $\tau(\rho_1) = \sigma(\rho_2)$, the consistency of $\langle \rho_1 ; \rho_2, id_{\sigma(\rho_1)}, \Pi_1 \mid \Pi_2, id_{\tau(\rho_2)}, \rho_1' ; \rho_2' \rangle$ follows from Proposition 7.13.(1).

($\mathsf{SH}-\mathsf{trans}$) The consistency of $\langle \rho, id_{\sigma(\rho)}, \Pi' \circ \Pi, id_{\tau(\rho)}, \rho'' \rangle$ follows by Proposition 7.13.(2), from the consistency of $\langle \rho, id_{\sigma(\rho)}, \Pi, id_{\tau(\rho)}, \rho' \rangle$ and of $\langle \rho', id_{\sigma(\rho')}, \Pi', id_{\tau(\rho')}, \rho'' \rangle$, which hold by inductive hypothesis.

($\mathsf{CTC}-\mathsf{trans}$) By induction hypothesis the tuples $\langle \rho_\psi, m_{\psi'} \circ m_\psi^{-1}, \Pi, M_{\psi'} \circ M_\psi^{-1}, \rho_{\psi'} \rangle$ and $\langle \rho_{\psi'}, m_{\psi''} \circ m_{\psi'}^{-1}, \Pi', M_{\psi''} \circ M_{\psi'}^{-1}, \rho_{\psi''} \rangle$ are consistent. Thus, by Proposition 7.13.(2), $\langle \rho_\psi, m_{\psi''} \circ m_\psi^{-1}, \Pi' \circ \Pi, M_{\psi''} \circ M_\psi^{-1}, \rho_{\psi''} \rangle$ is consistent as well.

Only if part of (2):
This implication follows from the statement just proved and from the rules ($\mathsf{TC}-\mathsf{ctc}$) and ($\mathsf{TC}-\mathsf{iso}$) defining equivalence $\equiv$, since they do not affect the consistency of the underlying four-tuple.

If part of (1):
Suppose that the five-tuple $\langle \rho_\psi, m_{\psi'} \circ m_\psi^{-1}, \Pi, M_{\psi'} \circ M_\psi^{-1}, \rho_{\psi'} \rangle$ is consistent and that $\Pi : \underline{\#\psi} \to \underline{\#\psi'}$ is a bijection (thus ψ and ψ' have the same order). By repeated applications of the analysis construction, ψ and ψ' can be transformed into equivalent, *sequential* derivations (i.e., such that at each direct derivation only one production is applied) ψ_1 and ψ_1', such that $\psi \equiv^{sh}_{\Pi_1} \psi_1$ and $\psi' \equiv^{sh}_{\Pi_2} \psi_1'$, for suitable permutations Π_1 and Π_2. By the *only if* part, five-tuples $\langle \rho_{\psi_1}, id_{\sigma(\psi_1)}, \Pi_1^{-1}, id_{\tau(\psi_1)}, \rho_\psi \rangle$ and $\langle \rho_{\psi'}, id_{\sigma(\psi')}, \Pi_2, id_{\tau(\psi')}, \rho_{\psi_1'} \rangle$ are consistent, thus $\langle \rho_{\psi_1}, m_{\psi_1'} \circ m_{\psi_1}^{-1}, \Pi_2 \circ \Pi \circ \Pi_1^{-1}, M_{\psi_1'} \circ M_{\psi_1}^{-1}, \rho_{\psi_1'} \rangle$ is consistent as well. Now there are two cases.

1. Suppose that $\Pi_2 \circ \Pi \circ \Pi_1^{-1}$ is the identity permutation. In this case it is possible to build a family of isomorphisms between the graphs of derivations ρ_{ψ_1} and $\rho_{\psi_1'}$, starting from $m_{\psi_1'} \circ m_{\psi_1}^{-1}$, and then continuing inductively by exploiting the function $\xi : \mathit{Items}(\rho_{\psi_1}) \to \mathit{Items}(\rho_{\psi_1'})$ of Definition 7.12. This family of isomorphisms satisfies all the conditions of Definition 5.16: thus it provides a proof that $\psi_1 \equiv^{abs} \psi_1'$, and therefore we have $\psi \equiv^{sh} \psi_1 \equiv^{abs} \psi_1' \equiv^{sh} \psi'$, showing that $\psi \equiv^{c} \psi'$.

2. If $\hat{\Pi} \stackrel{def}{=} \Pi_2 \circ \Pi \circ \Pi_1^{-1}$ is not the identity permutation, let $\hat{\imath} = min\{i \in \underline{\#\psi_1} \mid \hat{\Pi}(i) \neq i\}$. Then it can be shown that the $\hat{\Pi}(\hat{\imath})$-th direct derivation in ψ_1' is sequential independent from the preceding one, essentially because it can not consume any item produced or explicitly preserved after the $\hat{\imath}$-th step. By applying the synthesis and the analysis constructions the two direct derivations can be exchanged, and this procedure can be iterated producing eventually a derivation ψ_2 such that $\psi_1' \equiv^{sh}_{\Pi_3} \psi_2$ for some permutation Π_3, and such that $\Pi_3 \circ \hat{\Pi}$ is the identity permutation. Thus we are reduced to the previous case.

If part of (2):
Let $\langle \rho_\psi, m_{\psi'} \circ m_\psi^{-1}, \Pi, \rho_{\psi'} \rangle$ be a consistent four-tuple, where Π is a permutation. By exploiting isomorphisms $m_{\psi'} \circ m_\psi^{-1}$, Π and $\xi : \mathit{Items}(\rho_\psi) \to \mathit{Items}(\rho_{\psi'})$, it can be proved that the target graphs of ρ_ψ and $\rho_{\psi'}$ are isomorphic, and that there exists a unique isomorphism $h_\tau : \tau(\psi) \to \tau(\psi')$ such that $\langle \rho_\psi, m_{\psi'} \circ m_\psi^{-1}, \Pi, h_\tau, \rho_{\psi'} \rangle$ is a consistent five-tuple. Now, let α be the discrete decorated derivation $\langle M_{\psi'}, \tau(\psi'), h_\tau \circ M_\psi \rangle$. Then $\psi' ; \alpha = \langle m_{\psi'}, \rho_{\psi'}, h_\tau \circ M_\psi \rangle$. Clearly, the five-tuple $\langle \rho_\psi, m_{\psi'} \circ m_\psi^{-1}, \Pi, h_\tau \circ M_\psi \circ M_\psi^{-1}, \rho_{\psi'} \rangle$ is consistent, and by *if part of (1)* we get $\psi \equiv^{c} \psi' ; \alpha$. Then $\psi \equiv \psi'$ follows by rule ($\mathsf{CTC}-\mathsf{iso}$). □

7.3.2 From processes to traces and backwards

We are now ready to prove that for any grammar $\mathscr{G}$ the categories $\mathbf{Tr}[\mathscr{G}]$ and $\mathbf{CP}[\mathscr{G}]$ are isomorphic. We introduce two functions, the first one mapping each trace into a process and the second one performing the converse transformation. Then we show that such functions extend to functors from $\mathbf{Tr}[\mathscr{G}]$ to $\mathbf{CP}[\mathscr{G}]$ and backward, which are inverse to each other.

Given an abstract concatenable process $[\gamma]$, we construct a concatenable derivation trace by first taking the derivation which applies all the productions of γ in any order compatible with asymmetric conflict and then considering the corresponding ctc-equivalence class.

DEFINITION 7.15 (LINEARIZATION)
Let $\gamma = \langle m, \varphi, M\rangle$ be a concatenable process. A linearization *of the set P_φ of productions of the process is any bijection $e : \underline{|P_\varphi|} \to P_\varphi$, such that the corresponding linear order, defined by $q_0 \sqsubseteq q_1$ iff $e^{-1}(q_0) \leq e^{-1}(q_1)$, is compatible with the asymmetric conflict relation in φ, i.e., $q_0 \nearrow_\varphi q_1$ implies $q_0 \sqsubseteq q_1$.*

DEFINITION 7.16 (FROM PROCESSES TO TRACES)
Let $\mathscr{G} = \langle TG, G_s, P, \pi\rangle$ be a typed graph grammar and let $[\gamma]$ be an abstract concatenable process of $\mathscr{G}$, where $\gamma = \langle m, p, M\rangle$. Consider any linearization e of P_φ and define the decorated derivation $\psi(\gamma, e)$ as follows:

$$\psi(\gamma, e) = \langle m, \rho, M\rangle, \quad \text{where } \rho = \{G_{j-1} \Rightarrow_{q_j, g_j} G_j\}_{j \in \underline{|P_\varphi|}}$$

such that $G_0 = {}^\bullet\varphi$, $G_{|P_\varphi|} = \varphi^\bullet$, and for each $j \in \underline{|P_\varphi|}$

- *$q_j = \varphi_P(e(j))$;*
- *$G_j = \langle |G_j|, t_{G_j}\rangle$, where $|G_j| = reach(\{e(1), \ldots, e(j)\})$, i.e., $|G_j|$ is the subgraph of the type graph TG_p of the process determined as the reachable set $reach(\{e(1), \ldots, e(j)\})$, and t_{G_j} is the corresponding restriction of φ_T;*
- *each derivation step $G_{j-1} \Rightarrow_{q_j, g_j} G_j$ is as in Figure 7.2.(a), where unlabelled arrows represent inclusions, and the typing morphisms are not reported.*

Finally, we associate to the abstract process $[\gamma]$ the concatenable derivation trace $\mathscr{D}_A([\gamma]) = [\psi(\gamma, e)]_c$.

By using Theorem 6.11, it is not difficult to prove that $\psi(\gamma, e)$ is a legal decorated derivation in $\mathscr{G}$. The proof can be carried out by adapting the argument in [CMR96], Theorem 29. The only differences, here, are the fact that the source graph of the derivation is not, in general, the start graph of the grammar, and the presence of the decorations. Incidentally, observe that the construction above can be seen also as an instance of the pullback-retyping construction (see Section 5.3), which transforms a

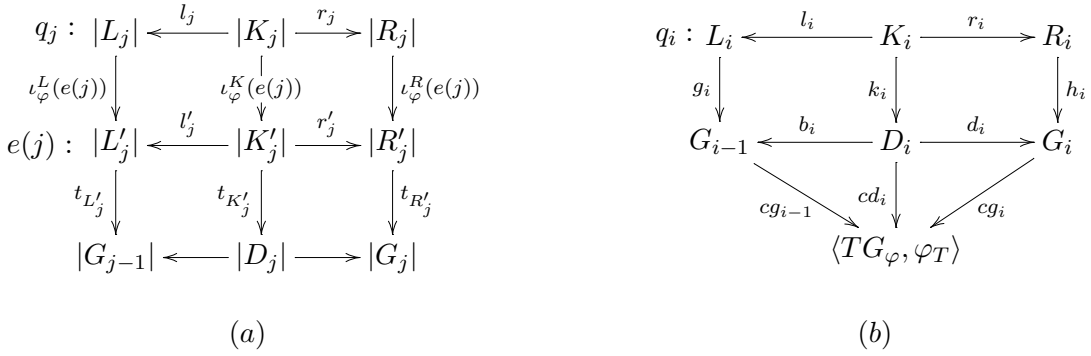


Figure 7.2: From abstract concatenable processes to concatenable derivation traces and backward.

derivation in $\mathscr{O}_\varphi$ (depicted in the low row of Figure 7.2.(a)) to a derivation in $\mathscr{G}$, by using the process morphism $\varphi : \mathscr{O}_\varphi \to \mathscr{G}$.

The following lemma shows that the mapping $\mathscr{D}_A$ can be lifted to a well-defined functor from the category of abstract concatenable processes to the category of derivation traces, which acts as identity on objects.

Lemma 7.17 (functoriality of $\mathscr{D}_A$)
Let $\mathscr{G}$ be a typed graph grammar, and let γ_1 and γ_2 be concatenable processes of $\mathscr{G}$. Then

1. ($\mathscr{D}_A$ is a well-defined mapping from abstract concatenable processes to traces)
   if $\gamma_1 \cong \gamma_2$ then $\psi(\gamma_1, e_1) \equiv^c \psi(\gamma_2, e_2)$, for any choice of the linearizations e_1 and e_2;
2. ($\mathscr{D}_A$ preserves sequential composition)
   if defined, $\psi(\gamma_1 ; \gamma_2, e) \equiv^c \psi(\gamma_1, e_1) ; \psi(\gamma_2, e_2)$, for any choice of the linearizations e_1, e_2 and e;
3. ($\mathscr{D}_A$ maps discrete processes to discrete derivation traces, and, in particular, it preserves identities)
   $\psi(Sym_{\mathscr{G}}(m, G, M), e) \equiv^c \langle m, G, M \rangle$, *for any choice of the linearization e.*

Hence $\mathscr{D}_A : \mathbf{CP}[\mathscr{G}] \to \mathbf{Tr}[\mathscr{G}]$, extended as the identity on objects, is a well-defined functor.

Proof. 1. Let $\gamma_1 \cong \gamma_2$ be two isomorphic concatenable processes of $\mathscr{G}$, with $\gamma_i = \langle m_i, \varphi_i, M_i \rangle$ for $i \in \{1, 2\}$. Let $f = \langle f_T, f_P \rangle : \gamma_1 \to \gamma_2$ be an isomorphism, and let e_1 and e_2 be linearizations of P_{φ_1} and P_{φ_2}, respectively. We show that the two decorated derivations $\psi_1 = \psi(\gamma_1, e_1)$ and $\psi_2 = \psi(\gamma_2, e_2)$ are ctc-equivalent by exhibiting a consistent five-tuple $\langle \rho_{\psi_1}, m_{\psi_2} \circ m^{-1}_{\psi_1}, \Pi, M_{\psi_2} \circ M^{-1}_{\psi_1}, \rho_{\psi_2} \rangle$.

First, it is quite easy to recognize that $Items(\psi_i)$ (see Definition 7.12) is isomorphic to $Items(TG_{\varphi_i})$, for $i \in \{1,2\}$ and thus $f_T : TG_{\varphi_1} \to TG_{\varphi_2}$ induces, in an obvious way, an isomorphism $\xi : Items(\psi_1) \to Items(\psi_2)$.

Since the restrictions of f_T to the source and target graphs of the processes are isomorphisms, we can take

$$h_\sigma = f_{T|{}^\bullet\varphi_1} : {}^\bullet\varphi_1 \to {}^\bullet\varphi_2 \qquad \text{and} \qquad h_\tau = f_{T|\varphi_1{}^\bullet} : \varphi_1{}^\bullet \to \varphi_2{}^\bullet,$$

which are compatible with $m_{\psi_2} \circ m_{\psi_1}^{-1}$ and $M_{\psi_2} \circ M_{\psi_1}^{-1}$, by definition of isomorphism of concatenable processes (and obviously compatible with ξ). Finally we can define the permutation $\Pi : \underline{\#\psi_1} \to \underline{\#\psi_2}$ as $\Pi = e_2^{-1} \circ f_P \circ e_1$.

Now, it is not difficult to check that $\langle \rho_{\psi_1}, m_{\psi_2} \circ m_{\psi_1}^{-1}, \Pi, M_{\psi_2} \circ M_{\psi_1}^{-1}, \rho_{\psi_2} \rangle$ is a consistent five-tuple. Thus, by Theorem 7.14.(1), we conclude that $\psi_1 \equiv^c \psi_2$.

2. Let $\gamma_1 = \langle m_1, \varphi_1, M_1 \rangle$ and $\gamma_2 = \langle m_2, \varphi_2, M_2 \rangle$ be two concatenable processes such that their sequential composition is defined, i.e., $\varphi_1{}^\bullet = {}^\bullet\varphi_2$, $M_1 = m_2$ and all other items in γ_1 and γ_2 are distinct. Let $\gamma = \langle m_1, \varphi, M_2 \rangle$ be their sequential composition. Let us fix linearizations e_1, e_2 and e of P_{φ_1}, P_{φ_2} and P_φ, respectively, and let $\psi_1 = \psi(\gamma_1, e_1)$, $\psi_2 = \psi(\gamma_2, e_2)$ and $\psi = \psi(\gamma, e)$.

Observe that $\sigma(\psi) = \sigma(\psi_1) = \sigma(\psi_1 \,;\, \psi_2)$ and similarly $\tau(\psi) = \tau(\psi_2) = \tau(\psi_1 \,;\, \psi_2)$. Furthermore, we have that $Items(\psi)$ and $Items(\psi_1; \psi_2)$ are (isomorphic to) $Items(TG_\varphi) = Items(TG_{\varphi_1}) \cup Items(TG_{\varphi_2})$. Therefore one can verify that the five-tuple $\langle \rho_\psi, id_{\sigma(\psi)}, \Pi, id_{\tau(\psi)}, \rho_{\psi_1 \,;\, \psi_2} \rangle$, where $\Pi = (e_1 \cup e_2)^{-1} \circ e$ is consistent (the function ξ being identity on classes). This fact, together with the observation that $m_\psi = m_{\psi_1} = m_{\psi_1;\psi_2}$ and $M_\psi = M_{\psi_2} = M_{\psi_1;\psi_2}$, allows us to conclude, by Theorem 7.14.(1), that $\psi \equiv^c \psi_1; \psi_2$.

3. Obvious. □

The backward step, from concatenable derivation traces to processes, is performed via a colimit construction that, applied to a decorated derivation ψ, essentially constructs the type graph as a copy of the source graph plus the items created during the rewriting process. Productions are instances of production applications.

DEFINITION 7.18 (FROM TRACES TO PROCESSES)
Let $\mathscr{G} = \langle TG, G_s, P, \pi \rangle$ be a typed graph grammar and let $\psi = \langle m, \rho, M \rangle$ be a decorated derivation, with $\#\psi = n$. We associate to ψ a concatenable process $cp(\psi) = \langle m', \varphi, M' \rangle$, defined as follows:

- *$\langle TG_\varphi, \varphi_T \rangle$ is a colimit object (in category TG-**Graph**) of the diagram representing derivation ψ, as depicted (for a single derivation step) in Figure 7.2.(b);*

- *$P_\varphi = \{\langle prod(\psi)(j), j \rangle \mid j \in \underline{n}\}$. For all $j \in \underline{n}$, if $prod(\psi)(j) = q_{is}$ (in the notation of Definition 5.11, Figure 5.6), then $\pi_\varphi(\langle prod(\psi)(j), j \rangle)$ is essentially production q_{is}, but typed over TG_φ (see Figure 7.2.(b)). Formally $\pi_\varphi(\langle prod(\psi)(j), j \rangle)$ is the production*

$$\langle |L_{q_{is}}|, cg_{i-1} \circ g_i \circ in^L_{is} \rangle \overset{l_{q_{is}}}{\leftarrow} \langle |K_{q_{is}}|, cd_i \circ k_i \circ in^K_{q_{is}} \rangle \overset{r_{q_{is}}}{\rightarrow} \langle |R_{q_{is}}|, cg_i \circ h_i \circ in^R_{q_{is}} \rangle$$

and $\varphi_P(\langle prod(\psi)(j), j\rangle) = prod(\psi)(j)$. *Finally the "$\iota$ component" is given by* $\iota_\varphi(\langle prod(\psi)(j), j\rangle) = \langle id_{|L_{q_{is}}|}, id_{|K_{q_{is}}|}, id_{|R_{q_{is}}|}\rangle$.

Note that for $j_1 \neq j_2$ *we may have* $prod(\psi)(j_1) = prod(\psi)(j_2)$*; instead, the productions in* P_φ *are all distinct, as they can be considered occurrences of production of* $\mathscr{G}$*;*

- *notice that* ${}^\bullet\varphi = cg_0(G_0)$ *and* $\varphi^\bullet = cg_n(G_n)$ *(and the* cg_i*'s are injective, because so are the horizontal arrows) so that we can define* $m' = cg_0 \circ m$ *and* $M' = cg_n \circ M$.

Finally we define the image of the trace $[\psi]_c$ *as* $\mathscr{C}_A([\psi]_c) = [cp(\psi)]$.

As an example, the process of Figure 7.1 can be obtained (up to isomorphism) by applying the construction just described to either the derivation of Figure 5.6 or that of Figure 5.11.

Notice that it is quite easy to have a concrete characterization of the colimit graph TG_φ. Since all the squares in the diagram representing derivation $\psi : G_0 \Rightarrow^* G_n$ commute (they are pushouts), TG_φ can be regarded equivalently as the colimit of the bottom line of the derivation, $G_0 \overset{b_1}{\leftarrow} D_1 \overset{d_1}{\rightarrow} G_1 \overset{b_2}{\leftarrow} \cdots D_n \overset{d_n}{\rightarrow} G_n$. Thus TG_φ can be characterized explicitly as the graph having as items $Items(\rho_\psi)$ (see Definition 7.12), and where source and target functions are determined in the obvious way. The injections cx_i ($x \in \{g, d\}$) simply map every item into its equivalence class.

LEMMA 7.19 ($\mathscr{C}_A$ IS WELL-DEFINED)
Let $\mathscr{G} = \langle TG, G_s, P, \pi\rangle$ *be a typed graph grammar and let* $\psi_1 \equiv^c \psi_2$ *be two decorated derivations of* $\mathscr{G}$*. Then* $cp(\psi_1) \cong cp(\psi_2)$.

PROOF. Let ψ_1 and ψ_2 be ctc-equivalent decorated derivations of $\mathscr{G}$. Then, by Theorem 7.14 there exists a five-tuple $\langle \rho_{\psi_1}, m_{\psi_2} \circ m_{\psi_1}^{-1}, \Pi, M_{\psi_2} \circ M_{\psi_1}^{-1}, \rho_{\psi_2}\rangle$ with a function $\xi : Items(\psi_1) \to Items(\psi_2)$ witnessing its consistence.

Let $cp(\psi_1) = \gamma_1 = \langle m_1, \varphi_1, M_1\rangle$ and $cp(\psi_2) = \gamma_2 = \langle m_2, \varphi_2, M_2\rangle$. First notice that $Items(\psi_i)$ is isomorphic to the set of items of the corresponding type graph TG_{φ_i} ($i \in \{1, 2\}$), and thus ξ induces readily a function $f_T : TG_{\varphi_1} \to TG_{\varphi_2}$, which, by definition of consistent five-tuple, is consistent with the decorations of the processes. Moreover the permutation Π induces a bijection $f_P : P_{\varphi_1} \to P_{\varphi_2}$, defined as $f_P(\langle prod(\psi_1)(j), j\rangle) = \langle prod(\psi_2)(\Pi(j)), \Pi(j)\rangle$, for $j \in \underline{|P_{\varphi_1}|}$.

From the definition of consistent five-tuple it follows easily that $\langle f_T, f_P\rangle : \gamma_1 \to \gamma_2$ is an isomorphism of concatenable processes. □

The next lemma shows that the constructions performed by $\mathscr{C}_A$ and $\mathscr{D}_A$ are, at abstract level, "inverse" to each other.

LEMMA 7.20 (RELATING $\mathscr{C}_A$ AND $\mathscr{D}_A$)
Let $\mathscr{G}$ *be a graph grammar. Then:*

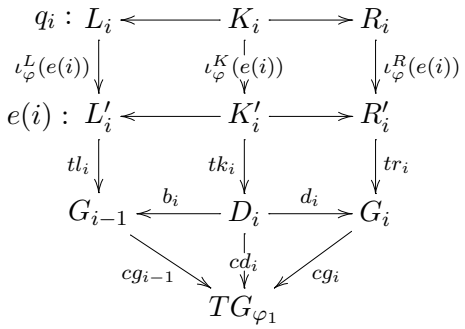
1. $\mathscr{C}_A(\mathscr{D}_A([\gamma])) = [\gamma]$*, for any concatenable process* γ*;*
2. $\mathscr{D}_A(\mathscr{C}_A([\psi]_c)) = [\psi]_c$*, for any decorated derivation* ψ.

PROOF. 1. Let $\gamma = \langle m, p, M\rangle$ be a concatenable process of $\mathscr{G}$ and let $e : \underline{|P_\varphi|} \to P_\varphi$ be a linearization of P_φ. Consider the decorated derivation:

$$\psi(\gamma, e) = \langle m, \{G_{i-1} \Rightarrow_{e(i)} G_i\}_{i\in\underline{|P_\varphi|}}, M\rangle$$

as in Definition 7.16.

Let us now construct the process $\gamma_1 = cp(\psi) = \langle m_1, \varphi_1, M_1\rangle$ as in Definition 7.18, with the type graph TG_{φ_1} obtained as colimit of the diagram:

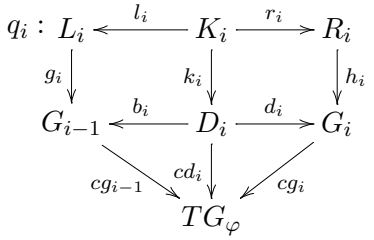


Observe that a possible concrete choice for TG_{φ_1} is TG_φ, with all cg_i's defined as inclusions.

If we define $f_P : P_{\varphi_1} \to P_\varphi$ as $f_P(\langle prod(\psi)(i), i\rangle) = e(i)$, for $i \in \underline{\#\psi}$, then it is easy to prove that $f = \langle id_{TG_\varphi}, f_P\rangle : \gamma_1 \to \gamma$ is a concatenable process isomorphism.

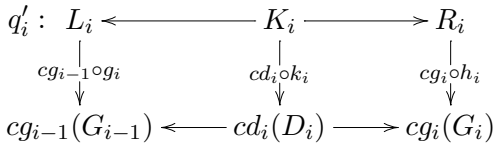
2. First of all, if ψ is a discrete decorated derivation, the result is obvious. Otherwise we can suppose $\psi = \langle m, \{G_{i-1} \Rightarrow_{q_i} G_i\}_{i\in\underline{n}}, M\rangle$ to be a derivation using a single production at each step (recall that, by Lemma 7.19, we can apply the concrete construction to any derivation in the trace and that in $[\psi]_c$ a derivation of this shape can always be found).

Let $\gamma = cp(\psi) = \langle m, p, M\rangle$ be the concatenable process built as in Definition 7.18. In particular, the type graph TG_φ is defined as the colimit of the diagram:



The set of productions is $P_\varphi = \{q'_i = \langle q_i, i\rangle \mid i \in \underline{n}\}$, with $\pi_\varphi(q'_i) = \langle |L_i|, cg_{i-1} \circ g_i\rangle \overset{l_i}{\leftarrow} \langle |K_i|, cd_i \circ k_i\rangle \overset{r_i}{\rightarrow} \langle |R_i|, cg_i \circ r_i\rangle$. Moreover $m = cg_0 \circ m_\psi$ and $M = cg_n \circ M_\psi$. Remember that all cg_i's are injections and $cg_0 : G_0 \to {}^\bullet\varphi$, $cg_n : G_n \to \varphi^\bullet$ are isomorphisms.

Now it is not difficult to verify that $prod(\psi)$ is a linearization of P_φ and $\psi' = \psi(\gamma, prod(\psi))$ is the derivation whose i^{th} step is depicted below.



The family of isomorphism $\{\theta_{X_i} : X_i \to X'_i \mid X \in \{G, D\}, i \in \underline{n}\} \cup \{\theta_{G_0}\}$ between corresponding graphs in the two (linear) derivations defined as:

$$\theta_{D_i} = cd_i \qquad \text{and} \qquad \theta_{G_i} = cg_i$$

satisfies the conditions of Definition 5.16 and thus we have that $\psi' \equiv^{abs} \psi$. Therefore $[\psi]_c = [\psi']_c = \mathscr{D}_A([\gamma]) = \mathscr{D}_A(\mathscr{C}_A([\psi]_c)$. □

The previous lemma, together with functoriality of $\mathscr{D}_A$, allows us to conclude that $\mathscr{C}_A : \mathbf{Tr}[\mathscr{G}] \to \mathbf{CP}[\mathscr{G}]$, defined as identity on objects and as $\mathscr{C}_A([\psi]_c) = [cp(\psi)]$ on morphisms, is a well-defined functor. In fact, given two decorated derivations ψ_1 and ψ_2, we have

$$\begin{aligned}
&\mathscr{C}_A([\psi_1]_c\,;\,[\psi_2]_c) = && \text{[by Lemma 7.20.(2)]}\\
&\quad = \mathscr{C}_A(\mathscr{D}_A(\mathscr{C}_A([\psi_1]_c))\,;\,\mathscr{D}_A(\mathscr{C}_A([\psi_2]_c))) = && \text{[by functoriality of } \mathscr{D}_A\text{]}\\
&\quad = \mathscr{C}_A(\mathscr{D}_A(\mathscr{C}_A([\psi_1]_c)\,;\,\mathscr{C}_A([\psi_2]_c))) = && \text{[by Lemma 7.20.(1)]}\\
&\quad = \mathscr{C}_A([\psi_1]_c)\,;\,\mathscr{C}_A([\psi_2]_c)
\end{aligned}$$

Similarly one proves also that $\mathscr{C}_A$ preserves identities. Moreover, again by Lemma 7.20, functors $\mathscr{D}_A$ and $\mathscr{C}_A$ are inverse to each other, thus implying the main result of this section.

THEOREM 7.21 (RELATING PROCESSES AND TRACES)
Let $\mathscr{G}$ be a graph grammar. Then $\mathscr{D}_A : \mathbf{CP}[\mathscr{G}] \to \mathbf{Tr}[\mathscr{G}]$ and $\mathscr{C}_A : \mathbf{Tr}[\mathscr{G}] \to \mathbf{CP}[\mathscr{G}]$ are inverse to each other, establishing an isomorphism of categories. □

7.4 Relating unfolding and deterministic processes

This section is aimed at reconciling the unfolding semantics of a graph grammar, which basically relies on the notion of nondeterministic graph process, with the semantics defined in terms of (concatenable) deterministic graph processes. The general pattern is the same we already followed in the case of nets. We show that the domain extracted from the unfolding of a grammar can be characterized as the set of deterministic processes with source in the start graph of the grammar and ordered by prefix, or, formally, that $\mathsf{Idl}(\langle [G_s] \downarrow \mathbf{CP}[\mathscr{G}]\rangle)$ is isomorphic to the domain $\mathscr{L}_i(\mathscr{E}_g(\mathscr{U}_g(\mathscr{G})))$ (see Sections 6.3-6.5 in the previous chapter).

The result can be proved for general (consuming) graph grammars, and not only for the subclass of semi-weighted graph grammars. In fact, as usual, a central role in the proof is played by the characterization of the unfolding as a universal construction. We know from the previous chapter (Section 6.4) that, when considering general morphisms, such a characterization only holds for the subcategory of semi-weighted grammars. However, limiting our attention to the (lluf) subcategories $\mathbf{GG}^R$ and $\mathbf{O\text{-}GG}^R$ where the left component of the type span of any morphism is mono (see Definition 6.25), the result generalizes to general (consuming) grammars. Simply observing that strong grammars morphisms (see Definition 6.12), which are

used to define graph processes, are particular morphisms in $\mathbf{GG}^R$, it results clear that the proof schema followed in the FIRST PART for nets can be easily adapted here to reach the desired result for general grammars.

The first observation regards the structure of the category $\langle [G_s] \downarrow \mathbf{CP}[\mathscr{G}] \rangle$, which, by definition of sequential composition between processes, can be easily shown to be a preorder.

PROPOSITION 7.22
Let $\mathscr{G}$ be a consuming grammar. Then the category $\langle [G_s] \downarrow \mathbf{CP}[\mathscr{G}] \rangle$ is a preorder.

Let $\lesssim$ be the preorder relation in $\langle [G_s] \downarrow \mathbf{CP}[\mathscr{G}] \rangle$. As in the case of nets, given two concatenable processes $\gamma : [G_s] \to [G]$ and $\gamma' : [G_s] \to [G']$, if $\gamma \lesssim \gamma' \lesssim \gamma$, then γ can be obtained from γ' simply by composing it with a discrete process. Hence the partial order induced by $\langle [G_s] \downarrow \mathbf{CP}[\mathscr{G}] \rangle$ intuitively consists of classes of processes which are "left-isomorphic", in the sense that they are related by a process isomorphism which is required to be consistent only with the decoration of the source.

The above considerations provide some intuition on the transformation defined below, which associates to each concatenable process γ a (non concatenable) marked process $\hat{\gamma}$ by forgetting the decoration of the target and using the decoration of the source to construct the "ι" component for the start graph.

DEFINITION 7.23 (FROM CONCATENABLE PROCESSES TO MARKED PROCESSES)
Let $\mathscr{G} = \langle TG, G_s, P, \pi \rangle$ be a graph grammar and let $\phi : G_s \to Can(G_s)$ be a fixed isomorphism. For any (abstract) concatenable process $\gamma = \langle m, \varphi, M \rangle : [G_s] \to [G]$ let $\hat{\gamma}$ be the marked process $\hat{\varphi}$ obtained from the unmarked process φ by adding as component $\iota^s_{\hat{\varphi}}$ the isomorphism $m \circ \phi$.

One can show that, for a fixed isomorphism ϕ, the above function establishes a bijective correspondence between the elements of the partial order induced by the preorder $\langle [G_s] \downarrow \mathbf{CP}[\mathscr{G}] \rangle$ and (finite) abstract marked processes.

Recall that the domain semantics of a graph grammar $\mathscr{G}$, as defined in the previous chapter, is obtained by taking the domain of configurations of the event structure associated to the unfolding $\mathscr{U}_g(\mathscr{G})$ of the grammar. In symbols the domain is $\mathscr{L}_i(\mathscr{E}_a(\mathscr{U}_g(\mathscr{G})))$. We next prove the announced result, characterizing the domain semantics in terms of the concatenable processes of a grammar.

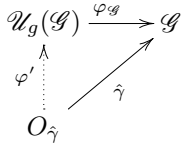
THEOREM 7.24 (UNFOLDING VS. CONCATENABLE PROCESSES)
Let $\mathscr{G}$ be a (consuming) graph grammar. Then the ideal completion of $\langle [G_s] \downarrow \mathbf{CP}[\mathscr{G}] \rangle$ is isomorphic to the domain $\mathscr{L}_i(\mathscr{E}_g(\mathscr{U}_g(\mathscr{G})))$.

PROOF (SKETCH). Let $\mathscr{G} = \langle TG, P, \pi, G_s \rangle$ be a graph grammar. As in the proof of the analogous result for nets (see, e.g., Theorem 3.67), we exploit Lemma 3.66, which ensures that the thesis follows form the existence of a surjective function $\xi : \langle [G_s] \downarrow \mathbf{CP}[\mathscr{G}] \rangle \to \mathsf{K}(\mathscr{L}_i(\mathscr{E}_g(\mathscr{U}_g(\mathscr{G}))))$ such that ξ is monotone and monic, namely for all γ_1, γ_2 in $\langle [G_s] \downarrow \mathbf{CP}[\mathscr{G}] \rangle$,

$$\gamma_1 \lesssim \gamma_2 \qquad \text{iff} \qquad \xi(\gamma_1) \sqsubseteq \xi(\gamma_2).$$

Recalling that the compact elements of the domain $\mathscr{L}_i(\mathscr{E}_g(\mathscr{U}_g(\mathscr{G})))$, associated to $\mathscr{G}$, are exactly the finite configurations of $\mathscr{E}_g(\mathscr{U}_g(\mathscr{G}))$, we can define the function ξ as follows. Let $\gamma = \langle m, \varphi, M\rangle$ be a concatenable process in $\langle [G_s] \downarrow \mathbf{CP}[\mathscr{G}]\rangle$ and consider the marked process $\hat{\gamma}$ as in Definition 7.23.

By definition of graph process, $\hat{\gamma} : \mathscr{O}_{\hat{\gamma}} \to \mathscr{G}$ is a (marked) strong morphism. Therefore, by Theorem 6.26, there exists a unique arrow $\varphi' : O_{\hat{\gamma}} \to \mathscr{U}_a(N)$, making the diagram below commute.



Then the configuration associated to γ can be defined as $\xi(\gamma) = \varphi'_P(P_{\hat{\gamma}})$. The proof that ξ is a well-defined surjective, monotone and monic function is routine. □

The above result can be informally interpreted by saying that the elements of the domain associated to a graph grammar corresponds to the marked processes of the grammar, namely to the deterministic computations starting form the initial state. Performing a further step, we can consider the PES associated to the above domain by Winskel's equivalence, namely $\mathscr{P}(\mathscr{L}_i(\mathscr{E}_g(\mathscr{U}_g(\mathscr{G}))))$. The events of such PES are special configurations which represent the possible histories of events in the IES $\mathscr{E}_g(\mathscr{U}_g(\mathscr{G}))$. Hence they correspond to processes where there is a production q which is maximum with respect to $\nearrow^*$, namely computations where there is a "last applied" production q which cannot be shifted backward.

Chapter 8

Relating Some Event Structure Semantics for DPO Graph Transformation

This chapter briefly reviews two other prime event structure semantics which have been proposed in the literature for DPO graph transformation systems. The first one [CEL+96b] is built on top of the "abstract truly concurrent model of computation" of a grammar, i.e., the category of concatenable derivation traces. The other one [Sch94] is based on a deterministic variation of the DPO approach. By the results in CHAPTERS 6 and 7, these two alternative event structures can be shown to coincide with the one obtained from the unfolding, which thus can be claimed to be "the" event structure semantics of DPO graph transformation.

8.1 Event structure from concatenable traces

The construction of a prime event structure for a (consuming) graph grammar proposed by Corradini et al. in the paper [CEL+96b], relies on the category $\mathbf{Tr}[\mathscr{G}]$ of (concatenable) derivation traces (see Section 5.2). More precisely the authors consider the category of objects of $\mathbf{Tr}[\mathscr{G}]$ under the start graph G_s, namely $\langle [G_s] \downarrow \mathbf{Tr}[\mathscr{G}] \rangle$, thought of as a synthetic representation of all the possible concurrent computations of the grammar beginning from the start graph. For consuming grammars, the partial order $\mathbf{Dom}[\mathscr{G}]$ obtained as the ideal completion of the preorder category $\langle [G_s] \downarrow \mathbf{Tr}[\mathscr{G}] \rangle$ is shown to have the desired algebraic structure, namely to be a prime algebraic, coherent and finitary partial order. Then, by Winskel's results presented in Section 2.4, $\mathbf{Dom}[\mathscr{G}]$ indirectly determines a PES, $\mathbf{ES}[\mathscr{G}] = \mathscr{P}(\mathbf{Dom}[\mathscr{G}])$, which is assumed to give the concurrent semantics of $\mathscr{G}$.

By the results in the previous chapter it is quite easy to recognize that $\mathbf{ES}[\mathscr{G}]$ coincides with the prime event structure extracted from the unfolding, namely $\mathscr{P}(\mathscr{L}_i(\mathscr{E}_g(\mathscr{U}_g(\mathscr{G}))))$. In fact, by Theorem 7.21, for any grammar $\mathscr{G}$, the category

of concatenable processes $\mathbf{CP}[\mathscr{G}]$ and that of concatenable derivation traces $\mathbf{Tr}[\mathscr{G}]$ are isomorphic, and thus also the corresponding comma categories $\langle [G_s] \downarrow \mathbf{Tr}[\mathscr{G}] \rangle$ and $\langle [G_s] \downarrow \mathbf{CP}[\mathscr{G}] \rangle$ are isomorphic. Therefore Theorem 7.24 immediately entails the desired result.

In [CEL+96b] the algebraic properties of the domain $\mathbf{Dom}[\mathscr{G}]$ are proved by presenting an explicit construction of the prime event structure $\mathbf{ES}[\mathscr{G}]$ associated to a graph grammar and then by showing that the finite configurations of $\mathbf{ES}[\mathscr{G}]$ are one-to-one with the finite elements of the domain $\mathbf{Dom}[\mathscr{G}]$. We briefly outline the construction of $\mathbf{ES}[\mathscr{G}]$, since we think that it is an interesting complement to the construction in CHAPTER 6. In fact, it provides an equivalent trace-based description of the prime event structure associated to a grammar, which, in our opinion, can be helpful to get a clearer understanding of the meaning of the events in such PES.

DEFINITION 8.1 (PES FROM TRACES)
Let $\mathscr{G} = \langle TG, G_s, P, \pi \rangle$ be a graph grammar. A pre-event *for $\mathscr{G}$ is a pair $\langle \psi, \alpha \rangle$, where ψ is a decorated derivation starting from a graph isomorphic to G_s, and α is a direct derivation which applies a single production in P (namely $\#\alpha = 1$) such that*

1. *$\psi\,;\alpha$ is defined (i.e., $\tau(\psi) = \sigma(\alpha)$, and $M_\psi = m_\alpha$) and*
2. *$\psi\,;\alpha \equiv^c_\Pi \psi'$ implies $\Pi(\#\psi + 1) = \#\psi + 1$.*

If $\langle \psi, \alpha \rangle$ is a pre-event, we denote by ε^ψ_α the corresponding derivation trace, namely $\varepsilon^\psi_\alpha = [\psi\,;\alpha]$.
An event *ε for $\mathscr{G}$ is then defined as a derivation trace $\varepsilon = \varepsilon^\psi_\alpha$ for some pre-event $\langle \psi, \alpha \rangle$. For each event ε let $Der(\varepsilon)$ denote the set of derivations containing such event, formally defined as:*

$$Der(\varepsilon) = \bigcup \{\delta \mid \exists \delta_\varepsilon \subseteq \varepsilon.\ \exists \delta'.\ \delta_\varepsilon ; \delta' = \delta\}.$$

Notice that, in particular, $\varepsilon \subseteq Der(\varepsilon)$, since each concatenable derivation trace $\delta_\varepsilon \subseteq \varepsilon$ can be concatenated with (the concatenable trace corresponding to) a discrete derivation. Then the prime event structure of grammar *$\mathscr{G}$, denoted by $\mathbf{ES}[\mathscr{G}]$, is the triple $\mathbf{ES}[\mathscr{G}] = \langle E, \leq, \# \rangle$, where E is the set of all events for $\mathscr{G}$, $\varepsilon \leq \varepsilon'$ if $Der(\varepsilon') \subseteq Der(\varepsilon)$, and $\varepsilon \# \varepsilon'$ if $Der(\varepsilon) \cap Der(\varepsilon') = \emptyset$.*

Conceptually, an event ε^ψ_α of a grammar $\mathscr{G}$ is determined by the application of a production to a graph reachable from the start graph of $\mathscr{G}$ (i.e., by the direct derivation α), together with the history that generated the graph items needed by that production application (i.e., the derivation ψ). The fact that in the pre-event $\langle \psi, \alpha \rangle$ the last step α cannot be shifted backward (requirement (2) for pre-events) guarantees that α is not independent from all the previous steps in ψ. It is worth stressing that the same requirement implies that if $\psi\,;\alpha \equiv^c_\Pi \psi'$ then $\psi' = \psi''\,;\alpha'$ with $\psi \equiv^c_{\Pi_{|\underline{\#\psi}}} \psi''$ and $\alpha \equiv^{abs} \alpha'$. Clearly, isomorphic production applications or different

linearizations of the same history should determine the same event. Therefore an event is defined as a *set* of equivalent derivations in $\mathscr{G}$, more precisely as a trace including all the derivations having (a copy of) α as last step and containing the corresponding history.

Given this notion of event, the causality and conflict relations are easily defined. In fact, considering for each event ε the set $Der(\varepsilon)$ of all the derivations that performed ε at some point, we have that two events are in conflict if there is no derivation that can perform both of them, and they are causally related if each derivation that performs one also performs the other.

EXAMPLE 8.2 (EVENT STRUCTURE OF GRAMMAR $\mathscr{C}$-$\mathscr{S}$)
Figure 8.1 depicts part of the event structure of the graph grammar $\mathscr{C}$-$\mathscr{S}$ of Example 5.9. Continuous arrows form the Hasse diagram of the causality relation, while dotted lines connect events in direct conflict (inherited conflicts are not drawn explicitly).

Recalling that, intuitively, an event of a grammar corresponds to a specific application of a production together with its "history", a careful analysis of grammar $\mathscr{C}$-$\mathscr{S}$ allows us to conclude that its event structure contains the following events:

$$E \;=\; \{req(n) \mid n \in \mathbb{N}\} \;\cup\; \{ser(w), rel(w) \mid w \in \mathbb{N}^{\otimes}\},$$

where $\mathbb{N}^{\otimes}$ denotes the set of non-empty sequences of *distinct* natural numbers.

In fact, an application of production *REQ* only depends on previous applications of the same production (because it consumes a *job* edge, and only *REQ* can produce such edges). Therefore a natural number is sufficient to represent its history: conceptually, $req(n)$ is the event corresponding to the n-th application of *REQ*. An application of production *SER*, instead, depends both on a specific application of *REQ* (because it consumes a *req* edge), and, because of the S node it consumes and produces, either on the start graph or on a previous application of *SER* itself followed by *REL* (*SER* cannot be applied in presence of a *busy* edge connected to node S because of the dangling condition). It is not difficult to check that such an event is uniquely determined by a non empty sequence of distinct natural numbers: $ser(n_1 n_2 \cdots n_k)$ is the event corresponding to the application of *SER* which serves the n_k-th *REQ*uest, after requests $n_1, \ldots, n_{k-1}$ have been served in this order. In turn, an application of production *REL* only depends on a previous application of *SER* (because of the *busy* edge), and we denote by $rel(w)$ the event caused directly by $ser(w)$.

This informal description should be sufficient to understand the part of the PES $\mathbf{ES}[\mathscr{C}\text{-}\mathscr{S}]$ shown in Figure 8.1, including only the events which are caused by the first three requests and the relationships among them. The causality and conflict relations of $\mathbf{ES}[\mathscr{C}\text{-}\mathscr{S}]$ are defined as follows:

- $req(n) \leq req(m)$ iff $n \leq m$;
- $req(n) \leq ser(w)$ iff $n \in w$, that is, an application of *SER* only depends on the request it serves and on those served in its history;

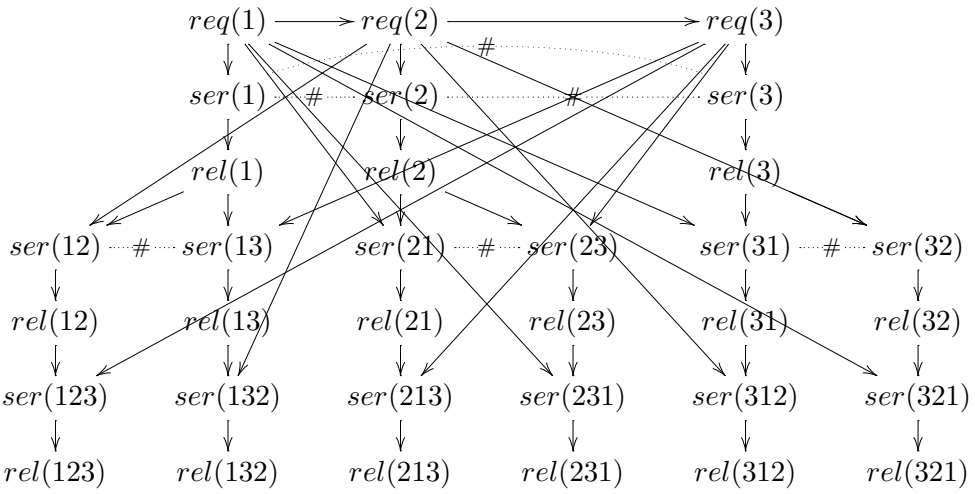


Figure 8.1: Event structure of the grammar $\mathscr{C}$-$\mathscr{S}$.

- $ser(w) \leq ser(w')$ iff $w \sqsubseteq w'$, where $\sqsubseteq$ is the prefix ordering (the application of *SER* depends only on applications of *SER* in its history);
- $ser(w) \leq rel(w')$ iff $w \sqsubseteq w'$;
- $rel(w) \leq ser(w')$ iff $w \sqsubset w'$;
- for $x, y \in \{rel, ser\}$, $x(w) \# y(w')$ iff w and w' are incomparable with respect to the prefix ordering. □

8.2 Event structure semantics from deterministic derivations

Schied in [Sch94] proposes a construction for defining an event structure semantics for *distributed rewriting systems*, an abstract unified model where several kinds of rewriting systems, such as graph grammars and term rewriting systems, naturally fit. He shows that, given a distributed rewriting system $\mathscr{R}$, a domain $\mathscr{D}_{\mathscr{R}}$ can be obtained as the quotient, with respect to shift equivalence, of the collection of derivations starting from the initial state, ordered by the prefix relation. To prove the algebraic properties of $\mathscr{D}_{\mathscr{R}}$ he constructs, as an intermediate step, a *trace language* based on the shift equivalence, and applies general results from [Bed88] to extract a prime event structure $\mathscr{E}_{\mathscr{R}}$ from the trace language. Finally he shows that $\mathscr{D}_{\mathscr{R}}$ is isomorphic to the domain of configurations of $\mathscr{E}_{\mathscr{R}}$.

The main interest in Schied's paper is for the application to graph grammars. Let us sketch how, according to Schied, the above construction instantiates to the case of graph grammars. Graph grammars are modelled as distributed rewriting systems by considering a *deterministic* variation of the DPO approach, where at each direct derivation the derived graph is uniquely determined by the rewritten graph, the applied production and the match. The idea consists of working on concrete graphs, where each item records his causal history. Formally the definition of *deterministic direct derivation* is as follows.

DEFINITION 8.3 (DETERMINISTIC DERIVATION)
Let $q : L_q \leftarrow K_q \rightarrow R_q$ be a production and let $m : L_q \rightarrow G$ be a match. Then a deterministic direct derivation $G \rightsquigarrow_{q,m} H$ *exists if m satisfies the gluing conditions and*

$$H = glue_{\langle q,m\rangle}(q, m, G) - m(L_q - l(K_q)).$$

Let $\mathscr{G} = \langle TG, G_s, P, \pi\rangle$ be a typed graph grammar. A deterministic derivation *in $\mathscr{G}$ is a sequence of deterministic direct derivations $G_s \rightsquigarrow_{q_1,m_1} G_1 \rightsquigarrow_{q_2,m_2} \ldots \rightsquigarrow_{q_n,m_n} G_n$, starting from the start graph and applying productions of $\mathscr{G}$.*

Actually the above definition instantiates the ideas of Schied to our setting which is slightly different, in that we work with *typed* graph grammars. Moreover in the original definition of Schied, since productions are nameless, the new items in H are marked only with the morphism m. Here we have to add also to name of the production.

Let $G_1 \rightsquigarrow_{q_1,m_1} G_2 \rightsquigarrow_{q_2,m_2} G_3$ be sequentially independent deterministic derivations. A basic observation is that we can shift the applications of the two productions without changing the matches, thus obtaining a new deterministic derivation $G_1 \rightsquigarrow_{q_2,m_2} G_2' \rightsquigarrow_{q_1,m_1} G_3$. The construction of the domain of a grammar is then based on the partial order of deterministic derivations endowed with the prefix relation, and on shift equivalence.

DEFINITION 8.4 (SCHIED'S DOMAIN)
The Schied's domain for a consuming grammar $\mathscr{G}$, denoted by $\mathscr{D}_{\mathscr{G}}$, is defined as the quotient, with respect to shift equivalence, of the partial order of deterministic derivations of a grammar $\mathscr{G}$ endowed with the prefix relation.

Now it is not difficult to prove that the (ideal completion of) Schied's domain coincides with the domain semantics for a grammar $\mathscr{G}$, as defined in CHAPTER 6. More precisely, the domain of configurations of the unfolding of a grammar $Conf(\mathscr{U}_g(\mathscr{G}))$ (or equivalently the domain of configurations of the IES associated to $\mathscr{G}$, i.e., $\mathscr{L}_i(\mathscr{E}_g(\mathscr{U}_g(\mathscr{G})))$) is isomorphic to $\mathsf{Idl}(\mathscr{D}_{\mathscr{G}})$.

THEOREM 8.5
For any (consuming) graph grammar $\mathscr{G}$, the ideal completion of $\mathscr{D}_{\mathscr{G}}$ and the domain $Conf(\mathscr{U}_g(\mathscr{G}))$ are isomorphic.

PROOF. Consider the function $\eta : \mathscr{D}_{\mathscr{G}} \to \mathit{Conf}(\mathscr{U}_g(\mathscr{G}))$, defined as

$$\eta([d]_{sh}) = \{\langle q_i, m_i\rangle \mid i \in \underline{n}\} \qquad \text{if } d : G_s \rightsquigarrow_{q_1,m_1} G_1 \rightsquigarrow_{q_2,m_2} \ldots \rightsquigarrow_{q_n,m_n} G_n.$$

Notice that the definition is independent from the particular choice of the representative d, since the shift construction does not change the pairs $\langle q_i, m_i\rangle$ in the derivation.

Vice versa given a configuration $C \in \mathit{Conf}(\mathscr{U}_g(\mathscr{G}))$, consider any linearization of C, compatible with the asymmetric conflict relation $\nearrow$ of the unfolding. Let $q'_1, \ldots, q'_n$ such a linearization, and suppose $q'_i = \langle q_i, m_i\rangle$ for $i \in \underline{n}$. Then (by correspondence between unfolding and deterministic processes, and by properties of deterministic processes) we can construct a derivation

$$d_C : G_s \rightsquigarrow_{q_1,m_1} G_1 \rightsquigarrow_{q_2,m_2} \ldots \rightsquigarrow_{q_n,m_n} G_n.$$

and different linearizations lead to shift equivalent derivations. Thus we can define a function $\gamma : \mathit{Conf}(\mathscr{U}_g(\mathscr{G})) \to \mathscr{D}_{\mathscr{G}}$ as $\gamma(C) = [d_C]_{sh}$, for any configuration C.

Finally, it is not difficult to verify that the two functions are monotonic and that they are inverse each other, establishing an isomorphism between the two domains. □

Final Remarks on Part II

The notions and results developed in the First Part for contextual and inhibitor nets have been fruitfully exploited to provide DPO graph transformation systems with a systematic theory of concurrency which reduces the gap existing between Petri nets and graph grammars.

The intuitive relationship between graph grammars and inhibitor nets, and the observation that safe graph grammars can be encoded by means of inhibitor nets have guided us in the definition of a Winskel's semantics for graph grammars. The notion of *occurrence grammar* and the *unfolding construction*, which can be expressed as a categorical coreflection between suitable categories of graph grammars and occurrence grammars, are introduced by following closely the approach used for inhibitor nets. Then, not surprisingly, *inhibitor event structures* turn out to be sufficiently expressive to represent the dependencies between events in graph grammar computations, and thus the unfolding can be naturally abstracted to an inhibitor event structure and finally, by using the results in Chapter 4, to a prime algebraic domain.

$$\mathbf{SW\text{-}GG} \underset{\mathscr{U}_g}{\overset{\mathscr{I}_O}{\underset{\longrightarrow}{\overset{\longleftarrow}{\bot}}}} \mathbf{O\text{-}GG} \xrightarrow{\mathscr{E}_g} \mathbf{IES} \underset{\mathscr{L}_i}{\overset{\mathscr{P}_i}{\rightleftarrows}} \mathbf{Dom} \underset{\mathscr{P}}{\overset{\mathscr{L}}{\underset{\longrightarrow}{\overset{\longleftarrow}{\sim}}}} \mathbf{PES}$$

As in the case of inhibitor nets, Winskel's construction has not been fully extended to graph grammars since the passage from occurrence grammars to inhibitor event structures is not expressed as a coreflection.

The notion of *nondeterministic graph process* which arises from our theory (the prototypical example of process being the unfolding), turns out to be a generalization of the deterministic processes of [CMR96].

Furthermore *concatenable graph processes* have been defined as a variation of the (deterministic finite) graph processes endowed with an operation of sequential composition, and have been shown to provide a semantics for graph grammars which is equivalent to the classical truly concurrent semantics based on the *shift-equivalence*. More precisely, we proved that the category $\mathbf{CP}[\mathscr{G}]$ of concatenable processes of a grammar $\mathscr{G}$ is isomorphic to the abstract truly concurrent model of computation based on traces $\mathbf{Tr}[\mathscr{G}]$ [CMR$^+$97, CEL$^+$96b, BCE$^+$99]. As already done for the other formalisms in the First Part, we have also proved that the process semantics is

strictly related to the unfolding semantics, in the sense that concatenable processes allow to recover the same domain extracted from the unfolding.

As a last result, we showed that the prime event structure extracted from the unfolding coincides both with the one in [CEL$^+$96b], obtained via a comma category construction on the category of concatenable derivation traces, and with the one in [Sch94], based on a deterministic variant of the DPO approach. Nicely, this result gives a unified view of the various event structure semantics for the DPO approach to graph transformation in the literature.

Chapter 9

Conclusions

In this thesis we singled out a general approach, inspired by the theory of ordinary Petri nets, for the development of a truly concurrent semantics of a class of systems. The semantics describes the concurrent behaviour of the systems through several mathematical structures at various levels of abstraction. The approach has been applied to contextual and inhibitor nets, two generalization of Petri nets in the literature, and finally to graph transformation systems.

The core of the approach is an *unfolding construction* which, when applied to a system, produces a single structure describing all the possible computations of the given system starting from its initial state. A more abstract representation of the behaviour of the system is obtained from the unfolding by abstracting from the nature of the states and recording only the events and the dependencies among events. This leads to a semantics based on suitable *event structure models* (*asymmetric* and *inhibitor event structures*), extending Winskel's prime event structure in order to allow for a faithful representation of the dependencies between events, without reducing them simply to causality and symmetric conflict. We developed a general theory of the mentioned event structure models, which allows one to recover, finally, a semantics in terms of more classical structures for concurrency like *prime algebraic domains* or equivalently *prime event structures*. The approach also includes a notion of deterministic process which captures the deterministic computations of the system. Deterministic processes, endowed with a suitable a notion of sequential composition, form a category which can be seen as a *model of computation* for the system at hand. The two approaches to the semantics based on the unfolding and on deterministic processes can be finally reconciled by showing that both allow to recover the same event structure for the system.

Besides the concrete results and constructions proposed for each single model, we think that an "informal" achievement of the thesis is the presentation and the treatment of (generalized) Petri nets and graph grammars in a unified framework, where also graph grammars are seen as a kind of enriched nets. This allowed us to use net-theoretical concepts to reason on graph transformation systems, a possibility which has been useful in extending constructions and results from nets to graph

grammars. We believe that the mentioned uniform presentation of nets and graph grammars can help net-theorists to get closer to graph grammars and thus can contribute, in the next future, to transfer to graph grammars some of the verification and modelling techniques existing for Petri nets.

A technical problem which remains open regards the possibility of fully extending Winskel's construction also to inhibitor nets and graph grammars, by expressing as a coreflection the whole semantical transformation leading from the category of systems to the category of domains. In fact, while the results on contextual nets can be considered completely satisfactory, in the case of inhibitor nets and graph grammars the absence of a coreflection with the category of inhibitor event structures suggests that the construction should still be improved. We observed that this problem cannot be overcome easily. A possible solution could be to look for a quite different unfolding construction, producing, for instance, in the case of nets, a flow net [Bou90] rather than an occurrence net.

An aspect which has not been considered in the thesis is the *abstract algebraic characterization* of the model of computation of a system. Well established results exist for ordinary Petri nets, whose computations have been characterized in terms of monoidal categories [MM90, Sas96]. In the case of contextual nets a partial answer to the question is given in [GM98], where it is shown that the category $\mathbf{CP}[N]$ of concatenable processes of a contextual net N can characterized as a (non full) subcategory of a free dgs-monoidal category (a variation of monoidal categories with non-natural duplicator and co-duplicator) constructed over the net. For graph transformation systems the PhD thesis [Hec98] provides a characterization of the model of computation of a grammar based on a *pseudo-free* construction in the setting of double categories with finite horizontal colimits. The result relies on the interesting assumption that, while the state of a Petri net is naturally characterized as a free commutative monoid, colimits are the right constructions for generating and composing graphs. A different solution has been proposed in [GHL99], where the *concrete* model of computation of a DPO grammar is axiomatized via the construction of a free dgs-monoidal bicategory. However the problem of giving an axiomatization of the *abstract* model of computation of a grammar as a true free construction is still unsolved and represents an interesting topic of future research.

We mentioned in the INTRODUCTION that a (concrete) truly concurrent semantics for a system represents the basis for defining more abstract observational semantics. For instance, *history preserving bisimulation* (HP-bisimulation) on P/T Petri nets [vGG89] is based on the notion of process and of deterministic event structure $\mathbf{Ev}(\varphi)$ associated to a process φ. Roughly speaking, two nets N_0 and N_1 are HP-bisimilar if, for any process φ_0 of N_0 we can find a process φ_1 of N_1 such that the underlying deterministic PES's are isomorphic. Moreover whenever φ_0 can perform an action becoming a process φ_0', then also φ_1 must be able to perform the same action becoming φ_1' and vice versa; the isomorphism between $\mathbf{Ev}(\varphi_0)$ and $\mathbf{Ev}(\varphi_1)$ is required to be extensible to an isomorphism between $\mathbf{Ev}(\varphi_0')$ and $\mathbf{Ev}(\varphi_1')$. Informally

this means that each event of a net can be simulated by an event of the other net with the same causal history. Relying on the process semantics and on the proposed event structure models we can easily define notions of history preserving bisimulation for generalized nets and for graph transformation systems. According to the chosen class of event structure models one obtains various notions of HP-bisimulation which "observe" a system at different levels of abstraction. Some preliminary results for contextual nets are illustrated in [BCM00], where it is shown that the decidability result for HP-bisimulation of ordinary nets extends also to contextual nets.

Furthermore, once an unfolding construction has been defined, a natural question suggested by the work initiated in [McM93] regards the possibility of extracting from the (possibly infinite) unfolding of a system a finite fragment which is still useful to study some relevant properties of the system. For a subclass of contextual nets, called *read persistent* nets, a generalization of McMillan's algorithm has been proposed in [VSY98]. We are confident on the possibility of further extending such result to the whole class of semi-weighted contextual nets by relying on the notion of "possible history" of a transition introduced in CHAPTER 3. However this extension could not be relevant for concrete applications since the need of considering different histories for the same event could significantly increase the complexity of the algorithm.

Finally, as already mentioned, although in this thesis we have concentrated only on basic graph rewriting acting on directed (typed) graphs, it would be interesting to understand if the presented constructions and results can extended to more general structures. While the generalization to hypergraphs is trivial, developing a similar theory for more general structures and for abstract categories (e.g., high level replacement systems [EHKPP91]) is not immediate and represents an interesting topic of further investigation.

Appendix A

Basic Category Theory

This appendix collects some basic notions of category theory which are used in the thesis. For a comprehensive introduction to the subject we refer the reader to [ML71, BW90]. The notion of category represents a formalization of the intuitive idea of a collection of objects with common structure, related by mappings which preserve such structure.

Definition A.1 (category)
A category $\mathbf{C}$ *consists of a collection* $O_{\mathbf{C}}$ *of* objects *(ranged over by* a*,* b*,...) and a collection* $A_{\mathbf{C}}$ *of* arrows *(ranged over by* f*,* g*,...) with the following structure:*

- *Each arrow has a* domain $dom(f)$ *and a* codomain $cod(f)$ *(also called* source *and* target*, respectively) that are objects. We write*

$$f : a \to b \quad \textit{or also} \quad a \xrightarrow{f} b$$

  if a *is the domain of* f *and* b *is the codomain of* f*.*

- *Given two arrows* f *and* g *such that* $cod(f) = dom(g)$*, the* composition *of* f *and* g*, written* $f;g$*, is an arrow with domain* $dom(f)$ *and codomain* $cod(g)$*:*

$$a \xrightarrow{f} b \xrightarrow{g} c = a \xrightarrow{f;g} c.$$

- *The composition operator is associative, i.e.,*

$$f;(g;h) = (f;g);h$$

  whenever f*,* g *and* h *can be composed.*

- *For every object* a *there is an* identity *arrow* $id_a : a \to a$*, such that for any arrow* $f : a \to b$

$$id_a; f = f = f; id_b.$$

The collection of arrows from an object a to an object b in $\mathbf{C}$, called homset*, is denoted by $\mathbf{C}[a, b]$.*

As usual, we write either $f \in A_{\mathbf{C}}$ or just $f \in \mathbf{C}$ to say that f is an arrow in $\mathbf{C}$, and similarly for objects.

EXAMPLE A.2 (CATEGORY **Set**)
We denote by **Set** the category whose objects are sets and whose arrows are total functions, with the ordinary composition of functions as arrow composition and the identity function id_X as identity for each set X.

EXAMPLE A.3 (CATEGORIES AND PREORDERS)
Recall that a *preorder* is a set X together with a binary relation $\leq$ which is reflexive and transitive. A preorder $(X, \leq)$ can be seen as a category $\mathbf{X}$ where the objects are the elements of X, and for any $x, y \in X$ the homset $\mathbf{X}[x, y]$ is a singleton if $x \leq y$ and the empty set otherwise.

Conversely, a category $\mathbf{A}$ where there is at most one arrow between any two objects is called a preorder category or simply a preorder. It will be often identified with the underlying preordered set $(\mathbf{A}_O, \leq)$, with $a \leq b$ iff $|\mathbf{A}[a, b]| = 1$.

Other examples are the *empty* category $\mathbf{0}$, the category $\mathbf{1}$ with one object $\bullet$ and one arrow $id_\bullet$, and the category of sets and *relations*. A category is called *discrete* if every arrow is the identity of some object. For example, $\mathbf{1}$ is a discrete category.

DEFINITION A.4 (SUBCATEGORY)
Given two categories $\mathbf{A}$ and $\mathbf{B}$, we say that $\mathbf{A}$ is a subcategory *of $\mathbf{B}$, if*

- $O_{\mathbf{A}} \subseteq O_{\mathbf{B}}$,
- *for all $a, b \in O_{\mathbf{A}}$, $\mathbf{A}[a, b] \subseteq \mathbf{B}[a, b]$,*
- *composition and identities in $\mathbf{A}$ coincide with those in $\mathbf{B}$.*

The subcategory $\mathbf{A}$ is called full *if $\mathbf{A}[a, b] = \mathbf{B}[a, b]$, for all $a, b \in O_{\mathbf{A}}$. The subcategory $\mathbf{A}$ is called* lluf *if it has the same objects of $\mathbf{A}$, namely if $O_{\mathbf{A}} = O_{\mathbf{B}}$.*

In category theory, it is common practice to express properties or requirements by means of commutative *diagrams*. A diagram is made up of objects and arrows, and we say that it *commutes* if picking any two objects in the diagram and tracing any path of arrows from one object to the other, the composition of the arrows yields always the same result. For example, the associativity of composition can be equivalently expressed by saying that the diagram *(i)* in Figure A.1 commutes. Similarly, the identities are characterized by the commutativity of the diagram *(ii)* in Figure A.1.

Figure A.1: Associativity and identity diagrams for arrow composition.

DEFINITION A.5 (INVERSE)
An arrow $f : a \to b$ in a category $\mathbf{C}$ is called an isomorphism *if there is an arrow $g : b \to a$ such that $f; g = id_a$ and $g; f = id_b$. In this case, g is called the* inverse *of f (and vice versa) and it is denoted by f^{-1} (it is immediate to prove that the inverse, if it exists, is unique).*

The set-theoretic notions of injective and surjective function are captured by the categorical concepts of *mono* and *epi* morphism.

DEFINITION A.6 (MONO AND EPI MORPHISMS)
An arrow $m : b \to c$ in a category $\mathbf{C}$ is called a monomorphism*, or simply* mono*, if for any pair of morphisms $f, g : a \to b$, $f; m = g; m$ implies $f = g$.*

Conversely, $e : a \to b$ is called an epimorphism*, or simply* epi*, if for any pair of morphisms $f, g : b \to c$, $e; f = e; g$ implies $f = g$.*

A.1 Functors

Functors are essentially morphisms between categories, in the sense that they are mappings which "preserve" the relevant structure of categories.

DEFINITION A.7 (FUNCTOR)
Given two categories $\mathbf{A}$ and $\mathbf{B}$, a functor $F : \mathbf{A} \to \mathbf{B}$ *consists of a pair of mappings (F_O, F_A) such that:*

- $F_O : O_{\mathbf{A}} \to O_{\mathbf{B}}$,
- $F_A : A_{\mathbf{A}} \to A_{\mathbf{B}}$,
- *if $f : a \to b \in \mathbf{A}$ then $F_A(f) : F_O(a) \to F_O(b)$ (but usually we omit subscripts and write $F(f) : F(a) \to F(b)$),*
- $F(f; g) = F(f); F(g)$ *for each $f : a \to b$ and $g : b \to c \in \mathbf{A}$,*
- $F(id_a) = id_{F(a)}$ *for each object $a \in \mathbf{A}$.*

A notion of composition for functors is naturally defined via the composition of their object and arrow components, seen as functions. It is easy to see that this operation is associative and admits identities, namely the (endo)functors $Id_{\mathbf{A}} : \mathbf{A} \to \mathbf{A}$ whose components are the identities on the sets of objects and arrows of $\mathbf{A}$. This yields the category **Cat** of all categories, whose objects are categories and whose arrows are functors. Consequently a notion of *isomorphism* of categories (as objects of **Cat**) naturally arises:

DEFINITION A.8 (ISOMORPHISM OF CATEGORIES)
Two categories $\mathbf{A}$ *and* $\mathbf{B}$ *are* isomorphic *if there are two functors* $F : \mathbf{A} \to \mathbf{B}$ *and* $G : \mathbf{B} \to \mathbf{A}$ *such that* $G \circ F = Id_{\mathbf{A}}$ *and* $F \circ G = Id_{\mathbf{B}}$.

Any functor $F : \mathbf{A} \to \mathbf{B}$ induces a set mapping $\mathbf{A}[a, b] \to \mathbf{B}[F(a), F(b)]$. According to the properties of such mapping, the next definition identifies special classes of functors.

DEFINITION A.9 (FAITHFUL, FULL AND EMBEDDING FUNCTORS)
A functor $F : \mathbf{A} \to \mathbf{B}$ *is called*

- faithful *if the induced mapping* $F : \mathbf{A}[a, b] \to \mathbf{B}[F(a), F(b)]$ *is injective for every pair of objects* a, b *in* $\mathbf{A}$*;*
- full *if the induced mapping* $F : \mathbf{A}[a, b] \to \mathbf{B}[F(a), F(b)]$ *is surjective for every pair of objects* a, b *in* $\mathbf{A}$*;*
- *an* embedding *if* $F_A : A_{\mathbf{A}} \to A_{\mathbf{B}}$ *is injective.*

Observe that an embedding is always injective on objects and faithful. Moreover, if it is also full it establishes an isomorphisms between $\mathbf{A}$ and the full subcategory $F(\mathbf{A})$ of $\mathbf{B}$.

A.2 Natural transformations

Natural transformations can be thought of as morphisms between functors. A natural transformation from a functor F to a functor G respects the structure of the functors in the sense that it provides a *uniform* way to translate images of objects and arrows through F to images through G.

DEFINITION A.10 (NATURAL TRANSFORMATION)
Let $F, G : \mathbf{A} \to \mathbf{B}$ *be two functors. A* natural transformation $\eta : F \dot{\longrightarrow} G : \mathbf{A} \to \mathbf{B}$ *consists of a family of arrows in* $\mathbf{B}$ *indexed by the objects of* $\mathbf{A}$*,*

$$\eta = \{\eta_a : F(a) \to G(a) \in A_{\mathbf{B}}\}_{a \in O_{\mathbf{A}}}$$

such that the diagram in Figure A.2 commutes for every arrow $f : a \to b \in \mathbf{A}$*, expressing the* naturality *of the transformation* η*. The arrow* η_a *is called the* component *at* a *of the natural transformation* η*.*

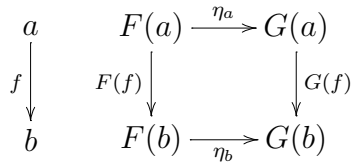


Figure A.2: Naturality square of the transformation $\eta : F \stackrel{\cdot}{\longrightarrow} G : \mathbf{A} \to \mathbf{B}$ for the arrow $f \in \mathbf{A}$.

Two natural transformations $\eta : F \stackrel{\cdot}{\longrightarrow} G$ and $\nu : G \stackrel{\cdot}{\longrightarrow} H$ can be composed elementwise, obtaining a natural transformation $\eta * \nu : F \stackrel{\cdot}{\longrightarrow} H$, with $(\eta * \nu)_a = \eta_a ; \nu_a$. Moreover, for each functor $F : \mathbf{A} \to \mathbf{B}$ there exists the obvious identity transformation $1_F : F \stackrel{\cdot}{\longrightarrow} F$ given by $1_F = \{id_{F(a)}\}_{a \in O_{\mathbf{A}}}$. This allows us to view the collection of functors between two categories $\mathbf{A}$ and $\mathbf{B}$ as the objects of a category, where arrows are natural transformations, usually denoted $\mathbf{B}^{\mathbf{A}}$ and called *functor category*.

The notion of equivalence of categories allows one to express the fact that two categories are "essentially the same". Informally, two categories are equivalent if they only differ for the fact that isomorphic objects are "counted more than once". Say that category $\mathbf{C}$ is *skeletal* two objects in $\mathbf{C}$ are isomorphic only if they are identical, and call a *skeleton* of $\mathbf{C}$ a maximal skeletal full subcategory of $\mathbf{C}$. It is possible to prove that each category $\mathbf{C}$ has a skeleton and that any two skeletons of $\mathbf{C}$ are isomorphic. Then two categories are called *equivalent* if they have isomorphic skeletons.

An alternative, more handy definition of equivalence can be given by using natural transformations.

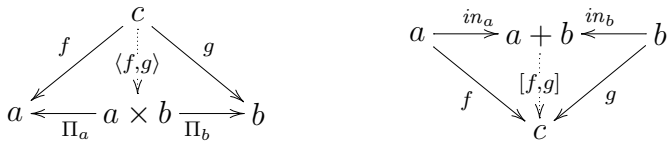
Definition A.11 (equivalence)
Two categories $\mathbf{A}$ *and* $\mathbf{B}$ *are* equivalent *if there are functors* $F : \mathbf{A} \to \mathbf{B}$ *and* $G : \mathbf{B} \to \mathbf{A}$*, and two natural isomorphisms* $Id_{\mathbf{A}} \simeq F ; G$ *and* $G ; F \simeq Id_{\mathbf{B}}$ *(where* $\simeq$ *denotes the natural isomorphism between functors).*

A.3 Universal properties, limits and colimits

In category theory, several notions are often stated by means of *universal properties*, namely by requiring the *existence* of a *unique* arrow that verifies certain properties.

A very commonly used construction is that of product, generalizing the set-theoretical cartesian product.

Definition A.12 (product)
We say that $\mathbf{C}$ *has binary* products *if for any pair of objects* $a, b \in \mathbf{C}$ *there exists*



(*i*) (*ii*)

Figure A.3: The diagrams for *(i)* products and *(ii)* coproducts.

an object $a \times b$ *together with two* projections $\Pi_a : a \times b \to a$ *and* $\Pi_b : a \times b \to b$ *satisfying the following condition:*

> *for each object* c *and arrows* $f : c \to a$ *and* $g : c \to b$ *in* $\mathbf{C}$ *there exists a unique arrow* $\langle f, g\rangle : c \to a \times b$ *such that* $f = \langle f, g\rangle; \Pi_a$ *and* $g = \langle f, g\rangle; \Pi_b$*, i.e., there exists a unique arrow* $\langle f, g\rangle : c \to a \times b$ *such that the diagram (i) in Figure A.3 commutes (the fact that* $\langle f, g\rangle$ *is depicted as a dotted arrow expresses its universal property, i.e., that it exists and is unique).*

The dual notion of product is called *coproduct*, and it generalizes the set-theoretical construction of disjoint sum.

DEFINITION A.13 (COPRODUCT)
We say that $\mathbf{C}$ *has binary* coproducts *if for any pair of objects* $a, b \in \mathbf{C}$ *there exists an object* $a + b$ *together with two* injections $in_a : a \to a + b$ *and* $in_b : b \to a + b$ *satisfying the following condition:*

> *for each object* c *and arrows* $f : a \to c$ *and* $g : b \to c$ *in* $\mathbf{C}$ *there exists a unique arrow* $[f, g] : a + b \to c$ *such that* $in_a; [f, g] = f$ *and* $in_b; [f, g] = g$*, i.e., there exists a unique arrow* $[f, g] : a + b \to c$ *such that the diagram (ii) in Figure A.3 commutes.*

The product (resp. coproduct) of two objects a and b is often denoted simply by $a \times b$ (resp. $a + b$), but notice that it is unique only up to isomorphism, and that it is uniquely determined only specifying also its projections (resp. injections).

Products and coproducts, as well as other useful notions like pullbacks, pushouts, equalizers, coequalizers, are all instances of the more general concept of *limit* and of its dual notion of *colimit*.

Given a category $\mathbf{C}$, a diagram D in $\mathbf{C}$ can be thought of as a graph $\mathbf{G}$ where each node a is labelled by an object $D(a)$ of $\mathbf{C}$, and each edge e with source a and target b is labelled by an arrow $D(e) : D(a) \to D(b)$. More precisely the diagram can be seen as a graph morphism $D : \mathbf{G} \to \mathbf{C}$, where $\mathbf{C}$ stands for the graph underlying the category $\mathbf{C}$. In this case we say that the diagram D has *shape* $\mathbf{G}$.

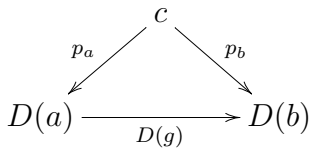


Figure A.4: Commutative cone.

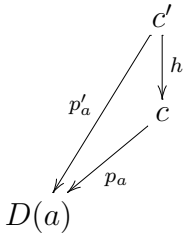


Figure A.5: An arrow h from the cone p' to p.

Definition A.14 (cone)
Let $\mathbf{C}$ *be a category and let* $D : \mathbf{G} \to \mathbf{C}$ *be a diagram in* $\mathbf{C}$ *with shape* $\mathbf{G}$*. A* cone *over the diagram* D *is an object* c *of* $\mathbf{C}$ *together with a family* $\{p_a\}_{a \in \mathbf{G}}$ *of arrows of* $\mathbf{C}$ *indexed by the nodes of* $\mathbf{G}$*, such that* $p_a : c \to D(a)$ *for each node* a *of* $\mathbf{G}$*. The arrow* p_a *is called the* component *of the cone at object* a*. We indicate the cone by writing* $p : c \to D$*.*

A cone is called *commutative* if for any arrow $g : a \to b \in \mathbf{G}$, the diagram in Figure A.4 commutes.[1] If $p' : c' \to D$ and $p : c \to D$ are cones, an *arrow* from the first to the second is an arrow $h : c' \to c$ such that for each node $a \in \mathbf{G}$, the diagram in Figure A.5 commutes.

Definition A.15 (Limit)
A commutative cone over the diagram D *is called* universal *if every commutative cone over the same diagram has a unique arrow to it. A universal cone, if such exists, is called a* limit *of the diagram* D*.*

Example A.16
A limit of the diagram (c_a, c_b) (i.e., a pair of objects), which is associated to the *discrete* graph with only two nodes a and b, is an object u, together with two arrows $p_1 : u \to c_a$ and $p_2 : u \to c_b$, such that for any other cone $(u', p'_1 : u' \to c_a, p'_2 : u' \to c_b)$ there exists a unique arrow $h : u' \to u$ with $p'_1 = h; p_1$ and $p'_2 = h; p_2$. Notice that

[1] We remark that the diagram D is not assumed to commute.

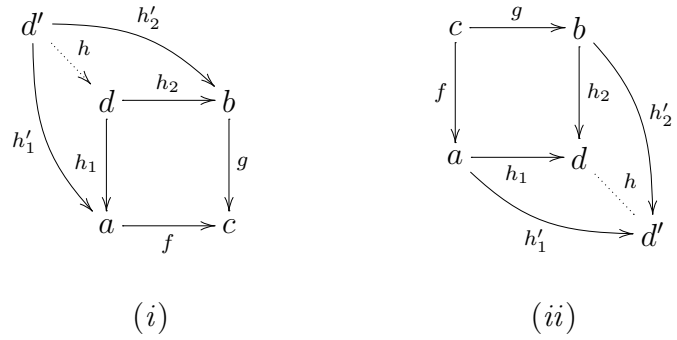


Figure A.6: Diagrams for *(i)* pullback and *(ii)* pushout.

this is just the definition of *product* of c_a and c_b. Dually, the colimit of the diagram (c_a, c_b) defines the *coproduct* of c_a and c_b (if it exists).

A very commonly used kind of (co)limit is given by *pullbacks* and *pushouts*.

DEFINITION A.17 (PULLBACK AND PUSHOUT)
Let $\mathbf{G}$ *be the graph* $\bullet \longrightarrow \bullet \longleftarrow \bullet$ *. A diagram of this shape in a category* $\mathbf{C}$ *is given by three objects* a*,* b *and* c*, and two morphisms* $f : a \to c$ *and* $g : b \to c$*. A cone for such diagram is an object* d *together with three arrows* $h_1 : d \to a$*,* $h_2 : d \to b$ *and* $h_3 : d \to c$*, such that* $h_3 = h_1; f$ *and* $h_3 = h_2; g$*. Hence,* $h_1; f = h_2; g$ *and the cone is equivalently expressed by* (d, h_1, h_2)*, because* h_3 *is uniquely determined by* h_1 *and by* h_2*. The cone is universal if for any other cone* (d', h'_1, h'_2) *of the same diagram there exists a unique arrow* $h : d' \to d$ *making the diagram in Figure A.6.(i) commute. A limit for this diagram (if it exists) is called a* pullback *of* f *and* g*.*

The dual notion is called pushout*. It can be defined as the colimit for the diagram* $(a, b, c, f : c \to a, g : c \to b)$*, which is associated to the graph* $\bullet \longleftarrow \bullet \longrightarrow \bullet$ *. Figure A.6.(ii) represents the corresponding diagram.*

An interesting property of pullbacks is the preservation of monomorphisms: if the arrow g in Figure A.6 is mono then also h_1 is mono. Dually, pushouts preserve epimorphisms.

A.4 Adjunctions

Adjunctions are extensively used in the thesis to characterize constructions by means of universal properties. There are several equivalent definitions of adjunction. We think that the more "intuitive" one relies on the scenario consisting of two categories $\mathbf{A}$ and $\mathbf{B}$ and a functor $F : \mathbf{A} \to \mathbf{B}$, where given an object b in $\mathbf{B}$ we want to find an object u in $F(\mathbf{A})$, i.e. $u = F(G_b)$ for some object G_b in $\mathbf{A}$, that better approximates b. The intuitive idea of *approximation* is formalized by the existence of an arrow

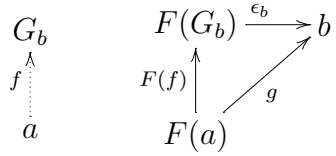


Figure A.7: The left adjoint F.

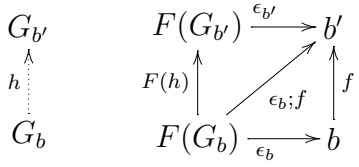


Figure A.8: The definition of the right adjoint to F.

from $F(G_b)$ to b. The fact that this approximation is *better than the others* means that any other arrow $f : F(a) \to b$ for some a in $\mathbf{A}$ factorizes through the better approximation of b via the image of a morphism in $\mathbf{A}$, in a unique way.

DEFINITION A.18 (ADJUNCTION)
Given two categories $\mathbf{A}$ *and* $\mathbf{B}$ *and a functor* $F : \mathbf{A} \to \mathbf{B}$*, we say that* F *is a* left adjoint *if for each object* b *in* $\mathbf{B}$ *there exists an object* G_b *in* $\mathbf{A}$ *and an arrow* $\epsilon_b : F(G_b) \to b$ *in* $\mathbf{B}$ *such that for any object* $a \in \mathbf{A}$ *and for any arrow* $g : F(a) \to b$ *there exists a* unique *arrow* $f : a \to G_b \in \mathbf{A}$ *such that* $g = F(f); \epsilon_b$ *(see Figure A.7).*

An immediate consequence of the fact that F is left adjoint is the existence of a functor from $\mathbf{B}$ to $\mathbf{A}$ that maps each object b into its approximation G_b, i.e., the mapping G extends to a functor. This point can be proved by noticing that, given an arrow $f : b \to b'$ in $\mathbf{B}$, the arrow $\epsilon_b; f : F(G_b) \to b'$ factorizes through $\epsilon_{b'}$ and the image of a unique arrow h from G_b to $G_{b'}$. Therefore the functor G can be defined by taking $G(f) = h$ (see Figure A.8). The functor G is called the *right adjoint* to F, and we write $F \dashv G$. The collection $\epsilon = \{\epsilon_b\}_{b \in \mathbf{B}}$ is called the *counit* of the adjunction and defines a natural transformation from $G; F$ to $1_{\mathbf{B}}$.

REMARK A.19
Adjoints are *unique* up to natural isomorphism. This is the reason why we are allowed to speak of *the* right adjoint to F.

We could have employed an equivalent *dual* approach to the definition of adjoints, by starting with the functor G and defining the "least upper" approximation F_a for each object a. This construction yields the *unit* $\eta = \{\eta_a : a \to G(F_a)\}_{a \in \mathbf{A}}$ of the adjunction (see Figure A.9).

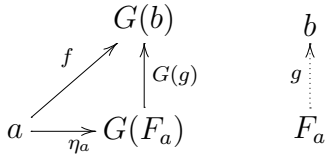


Figure A.9: The right adjoint G.

An important property of adjunctions is the preservation of universal constructions: left adjoints preserve colimits and right adjoints preserve limits.

THEOREM A.20 (ADJOINTS AND (CO)LIMITS)
Let $F : \mathbf{A} \to \mathbf{B}$ be a right adjoint functor, let $D : \mathbf{G} \to \mathbf{A}$ be a diagram in $\mathbf{A}$ and let $p : c \to D$ be the limit of D. Then $F(p) : F(c) \to F(D)$ is the limit of the diagram $F(D)$ (defined as $F(D)(x) = F(D(x))$ for any x in $\mathbf{G}$) in $\mathbf{B}$.

A dual result holds for colimits and left adjoints.

A typical example of adjunction consists of a *forgetful functor* $U : \mathbf{S} \to \mathbf{C}$, from a category $\mathbf{S}$ of certain *structures* and *structure-preserving* functions (e.g., the category **Mon** of monoids and monoid homomorphisms) to a category $\mathbf{C}$ with *less* structure (e.g., **Set**). Forgetful functors have often a left adjoint that adds the structure missing in $\mathbf{C}$ by means of a *free construction*.

Reflections and *coreflections* are two particularly important kinds of adjunction, where respectively the counit and the unit are natural isomorphisms. For instance, in the case of a coreflection, referring to the introductory discussion, we have that $\mathbf{B}$ is equivalent to a full subcategory of $\mathbf{A}$. Thus we can think that G gives the best approximation of an object of $\mathbf{A}$ inside such a subcategory. Observe that an equivalence is an adjunction which is both a reflection and a coreflection.

A.5 Comma category constructions

Given two functors $F : \mathbf{A} \to \mathbf{C}$ and $G : \mathbf{B} \to \mathbf{C}$, with a common target $\mathbf{C}$ we can consider the *comma category* $\langle F \downarrow G \rangle$. This is a standard categorical construction which makes (a selected subset) of the arrows of $\mathbf{C}$ be objects of a new category. More precisely, the objects of $\langle F \downarrow G \rangle$ are arrows of the kind $x : F(a) \to G(b)$ in $\mathbf{C}$, and an arrow $h : (x : F(a) \to G(b)) \to (y : F(a') \to G(b'))$ is a pair $(f : a \to a', g : b \to b')$ of arrows in $\mathbf{A}$ and $\mathbf{B}$, respectively, such that the diagram of Figure A.10 commutes.

Clearly, by varying the choice of F and G we obtain several different "comma constructions". Next we give an explicit definition of two relevant instances of comma construction. First if $\mathbf{A}$ is $\mathbf{1}$, then F just selects an object c in $\mathbf{C}$. If moreover $\mathbf{B} = \mathbf{C}$ and $G = Id_{\mathbf{C}}$ then $\langle F \downarrow G \rangle$, denoted in this case by $\langle c \downarrow \mathbf{C} \rangle$, is the comma category of objects of $\mathbf{C}$ under c.

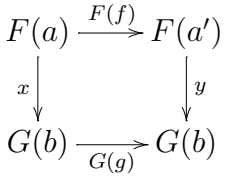


Figure A.10: Arrows in the comma category $\langle F \downarrow G \rangle$.

(i) (ii)

Figure A.11: Category of objects under/over a given object.

Definition A.21 (category of objects under a given object)
Let $\mathbf{C}$ *be a category and let* c *be an object of* $\mathbf{C}$. *The* category of objects (of $\mathbf{C}$) under c, *denoted* $\langle c \downarrow \mathbf{C} \rangle$, *has arrows like* $x : c \to c'$ *as objects. Furthermore* $f : (x : c \to c') \to (y : c \to c'')$ *is an arrow of* $\langle c \downarrow \mathbf{C} \rangle$ *if* $f : c' \to c''$ *is an arrow of* $\mathbf{C}$ *and* $x; f = y$ *(see Figure A.11.(i)).*

Dually, when $F = Id_{\mathbf{C}}$, $\mathbf{B} = \mathbf{1}$ and G is the constant c, we obtain the category of objects over a given object.

Definition A.22 (category of objects over a given object)
Let $\mathbf{C}$ *be a category and let* c *be an object of* $\mathbf{C}$. *The* category of objects (of $\mathbf{C}$) over c, *denoted* $\langle \mathbf{C} \downarrow c \rangle$, *has arrows like* $x : c' \to c$ *as objects. Furthermore* $f : (x : c' \to c) \to (y : c'' \to c)$ *is an arrow of* $\langle \mathbf{C} \downarrow c \rangle$ *if* $f : c' \to c''$ *is an arrow of* $\mathbf{C}$ *and* $x = f; y$ *(see Figure A.11.(ii)).*

Interestingly, to obtain limits and colimits in the category $\langle \mathbf{C} \downarrow c \rangle$ one can consider the corresponding diagram in $\mathbf{C}$ and compute the limit there. Hence $\langle \mathbf{C} \downarrow c \rangle$ inherits from $\mathbf{C}$ the properties of being complete and cocomplete.

Bibliography

[AF73] T. Agerwala and M. Flynn. Comments on capabilities, limitations and "correctness" of Petri nets. *Computer Architecture News*, 4(2):81–86, 1973.

[Age74] T. Agerwala. A complete model for representing the coordination of asynchronous processes. Hopkins Computer Research Report 32, John Hopkins University, 1974.

[BC88] G. Boudol and I. Castellani. Permutation of transitions: an event structure semantics for CCS and SCCS. In *Linear Time, Branching Time and Partial Order Semantics in Logics and Models for Concurrency*, volume 354 of *LNCS*, pages 411–427. Springer Verlag, 1988.

[BC96] R. Banach and A. Corradini. An opfibration account of typed DPO and DPB graph transformation: General productions. Technical Report UMCS-96-11-2, University of Manchester, Department of Computer Science, 1996.

[BCE+99] P. Baldan, A. Corradini, H. Ehrig, M. Löwe, U. Montanari, and F. Rossi. Concurrent Semantics of Algebraic Graph Transformation Systems. In G. Rozemberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation*, volume III: Concurrency, pages 107–187. World Scientific, 1999.

[BCM98a] P. Baldan, A. Corradini, and U. Montanari. Concatenable graph processes: relating processes and derivation traces. In *Proceedings of ICALP'98*, volume 1443 of *LNCS*, pages 283–295. Springer Verlag, 1998.

[BCM98b] P. Baldan, A. Corradini, and U. Montanari. An event structure semantics for P/T contextual nets: Asymmetric event structures. In M. Nivat, editor, *Proceedings of FoSSaCS '98*, volume 1378, pages 63–80. Springer Verlag, 1998.

[BCM99a] P. Baldan, A. Corradini, and U. Montanari. Contextual Petri nets, asymmetric event structures and processes. Technical Report TR-99-18, Computer Science Department, University of Pisa, 1999.

[BCM99b] P. Baldan, A. Corradini, and U. Montanari. Unfolding and Event Structure Semantics for Graph Grammars. In W. Thomas, editor, *Proceedings of FoSSaCS '99*, volume 1578, pages 73–89. Springer Verlag, 1999.

[BCM99c] P. Baldan, A. Corradini, and U. Montanari. Unfolding of double-pushout graph grammars is a coreflection. In G. Engels, editor, *TAGT'98 Conference Proceedings*. Springer Verlag, 1999. To appear.

[BCM00] P. Baldan, A. Corradini, and U. Montanari. History preserving bisimulations for contextual nets. In *WADT'99 Conference Proceedings*. Springer Verlag, 2000. To appear.

[BD87] E. Best and R. Devillers. Sequential and concurrent behaviour in Petri net theory. *Theoretical Computer Science*, 55(1):87–136, 1987.

[BDKP91] E. Best, R. Devillers, A. Kiehn, and L. Pomello. Concurrent bisimulations in Petri nets. *Acta Informatica*, 28(3):231–264, 1991.

[Bed88] M.A. Bednarczyk. *Categories of asynchronous systems*. PhD thesis, University of Sussex, 1988. Report no. 1/88.

[Ber78] G. Berry. Stable models of typed lambda-calculi. In *Proceeding of the 5^{th} ICALP*, volume 62 of *LNCS*, pages 72–89. Springer-Verlag, 1978.

[BG00] R. Bruni and F. Gadducci. Algebraic properties of (Co)Spans. To appear., 2000.

[BHMR98] F. Bueno, M. Hermenegildo, U. Montanari, and F. Rossi. Partial order and contextual net semantics for atomic and locally atomic CC programs. *Science of Computer Programming*, 30:51–82, 1998.

[Bod98] C. Bodei. Some concurrency models in a categorical framework. In *Proceedings of ICTCS'98*, pages 180–191. World Scientific, Prato 1998, 1998.

[Bou90] G. Boudol. Flow Event Structures and Flow Nets. In *Semantics of System of Concurrent Processes*, volume 469 of *LNCS*, pages 62–95. Springer Verlag, 1990.

[BP96] N. Busi and G. M. Pinna. Non Sequential Semantics for Contextual P/T Nets. In *Application and Theory of Petri Nets*, volume 1091 of *LNCS*, pages 113–132. Springer Verlag, 1996.

[BP99] N. Busi and G. M. Pinna. Process semantics for Place/Transition nets with inhibitor and read arcs. *Fundamenta Informaticæ*, 40(2-3):165–197, 1999.

[Bus98] N. Busi. *Petri Nets with Inhibitor and Read Arcs: Semantics, Analysis and Application to Process Calculi*. PhD thesis, University of Siena, Department of Computer Science, 1998.

[BW90] M. Barr and C. Wells. *Category Theory for Computing Science*. Prentice Hall, 1990.

[CEL+94a] A. Corradini, H. Ehrig, M. Löwe, U. Montanari, and F. Rossi. Abstract graph derivations in the double-pushout approach. In H.-J. Schneider and H. Ehrig, editors, *Proceedings of the Dagstuhl Seminar 9301 on Graph Transformations in Computer Science*, volume 776 of *LNCS*, pages 86–103. Springer Verlag, 1994.

[CEL+94b] A. Corradini, H. Ehrig, M. Löwe, U. Montanari, and F. Rossi. Note on standard representation of graphs and graph derivations. In H.-J. Schneider and H. Ehrig, editors, *Proceedings of the Dagstuhl Seminar 9301 on Graph Transformations in Computer Science*, volume 776 of *LNCS*, pages 104–118. Springer Verlag, 1994.

[CEL+96a] A. Corradini, H. Ehrig, M. Löwe, U. Montanari, and J. Padberg. The category of typed graph grammars and its adjunctions with categories of derivations. In J. Cuny, H. Ehrig, G. Engels, and G. Rozenberg, editors, *Proceedings of the 5th International Workshop on Graph Grammars and their Application to Computer Science*, volume 1073 of *LNCS*. Springer Verlag, 1996.

[CEL+96b] A. Corradini, H. Ehrig, M. Löwe, U. Montanari, and F. Rossi. An event structure semantics for graph grammars with parallel productions. In J. Cuny, H. Ehrig, G. Engels, and G. Rozenberg, editors, *Proceedings of the 5th International Workshop on Graph Grammars and their Application to Computer Science*, volume 1073 of *LNCS*. Springer Verlag, 1996.

[CH93] S. Christensen and N. D. Hansen. Coloured Petri nets extended with place capacities, test arcs and inhibitor arcs. In M. Ajmone-Marsan, editor, *Applications and Theory of Petri Nets*, volume 691 of *LNCS*, pages 186–205. Springer Verlag, 1993.

[CMR96] A. Corradini, U. Montanari, and F. Rossi. Graph processes. *Fundamenta Informaticae*, 26:241–265, 1996.

[CMR+97] A. Corradini, U. Montanari, F. Rossi, H. Ehrig, R. Heckel, and M. Löwe. Algebraic Approaches to Graph Transformation I: Basic Concepts and Double Pushout Approach. In G. Rozenberg, editor, *Handbook of Graph Grammars and Computing by Graph Transformation. Volume 1: Foundations.* World Scientific, 1997.

[Cor96] A. Corradini. Concurrent graph and term graph rewriting. In U. Montanari and V. Sassone, editors, *Proceedings CONCUR'96*, volume 1119 of *LNCS*, pages 438–464. Springer Verlag, 1996.

[DFMR94] N. De Francesco, U. Montanari, and G. Ristori. Modeling Concurrent Accesses to Shared Data via Petri Nets. In *Programming Concepts, Methods and Calculi, IFIP Transactions A-56*, pages 403–422. North Holland, 1994.

[DGV93] P. Degano, R. Gorrieri, and S. Vigna. On relating some models for concurrency. In M. C. Gaudel and J. P. Jouannaud, editors, *4th Conference on Theory and Practice of Software Development*, volume 668 of *LNCS*, pages 15–30. Springer-Verlag, 1993.

[DMM89] P. Degano, J. Meseguer, and U. Montanari. Axiomatizing net computations and processes. In *Proc. 4th Annual Symposium on Logic in Computer Science, Asilomar, CA, USA*, pages 175–185, 1989.

[DMM96] P. Degano, J. Meseguer, and U. Montanari. Axiomatizing the algebra of net computations and processes. *Acta Informatica*, 33:641–647, 1996.

[EHKPP91] H. Ehrig, A. Habel, H.-J. Kreowski, and F. Parisi-Presicce. Parallelism and concurrency in High-Level Replacement Systems. *Mathematical Structures in Computer Science*, 1:361–404, 1991.

[Ehr79] H. Ehrig. Introduction to the algebraic theory of graph grammars. In V. Claus, H. Ehrig, and G. Rozenberg, editors, *Proceedings of the 1st International Workshop on Graph-Grammars and Their Application to Computer Science and Biology*, volume 73 of *LNCS*, pages 1–69. Springer Verlag, 1979.

[Ehr87] H. Ehrig. Tutorial introduction to the algebraic approach of graph-grammars. In H. Ehrig, M. Nagl, G. Rozenberg, and A. Rosenfeld, editors, *Proceedings of the 3rd International Workshop on Graph-Grammars and Their Application to Computer Science*, volume 291 of *LNCS*, pages 3–14. Springer Verlag, 1987.

[EKT94] H. Ehrig, H.-J. Kreowski, and G. Taentzer. Canonical derivations for high-level replacement systems. In *[SE94]*, pages 153–169, 1994.

[GHL99] F. Gadducci, R. Heckel, and M. LLabrés. A bicategorical axiomatization of concurrent graph rewriting. In *Proceedings of CTCS'99*, volume 7 of *ENTCS*. Elsevier, 1999.

[GM98] F. Gadducci and U. Montanari. Axioms for contextual net processes. In *Proceedings of ICALP'98*, LNCS, pages 296–308. Springer Verlag, 1998.

[Gog91] J.A. Goguen. A categorical manifesto. *Mathematical Structures in Computer Science*, 1, 1991.

[GR83] U. Golz and W. Reisig. The non-sequential behaviour of Petri nets. *Information and Control*, 57:125–147, 1983.

[Hac76] M. Hack. Petri net languages. Technical Report 159, MIT, Cambridge, Massachusetts, 1976.

[HCEL96] R. Heckel, A. Corradini, H. Ehrig, and M. Löwe. Horizontal and vertical structuring of graph transformation systems. *Mathematical Structures in Computer Science*, 6(6):613–648, 1996.

[Hec98] R. Heckel. *Open Graph Transformation Systems*. PhD thesis, TU Berlin, 1998.

[JK91] R. Janicki and M. Koutny. Invariant semantics of nets with inhibitor arcs. In *Proceedings CONCUR '91*, volume 527 of *LNCS*. Springer Verlag, 1991.

[JK93] R. Janicki and M. Koutny. Structure of concurrency. *Theoretical Computer Science*, 112:5–52, 1993.

[JK95] R. Janicki and M. Koutny. Semantics of inhibitor nets. *Information and Computation*, 123:1–16, 1995.

[Kel72] R. Keller. Vector replacement systems: a formalism for modelling asynchronous systems. Technical Report 117, Princeton University, Princeton, New Jersey, 1972.

[Kos73] S. Kosaraju. Limitations of Dijkstra's semaphore primitives and Petri nets. *Operating Systems Review*, 7(4):122–126, 1973.

[Kre77] H.-J. Kreowski. *Manipulation von Graphmanipulationen*. PhD thesis, Technische Universität Berlin, 1977.

[Kre81] H.-J. Kreowski. A comparison between Petri nets and graph grammars. In H. Noltemeier, editor, *Proceedings of the Workshop on Graphtheoretic Concepts in Computer Science*, volume 100 of *LNCS*, pages 306–317. Springer Verlag, 1981.

[Kre87] H.-J. Kreowski. Is parallelism already concurrency? Part 1: Derivations in graph grammars. In H. Ehrig, M. Nagl, G. Rozenberg, and A. Rosenfeld, editors, *Proceedings of the 3rd International Workshop on Graph-Grammars and Their Application to Computer Science*, volume 291 of *LNCS*, pages 343–360. Springer Verlag, 1987.

[KW86] H.-J. Kreowski and A. Wilharm. Net processes correspond to derivation processes in graph grammars. *Theoretical Computer Science*, 44:275–305, 1986.

[Lan92a] R. Langerak. Bundle Event Structures: A Non-Interleaving Semantics for Lotos. In *5th Intl. Conf. on Formal Description Techniques (FORTE'92)*, pages 331–346. North-Holland, 1992.

[Lan92b] R. Langerak. *Transformation and Semantics for LOTOS*. PhD thesis, Department of Computer Science, University of Twente, 1992.

[McM93] K.L. McMillan. *Symbolic Model Checking*. Kluwer, 1993.

[ML71] S. Mac Lane. *Categories for the working mathematician*. Springer Verlag, 1971.

[MM90] J. Meseguer and U. Montanari. Petri nets are monoids. *Information and Computation*, 88:105–155, 1990.

[MMS92] J. Meseguer, U. Montanari, and V. Sassone. On the semantics of Petri nets. In *Proceedings CONCUR '92*, volume 630 of *LNCS*, pages 286–301. Springer Verlag, 1992.

[MMS96] J. Meseguer, U. Montanari, and V. Sassone. Process versus unfolding semantics for Place/Transition Petri nets. *Theoretical Computer Science*, 153(1-2):171–210, 1996.

[MMS97] J. Meseguer, U. Montanari, and V. Sassone. On the semantics of Place/Transition Petri nets. *Mathematical Structures in Computer Science*, 7:359–397, 1997.

[MP98] U. Montanari and M. Pistore. History-dependent automata. Technical Report TR-98-11, Dipartimento di Informatica, 1998. Available as ftp://ftp.di.unipi.it/pub/techreports/TR-98-11.ps.Z.

[MR94] U. Montanari and F. Rossi. Contextual occurrence nets and concurrent constraint programming. In H.-J. Schneider and H. Ehrig, editors, *Proceedings of the Dagstuhl Seminar 9301 on Graph Transformations in Computer Science*, volume 776 of *LNCS*. Springer Verlag, 1994.

[MR95] U. Montanari and F. Rossi. Contextual nets. *Acta Informatica*, 32, 1995.

[MSW96] A. Maggiolo-Schettini and J. Winkowski. Dynamic graphs. In *Proceedings of MFCS'96*, volume 1113 of *LNCS*, pages 431–442, 1996.

[NPW81] M. Nielsen, G. Plotkin, and G. Winskel. Petri Nets, Event Structures and Domains, Part 1. *Theoretical Computer Science*, 13:85–108, 1981.

[Pet62] C.A. Petri. *Kommunikation mit Automaten*. PhD thesis, Schriften des Institutes für Instrumentelle Matematik, Bonn, 1962.

[Pet77] C.A. Petri. Non-sequential processes. Technical Report GMD-ISF-77-5, Gesellshaft für Mathematik und Datenverarbeitung, Bonn, 1977.

[Pet81] J.L. Peterson. *Petri Net Theory and the Modelling of Systems*. Prentice-Hall, 1981.

[PP92] G. M. Pinna and A. Poigné. On the nature of events. In *Mathematical Foundations of Computer Science*, volume 629 of *LNCS*, pages 430–441. Springer Verlag, 1992.

[PP95] G. M. Pinna and A. Poigné. On the nature of events: another perspective in concurrency. *Theoretical Computer Science*, 138:425–454, 1995.

[Rei85] W. Reisig. *Petri Nets: An Introduction*. EACTS Monographs on Theoretical Computer Science. Springer Verlag, 1985.

[Rib96] L. Ribeiro. *Parallel Composition and Unfolding Semantics of Graph Grammars*. PhD thesis, Technische Universität Berlin, 1996.

[Ris94] G. Ristori. *Modelling Systems with Shared Resources via Petri Nets*. PhD thesis, Department of Computer Science - University of Pisa, 1994.

[RT88] A. Rabinovich and B. A. Trakhtenbrot. Behavior Structures and Nets. *Fundamenta Informaticæ*, 11(4):357–404, 1988.

[Sas96] V. Sassone. An axiomatization of the algebra of Petri net concatenable processes. *Theoretical Computer Science*, 170:277–296, 1996.

[Sch93] H.-J. Schneider. On categorical graph grammars integrating structural transformations and operations on labels. *Theoretical Computer Science*, 109:257–274, 1993.

[Sch94] G. Schied. On relating rewriting systems and graph grammars to event structures. In H.-J. Schneider and H. Ehrig, editors, *Proceedings of the Dagstuhl Seminar 9301 on Graph Transformations in Computer Science*, volume 776 of *LNCS*, pages 326–340. Springer Verlag, 1994.

[Sco70] D. S. Scott. Outline of a mathematical theory of computation. In *Proceedings of the Fourth Annual Princeton Conference on Information Sciences and Systems*, pages 169–176, 1970.

[SE94] H.-J. Schneider and H. Ehrig, editors. *Proceedings of the Dagstuhl Seminar 9301 on Graph Transformations in Computer Science*, volume 776 of *LNCS*. Springer Verlag, 1994.

[vGG89] R. van Glabbeek and U. Goltz. Equivalence notions for concurrent systems and refinement of actions. In A. Kreczmar and G. Mirkowska, editors, *Mathematical Foundations of Computer Science*, volume 39 of *LNCS*, pages 237–248. Springer Verlag, 1989.

[Vog91] W. Vogler. Deciding history preserving bisimilarity. In J. Leach Albert, B. Monien, and M. Rodríguez-Artalejo, editors, *Proceedings of ICALP'91*, volume 510 of *LNCS*, pages 495–505. Springer-Verlag, 1991.

[Vog96] W. Vogler. Efficiency of asynchronous systems and read arcs in Petri nets. Technical Report 352, Institüt für Mathematik, Augsburg University, 1996.

[Vog97a] W. Vogler. Efficiency of asynchronous systems and read arcs in Petri nets. In *Proc. ICALP'97*, volume 1256 of *LNCS*, pages 538–548. Springer Verlag, 1997.

[Vog97b] W. Vogler. Partial Order Semantics and Read Arcs. In *Mathematical Foundations of Computer Science*, volume 1295 of *LNCS*, pages 508–518. Springer Verlag, 1997.

[VSY98] W. Vogler, A. Semenov, and A. Yakovlev. Unfolding and finite prefix for nets with read arcs. In *Proceedings of CONCUR'98*, volume 1466 of *LNCS*, pages 501–516. Springer-Verlag, 1998.

[Win87a] G. Winskel. Event Structures. In *Petri Nets: Applications and Relationships to Other Models of Concurrency*, volume 255 of *LNCS*, pages 325–392. Springer Verlag, 1987.

[Win87b] G. Winskel. Petri nets, algebras, morphisms, and compositionality. *Information and Computation*, 72(3):197–238, 1987.

[Win98] J. Winkowsky. Processes of Contextual Nets and their Characteristics. Technical Report 847, Institute of Computer Science, Polish Academy of Sciences, 1998.

Index

Symbols

A

L

M

N