Agent-based simulation of animal behaviour

C.M. Jonker, J. Treur

Software Engineering (SEN)

Agent-based Simulation of Animal Behaviour

Catholijn M. Jonker, Jan Treur[1]

[1] *CWI*
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

and

Vrije Universiteit Amsterdam, Department of Artificial Intelligence
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
URL: http://www.cs.vu.nl/~{jonker,treur} Email: {jonker,treur}@cs.vu.nl

ABSTRACT

In this paper it is shown how animal behaviour can be simulated in an agent-based manner. Different models are shown for different types of behaviour, varying from purely reactive behaviour to pro-active, social and adaptive behaviour. The compositional development method for multi-agent systems DESIRE and its software environment supports the conceptual and detailed design, and execution of these models. Experiments reported in the literature on animal behaviour have been simulated for a number of agent models.

1991 Computing Reviews Classification System: I.2.11
Keywords and Phrases: computer simulation, agent, animal behaviour, delayed response, reactive, pro-active, social, adaptive
Note: work carried out in theme SEN4.

1. INTRODUCTION

One of the most important aspects of agents (cf. [25]) is their behaviour. In the past, behaviour has been studied in different disciplines. In Cognitive Psychology the analysis of human behaviour is a major topic. In Biology, animal behaviour has been and is being studied extensively. Around 1900 a discussion took place about the manner in which observed animal behaviour can be interpreted in order to obtain an objective and testable description; for an overview, see [2], [23]. A risk of taking the intentional stance (e.g., [10]) as a perspective to explain behaviour, is that explanations are generated that make use of (a large number of) mental concepts that cannot be tested empirically. Therefore the *principle of parsimony* was introduced, stating that 'in no case may we interpret an action as the outcome of the exercise of a higher psychical faculty, if it can be interpreted as the outcome of the exercise of one which stands lower in the psychological scale'; see [18].

Building further on this perspective *behaviourism* was developed, e.g., [12], [21], [24]. In this approach animal behaviour is explained only in terms of a black box that for each pattern of *stimuli* (input of the black box) from the environment generates a *response* (output of the black box), that functionally depends on the input pattern of stimuli; i.e., if two patterns of stimuli are offered, then the same behaviour occurs if the two patterns of stimuli are equal. This view was also extended to human behaviour. Because of the underlying black box view, behaviourism discouraged reference to internal (mental) activities of organisms: any speculation about the internal functioning of the black box (i.e., the processes that might mediate between sensory inputs and behavioural outputs) was forbidden; cf. [23], p. 4.

In this paper, the compositional development method for multi-agent systems DESIRE (see [4]) is used to design, implement and experiment with agent-based simulation models for animal behaviour. In Section 4 a generic model of a *purely reactive agent* is introduced which is an adequate agent model to describe the (immediate) functional character of stimulus-response behaviour. The black box is represented by the agent. The stimuli form the input (observation results), and the response is formed by the actions generated as output.

Viewed from a Software Engineering perspective, modelling behaviour by a functional relation between input and output provides a system that can be described as a (mathematical) *function*

$$F : \text{Input\_states} \rightarrow \text{Output\_states}$$

of the set of possible input states to the set of possible output states. Such a system is transparent and predictable. For the same input always the same behaviour is repeated: its behaviour does not depend on earlier

processes; no information on previous experiences is used to affect behaviour. Well-known traditional programming methods are based on this paradigm; for example, program specification and refinement based on preconditions and postconditions as developed in, e.g., [11].

As opposed to behaviour defined by a purely functional dependency between input and output, an agent's behaviour often takes previous processes in which it was involved into account. These previous processes may have led to internal storage of information in a *memory* so that the same input pattern of stimuli can lead to different behaviour a next time it is encountered; the agent may be able to deliberate about it. Again viewed from a Software Engineering perspective, this makes that agents do not fit strictly in the paradigm based on a functional relation: to keep the functional relation, not only the actual input, but also the complete history of input should be taken into account, or the internal information in memory should be considered to be additional input.

In Section 5 a generic agent model is presented that can be used to model more complex behaviour. It includes not only components that represent the agent's memory (the agent's beliefs on the world and on other agents), but also components that represent the agent's capabilities to control its own behaviour (goal-directness, adaptation), the agent's interaction with the world, and the agent's communication with other agents. This generic agent model has been used to obtain different agent models for different types of animal behaviour that go beyond purely reactive behaviour: *delayed response behaviour*, *deliberate pro-active behaviour*, *social behaviour* (in Section 6) and *adaptive behaviour* (in Section 7). In Section 2 a problem description (a description of a pseudo-experiment) is presented; in Section 8 the behaviours of the different agent models introduced in Section 4, 5, 6 and 7 are compared for each of the situations defined in Section 2 and example traces are discussed.

2. PROBLEM DESCRIPTION

The deliberations put forward in the introduction can be illustrated by a very concrete example, taken from the discipline that studies animal behaviour; e.g., [23]. Animals, for example dogs, sometimes show a *delayed response*: they look for food in places where they have seen food before. This suggests that these animals might have some internal representation or memory of stimuli received earlier. More systematic experiments on this delayed response issue, for example those reported in [13] and [22], support this suggestion.

2.1. The Domain

The type of experiment reported in [22] is set up as follows (see Figure 1). Separated by a transparent screen (a window, at position p0), at each of two positions p1 and p2 a cup (upside down) and/or a piece of food can be placed. At some moment (with variable delay) the screen is raised, and the animal is free to go to any position. Consider the following three possible situations:

Situation 1 At both positions p1 and p2 an empty cup is placed.

Situation 2 At position p1 an empty cup is placed, and at position p2 a piece of food, which is (and remains) visible for the animal.

Situation 3 At position p1 an empty cup is placed and at position p2 a piece of food is placed, after which a cup is placed at the same position, covering the food. After the food disappears under the cup it cannot be sensed anymore by the animal.

Situation 4 At position p1 an empty cup is placed and at position p2 a cup and a piece of food is placed, in such a manner that the animal did not see the food.

In situation 1 the animal will not show a preference for either position p1 or p2; it may even go elsewhere or stay where it is. In situation 2 the animal will go to position p2, which can be explained as pure stimulus-response behaviour. In situation 3 the immediate stimuli are the same as in situation 1. Animals that react in a strictly functional stimulus-response manner will respond to this situation as in situation 1. Animals that show delayed response behaviour will go to p2, where food can be found. In situation 4 animals will behave as in situation 1; however, here they can find food and eat it, and therefore be reinforced in successful behaviour.

In the literature, many reports can be found of observed delayed response behaviour in experiments of the type described above: [23], p. 4-5. The animal species used in these experiments vary from rats and dogs to macaques, chimpanzees and human infants. Therefore, it is assumed that animals of the type studied maintain internal (mental) representations on the basis of their sensor input, and that they make use of these representations (in addition to the actual sensory input that is used) to determine their behaviour. In a way it can be said that they may act as deliberate agents.
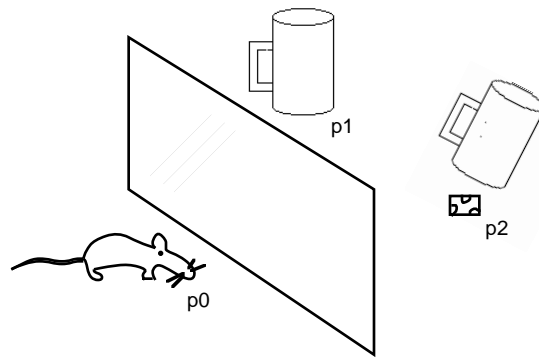
Figure 1: Situation 3 of the experiment

2.2. *The Requirements*

In this paper four agent models A, B, C and D for the experiment are described. The following requirements on their behaviour express possible hypotheses that can be made about the behaviour of animals in the experiments:

A .   An agent with purely reactive behaviour should behave the same for the two situations 1 and 3 described above: doing nothing, as if no food is present. Only in situation 2 should it go to the position of the food.

B .   An agent with delayed response behaviour should behave the same in the situations 2 and 3: it should go to the position of the food. In situation 1 it should do nothing.

C .   A deliberate pro-active agent's behaviour in the situations 1, 2 and 3 depends on whether the agent has a motivation or goal to do so. E.g., the agent may start acting in a pro-active manner (without any specific stimulus) in situation 1.

D .   A social agent is able to take into account communication with other agents. If another animal is present that communicates that it wants to have the food (e.g., by growling), and the agent believes that this other agent is higher in the hierarchy, then the agent will not try to get the food.

E .   An adaptive agent is able to adapt its behaviour on the basis of experiences. If the cups have different colours, and a certain colour is used more often for cups under which food is hidden, then after some learning time the agent has a preference to look for food at positions with a cup of that colour with priority.

3. COMPOSITIONAL DEVELOPMENT OF MULTI-AGENT SYSTEMS

The agent-based models for animal behaviour described in this paper have been developed using the compositional development method DESIRE for multi-agent systems (DEsign and Specification of Interacting REasoning components); cf. [4]. The development of a multi-agent system is supported by graphical design tools within the DESIRE software environment. Translation to an operational system is straightforward; the software environment includes implementation generators with which formal specifications can be translated into executable code of a prototype system. In DESIRE, a design consists of knowledge of the following three types: process composition, knowledge composition, the relation between process composition and knowledge composition. These three types of knowledge are discussed in more detail below.

3.1.   *Process Composition*

Process composition identifies the relevant processes at different levels of (process) abstraction, and describes how a process can be defined in terms of (is composed of) lower level processes.

3.1.1. Identification of Processes at Different Levels of Abstraction

Processes can be described at different levels of abstraction; for example, the process of the multi-agent system as a whole, processes defined by individual agents and the external world, and processes defined by task-related components of individual agents. The identified processes are modelled as *components*. For each process the *input and output information types* are modelled. The identified levels of process abstraction are modelled as *abstraction/specialisation relations* between components: components may be *composed* of other components or they may be *primitive*. Primitive components may be either reasoning components (i.e., based on a knowledge base), or, components capable of performing tasks such as calculation, information retrieval, optimisation. These levels of process abstraction provide process hiding at each level.

3.1.2. Composition of Processes

The way in which processes at one level of abstraction are composed of processes at the adjacent lower abstraction level is called *composition*. This composition of processes is described by a specification of the possibilities for *information exchange* between processes (*static view* on the composition), and a specification of *task control knowledge* used to control processes and information exchange (*dynamic view* on the composition).

3.2. Knowledge Composition

Knowledge composition identifies the knowledge structures at different levels of (knowledge) abstraction, and describes how a knowledge structure can be defined in terms of lower level knowledge structures. The knowledge abstraction levels may correspond to the process abstraction levels, but this is often not the case.

3.2.1. Identification of Knowledge Structures at Different Abstraction Levels

The two main structures used as building blocks to model knowledge are: *information types* and *knowledge bases*. Knowledge structures can be identified and described at different levels of abstraction. At higher levels details can be hidden. An *information type* defines an ontology (lexicon, vocabulary) to describe objects or terms, their sorts, and the relations or functions that can be defined on these objects. Information types can logically be represented in order-sorted predicate logic. A *knowledge base* defines a part of the knowledge that is used in one or more of the processes. Knowledge is represented by formulae in order-sorted predicate logic, which can be normalised by a standard transformation into rules.

3.2.2. Composition of Knowledge Structures

Information types can be composed of more specific information types, following the principle of compositionality discussed above. Similarly, knowledge bases can be composed of more specific knowledge bases. The compositional structure is based on the different levels of knowledge abstraction distinguished, and results in information and knowledge hiding.

3.3. Relation between Process and Knowledge Composition

Each process in a process composition uses knowledge structures. Which knowledge structures are used for which processes is defined by the relation between process composition and knowledge composition.

3.4. Generic Models and Reuse

Instead of designing each and every new agent application from scratch, an existing generic model can be used. Generic models can be distinguished for specific types of agents, of specific agent tasks and of specific types of multi-agent organisation. The use of a generic model in an application structures the design process: the acquisition of a conceptual model for the application is based on the generic structures in the model. A model can be generic in two senses:
- generic with respect to the *processes or tasks*
- generic with respect to the *knowledge structures*

Genericity with respect to processes or tasks refers to the level of process abstraction: a generic model abstracts from processes at lower levels. A more specific model with respect to processes is a model within which a number of more specific processes, at a lower level of process abstraction are distinguished. This type of refinement is called *specialisation*. Genericity with respect to knowledge refers to levels of knowledge abstraction: a generic model abstracts from more specific knowledge structures. Refinement of a model with

respect to the knowledge in specific domains of application, is refinement in which knowledge at a lower level of knowledge abstraction is explicitly included. This type of refinement is called *instantiation*.

Reuse as such, reduces the time, expertise and effort needed to design and maintain system designs. Which components, links and knowledge structures from the generic model are applicable in a given situation depends on the application. Whether a component can be used immediately, or whether instantiation, modification and/or specialisation is required, depends on the desired functionality. Other existing (generic) models can be used for specialisation of a model; existing knowledge structures (e.g., ontologies, thesauri) can be used for instantiation. Which models and structures are used depends on the problem description: existing models and structures are examined, rejected, modified, specialised and/or instantiated in the context of the problem at hand.

3.5. Process hiding and external observation of behaviour

The notion of process hiding in terms of process abstraction levels gives a sharp distinction in what is visible from outside an agent, and what happens internally. At the process abstraction level of the whole agent no reference can be made to the internal concepts. Only the actions and observation results are specified at this process abstraction level. This corresponds to what a human observer can observe from the behaviour (from the black box perspective): the 'stimuli', and the 'response behaviour'. For example, the term 'the animal shows a delayed response' refers to this process abstraction level, and leaves open by which internal processes the animal achieves its behaviour. Descriptions that refer to processes that are assumed to take place within the agent are at one process abstraction level lower. A (theoretically debatable) phrase such as 'the animal shows to have memory' refers to this lower process abstraction level, and entails an assumption on how things are arranged internally.

4. AN AGENT MODEL FOR PURELY REACTIVE BEHAVIOUR

An agent is *purely reactive* if it immediately responds to stimuli from its environment. Such agents are also called behaviour-based or situated agents; e.g., see [19]. These agents make their decisions based on a very limited amount of information, and simple situation-action rules. The stimuli can either consist of perceived changes in the external world or received communications from other agents. Changes in the external world are perceived by the agent by observation. The response behaviour of the agent affects its environment. Several architectures have been developed for reactive agents, see [1], [8], [16], [17]. In [19] an extensive overview of these architectures and the motivations behind them can be found.
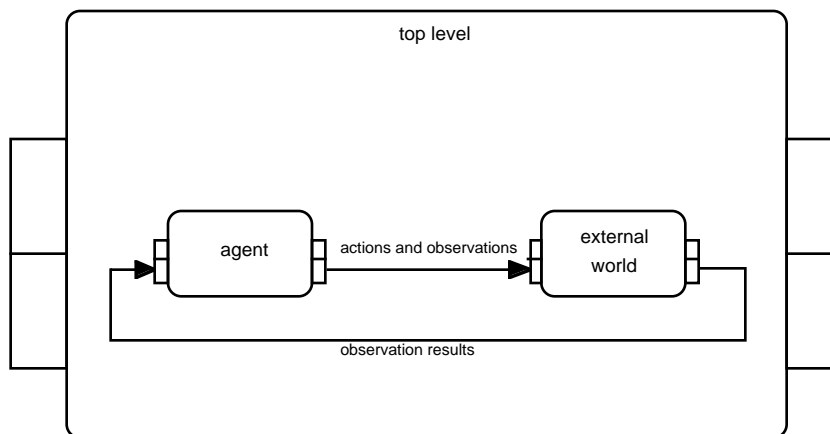

Figure 2: A generic agent model for purely reactive behaviour

4.1. Process Composition

For the design and implementation of the different models the compositional development method for multi-agent systems DESIRE has been used; see [4] for more details. A generic agent model for purely reactive behaviour developed earlier within the DESIRE environment (and applied in chemical process control) was reused. The (rather simple) agent system in this model consists of two components, one for the agent (of type A) and one for the external world with which it interacts (see Figure 2).

In the current domain, the observation information that plays a role describes that certain *objects* (cup1, cup2, food, screen, self) are at certain *positions* (i.e., p0, p1, p2). This is modelled by two sorts OBJECT and POSITION and a relation at position between these two sorts. Moreover, two types of *actions* can be distinguished: eat and goto some position. The latter type of actions is parameterized by positions; this can be modelled by a function goto from POSITION to ACTION. E.g., goto(p1) is the action to go to position p1. The action eat that is specified assumes that if the animal is at the position of the food, it can have the food: if a cup is covering the food, as part of the action eat the animal can throw the cup aside to get the food. Variables over a sort, e.g., POSITION, are denoted by a string, e.g., P, followed by : POSITION, i.e., P : POSITION is a variable over the sort POSITION. The unary relation to_be_performed is used to express the information that the agent has decided to *perform an action*; for example, to_be_perfomed(goto(p1)) expresses that the agent has decided to go to position p1. The relation observation_result is used to express the information that certain information has been acquired by *observation*; e.g., observation_result(at_position(food, p1), pos) expresses that the agent has observed that there is food at position p1, whereas the statement observation_result(at_position(food, p1), neg) expresses that the agent has observed that there is *no* food at position p1.

4.2. Knowledge composition

In this section, first the information types to specify the knowledge are defined, and next, using these information types the domain knowledge is specified to obtain the required functionality.
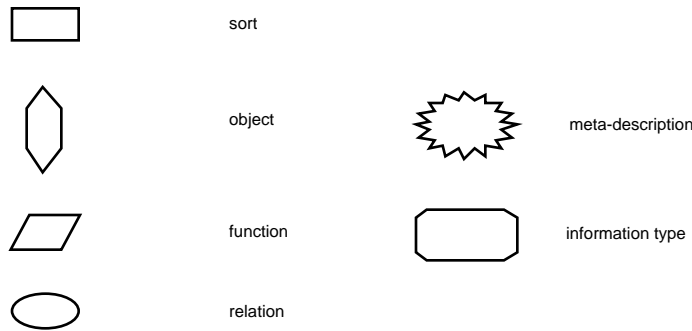

Figure 3: Information types: Legenda

4.2.1. Information types

Information types provide the ontology with which knowledge used in the processes can be expressed. Information types provide the *ontology* (or lexicon, or vocabulary) for the languages used in one (or more) components, knowledge bases and information links. In information type specifications the following concepts are used: sorts, objects, relations, functions, and meta-descriptions. Furthermore, information types can be composed from other information types.

Each concept is represented graphically, see Figure 3. The icon for information types is used as depicted in Figure 4 containing only the name of an information type, but also as depicted in Figure 5 containing the sorts, object, functions, relations, and meta-descriptions used in the design of that information type.
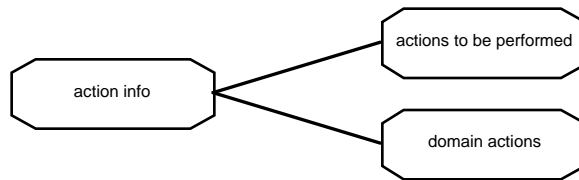
Figure 4: action info as a composition of a generic and domain specific information type

All information types are (either directly or indirectly) composed of (1) generic information types and (2) domain specific information types. Generic information types are fully specified within the generic model. Domain specific information types are defined by references; they are instantiated for a specific domain of application. For example, the information type action info is composed of the generic information type actions to be performed and the domain specific information type domain actions (see Figure 4). The generic information type actions to be performed enables the agent to reason about actions; see Figure 5.
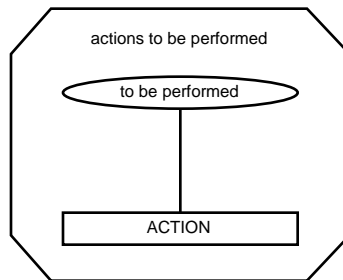

Figure 5: Generic information type: actions to be performed

The specific actions for a given domain of application are not specified within the generic model, but as part of the domain knowledge structures (see Figure 6).
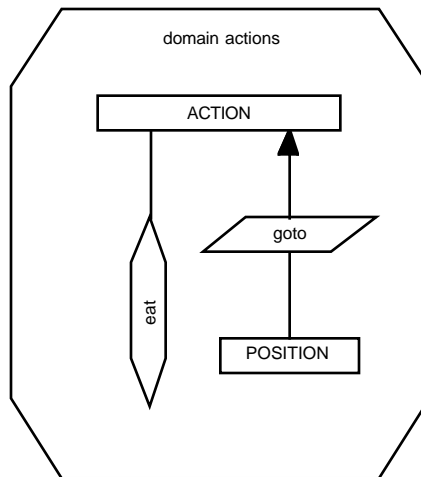

Figure 6: The information type domain actions

In a similar manner the information type observation result info is composed of the generic information types observation results and truth indication, and the domain specific information type domain meta-info.
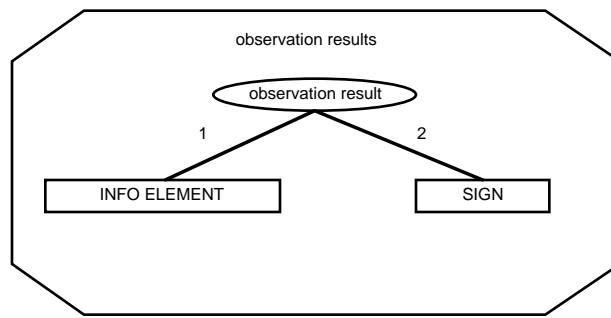
Figure 7:   Generic information type observation results

The generic information type observation results (see Figure 7) enables the agent to express statements on observation results. In the applications the observations are assumed *passive*: without taking any initiative, the agent automatically receives the observation results from the external world. The generic information type truth indication as presented in Figure 8 is also used in the information type observation results. By these information types it is possible to make statements about the process of observation of the state of the world in contrast to statements about the world. It is possible for the statement 'my observation result is that food is present at p2' to be true, while in the world state 'food is present at p2' is false. For example, a sensor could give the wrong information. Similarly, it could also be the other way around: the statement 'food is present at p2' can be true in the world state, while the statement 'my observation result is that food is present at p2' is false, simply because it was not observable, or at least I did not observe it. Note also that 'I did not observe that food is present at p2' means something different from 'I observed that no food is present at p2'. A statement of the form 'my observation result is that food is present at p2' cannot be expressed using the information type that describes the world. For example, the statement 'food is present at p2' is not adequate. Therefore, another structure is necessary to express statements about statements. Statements about statements are called *meta-level* statements. The statements that form the subjects of such meta-level statements are called *object level* statements.
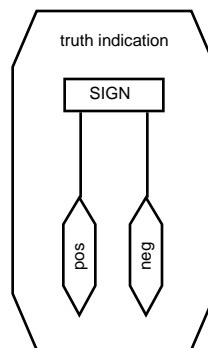

Figure 8: Generic information type truth indication

4.2.2. Domain knowledge

Assuming that food is offered at at most one position (for example, position p2), the stimulus-response behaviour of agent model A expresses that if the agent observes that there is food at any position and that no screen at position p0 separates the agent from this position, then it goes to this position. This knowledge has been modelled in the following form:

   if   observation_result(at_position(food, P:POSITION), pos)
    and   observation_result(at_position(screen, p0), neg)
    and   observation_result(at_position(self, P:POSITION), neg)
   then   to_be_performed(goto(P:POSITION))

if   observation_result(at_position(self, P:POSITION), pos)

and   observation_result(at_position(food, P:POSITION), pos)

then   to_be_performed(eat)

4.3. The Behaviour of the Purely Reactive Agent

The requirement imposed on agent A was that it shows the same behaviour for Situations 1 and 3 in the problem description: do nothing. Moreover, in Situation 2 the agent is required to go to the position of the food. The agent of type A indeed shows behaviour as expressed by the requirements. For a more detailed account of the behaviour, see Appendix A.
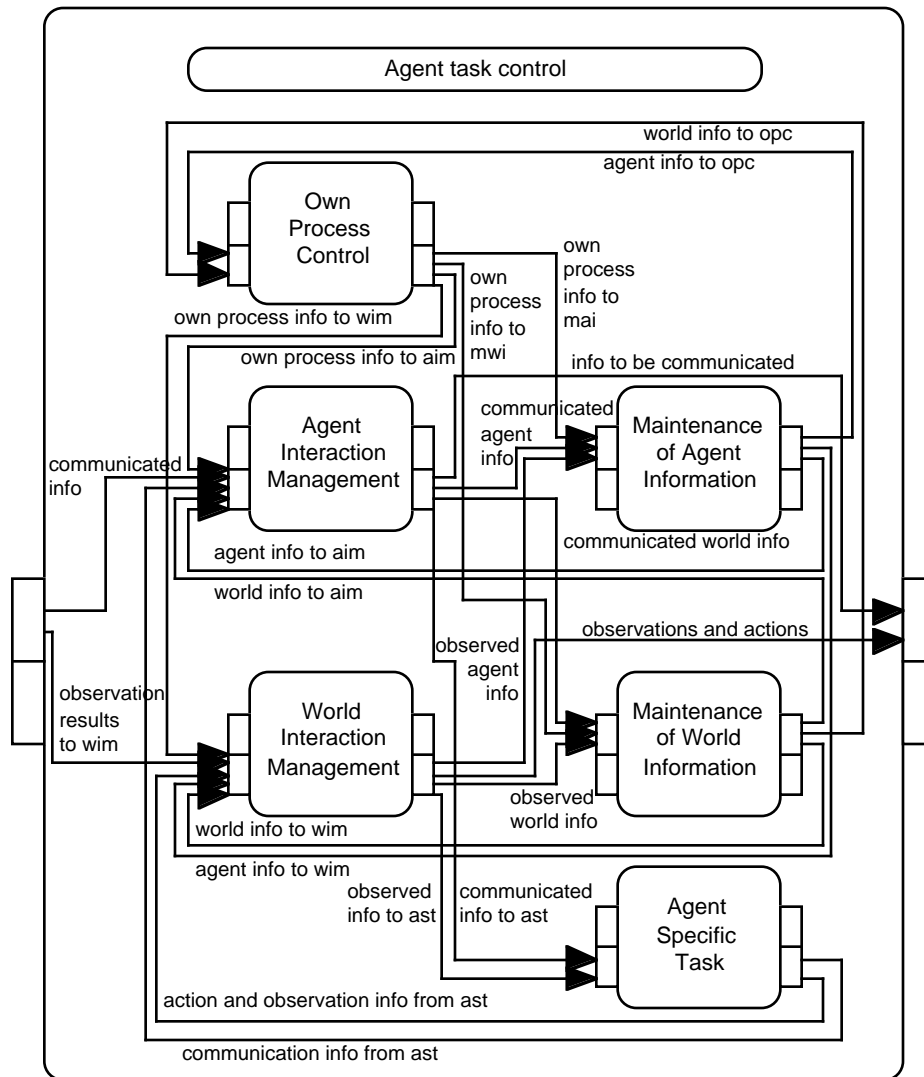

Figure 9: A generic agent model

5. AGENT MODELS FOR DELAYED RESPONSE AND PRO-ACTIVE BEHAVIOUR

To design an agent model that will show delayed response behaviour, the internal structure of the agent is made more complex. Within the agent a component maintenance of world information to maintain the observation results as beliefs (a memory) is distinguished from a component world interaction management that manages the interaction with the world. Moreover, if the agent has to generate its own goals in order to show pro-active behaviour, a component own process control is added, and if the agent has to show social behaviour, components are added to manage communication (agent interaction management) and to maintain beliefs on other agents (maintenance of agent information). The generic agent model depicted in Figure 9 (see also [7]) is composed of all of these components.

5.1. An Agent Model with Delayed Response Behaviour

For an agent with *delayed response behaviour* (type B), the component maintenance of world information is used, in addition to the component world interaction management. The only task performed by the component maintenance of world information is storage of observation information. No further knowledge is used within this component.
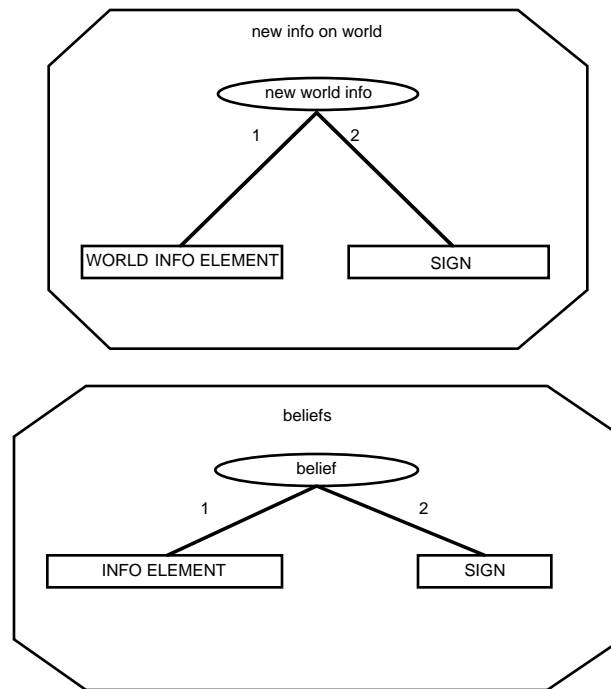

Figure 10: The information types new world info and belief info

The part of the knowledge of the component world interaction management that determines the actions is a variant of the knowledge used in agent model A. An additional part determines that the world information that was acquired by observation has *to be maintained*, expressed by the relation new_world_info.

     if   observation_result(I:INFO_ELEMENT, S:SIGN)
     then   new_world_info(I:INFO_ELEMENT, S:SIGN)

     if   belief(at_position(food, P:POSITION), pos)
       and   belief(at_position(screen, p0), neg)
       and   belief(at_position(self, P:POSITION), neg)
     then   to_be_performed(goto(P:POSITION))

```
if       belief(at_position(food, P:POSITION), pos)
  and    belief(at_position(self, P:POSITION), pos)
then     to_be_performed(eat)
```

An essential difference with the knowledge in agent model A is that in the knowledge above the relation observation result is replaced by the relation belief. Not only can information from direct observation be used, but also information retrieved from memory: all input information gets the status of belief, in contrast to observation. The behaviour of this agent model in comparison to the behaviours of the other models is discussed in Section 8.

5.2. An Agent Model with Pro-active Behaviour

The third agent model to be discussed is a model for a *pro-active agent* (type C). A pro-active agent does not simply respond to stimuli, neither immediately, nor delayed. In addition to the observation information, its so-called *motivational attitudes* (such as goals and plans) play an important role (e.g., see [25]) in determining its actions. These motivational attitudes can be based on the agent's own character (for example, an agent's character may be that it always wants to eat, or that it is totally apathic), but also on specific aspects of the agent's own state, such as being hungry, or being depressed. To determine the motivational attitudes of the agent, the component own process control is used; additional knowledge structures are introduced for this new component.
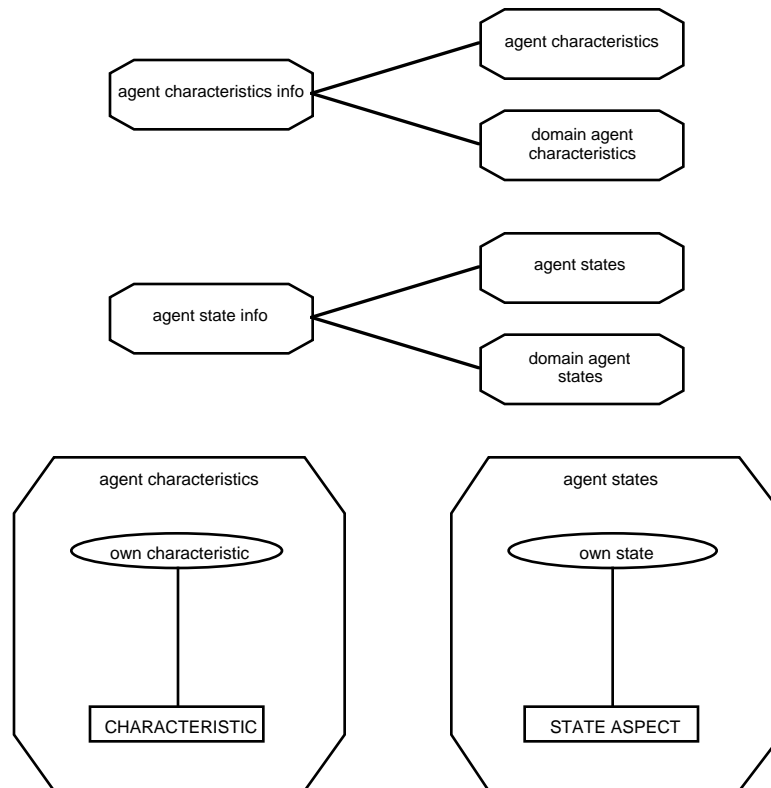
Figure 11: Composition and generic information types for agent properties

In addition to the existing information types of agent model B, information types are required for knowledge on own process control; these information types express information on (see Figure 11 and 12):

- the agent's beliefs
- aspects of the agent's own state, such as being hungry, or being depressed, and specific characteristics of the agent, such as always eager to eat, or totally apathic
- the agent's goals, such as be fed, just hang around, find food, or get food inside.
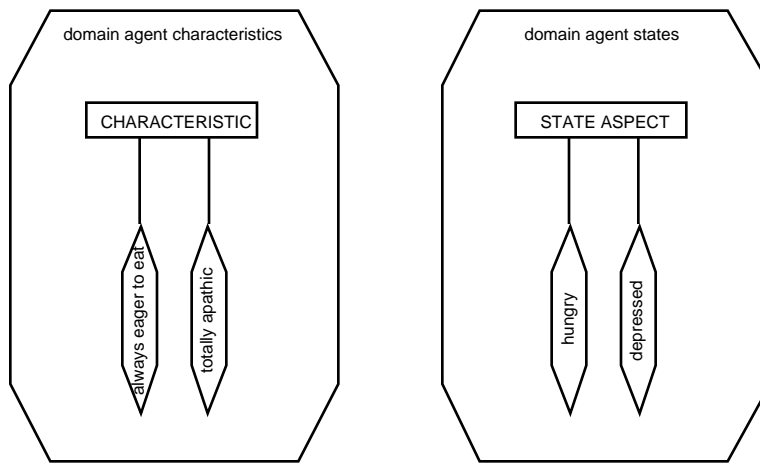
Figure 12: Domain-specific information types for agent properties

Information on the agent's *own state* can be expressed using the unary relation own_state; for example, the statement own_state(hungry) expresses that the agent is hungry. The agent's *own characteristics* can be expressed using the unary relation own_characteristic; e.g., the statement own_characteristic(totally_apathic) expresses the information that the agent is totally apathic. The *goal* that has been selected by the agent is expressed using the unary relation selected_goal. The knowledge to be used in the component own process control models (see Figure 13):

- an agent that is always eager to eat, always selects the goal be fed
- any not apathic agent that is hungry or depressed selects the goal be fed
- a totally apathic agent never selects a goal
- an agent which has be fed as a goal, selects the goal get food inside if it has a belief that food is present at a specific position; in the other case it selects the goal find food
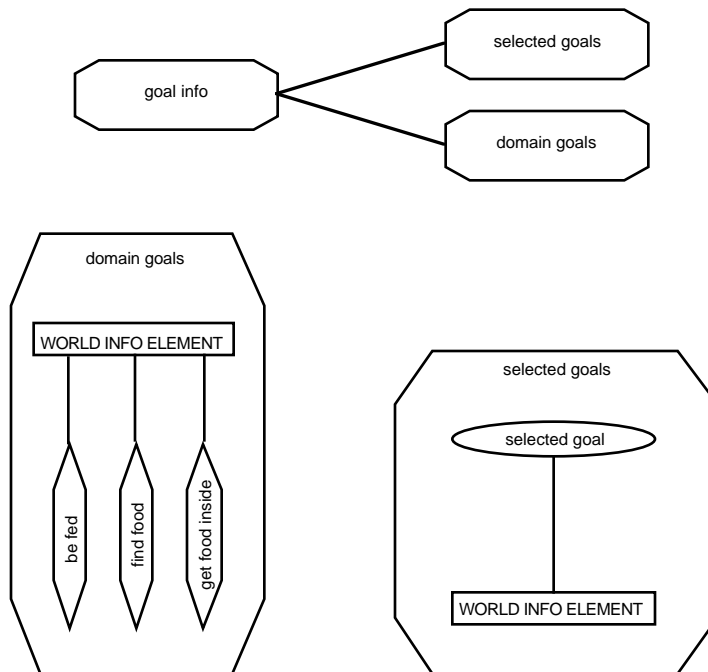

Figure 13: Generic and domain-specific information types for the agent's goals

The knowledge used in own process control knowledge can formulated in a concise form as follows:

if   own_characteristic(always_eager_to_eat)
then   selected_goal(be_fed)

if   own_state(hungry)
  and   **not** own_characteristic(totally_apathic)
then   selected_goal(be_fed)

if   own_state(depressed)
  and   **not** own_characteristic(totally_apathic)
then   selected_goal(be_fed)

if   selected_goal(be_fed)
  and   belief(at_position(food, P:POSITION), pos)
then   selected_goal(get_food_inside)

if   selected_goal(be_fed)
  and   **not** belief(at_position(food, p1), pos)
  and   **not** belief(at_position(food, p2), pos)
then   selected_goal(find_food)

Depending on the type of agent modelled, some facts can be added to this knowledge base, for example in the agent of type C:

    own_characteristic(always_eager_to_eat)

(alternatively, for example, own_state(hungry), **not** own_characteristic(totally_apathic) could be specified, or own_characteristic(totally_apathic)).

Depending on the agent characteristics specified, the agent determines one or more goals. To actually show certain pro-active behaviour, also suitable knowledge has to be specified on which actions are to be performed for a given goal. Moreover, not only actions can be performed in a pro-active manner, but also observations. In particular, if the animal visits a position for which it is not known yet whether these is food, it can decide to explore the position, e.g., by putting the cup a bit aside and/or activating its smell sensor (sniffing). To model active observations, the information type observation info is used, defining the relation to_be_observed on sort WORLD_INFO_ELEMENT. For example, the atom

        to_be_observed(at_position(food, p1))

means that the agent has decided to observe whether food is present at position p1. Knowledge to initiate actions and observations is modelled in the component world interaction management. To determine actions related to the goal get food inside, two possible cases are considered:
- the agent believes that food is present at its own position; in this case it simply can start eating
- the agent believes that no food is present at its own position, but it believes that food is present at another position; in this case the agent can go to such a position (and if it arrives there it can start eating, according to the previous item)

This knowledge is expressed in a concise form as follows:

if   selected_goal(get_food_inside)
  and   belief(at_position(food, P:POSITION), pos)
  and   belief(at_position(self, P:POSITION), pos)
then   to_be_performed(eat)

```
if        selected_goal(get_food_inside)
  and     belief(at_position(self, P1:POSITION), pos)
  and     belief(at_position(food, P1:POSITION), neg)
  and     belief(at_position(food, P2:POSITION), pos)
  and     belief(at_position(screen, p0), neg)
then      to_be_performed(goto(P2:POSITION))
```

The goal get food inside assumes that the agent already knows at least one position where food is present. If this is not the case, the goal find food may be selected by the agent. To determine the actions or observations for the goal find food, the following cases are considered:

- the agent does not know whether food is present at its own position; then the observation to explore the own position is initiated (which determines whether food is present at the agent's own position)
- the agent believes that no food is present at its own position, and it does not know whether food is present at positions p1 and p2; in this case the action go to p1 is selected (and if it arrives there it can start exploring it, according to the previous item)
- the agent believes that no food is present at its own position and at position p1; it does not know whether food is present at p2; in this case the agent goto p2 is selected (and if it arrives there it can start exploring the position, according to the first item)

This knowledge is expressed in a concise form as follows:

```
if        selected_goal(find_food)
  and     belief(at_position(self, P:POSITION), pos)
  and     not  belief(at_position(food, P:POSITION), pos)
  and     not  belief(at_position(food, P:POSITION), neg)
then      to_be_observed(at_position(food, P:POSITION))
```

```
if        selected_goal(find_food)
  and     belief(at_position(self, P:POSITION), pos)
  and     belief(at_position(food, P:POSITION), neg)
  and     not  belief(at_position(food, p1), neg)
  and     not  belief(at_position(food, p1), pos)
  and     not  belief(at_position(food, p2), neg)
  and     not  belief(at_position(food, p2), pos)
then      to_be_performed(goto(p1))
```

```
if        selected_goal(find_food)
  and     belief(at_position(self, P:POSITION), pos)
  and     belief(at_position(food, P:POSITION), neg)
  and     belief(at_position(food, p1), neg)
  and     not  belief(at_position(food, p2), neg)
  and     not  belief(at_position(food, p2), pos)
then      to_be_performed(goto(p2))
```

For more details of the behaviour of the agent types B and C, see Appendices B and C.

6. AN AGENT MODEL WITH SOCIAL BEHAVIOUR

To obtain *social behaviour* (an agent model of type D), also the components agent interaction management and maintenance of agent information are used in the model.

A social agent is able to receive incoming communication and to generate outgoing communication. The generic information types for communication are depicted in Figure 14. By these information types it is possible to make statements about the process of communication (in contrast to, for example, statements about the world). It is possible for the statement 'I was told that the pressure is high' to be true, while in the world state 'the pressure is high' is false: the other agent may simply not tell the truth. It could also be the other way

around: the statement 'the pressure is high' could be true in the world state, while the statement 'sombody told me that the pressure is high' is false, simply because nobody told me. Note also that 'he did not tell me that the pressure is high' does not mean the same as 'he told me that the pressure is not high'. Similar to statements about observation, statements about comunication are meta-level statements.
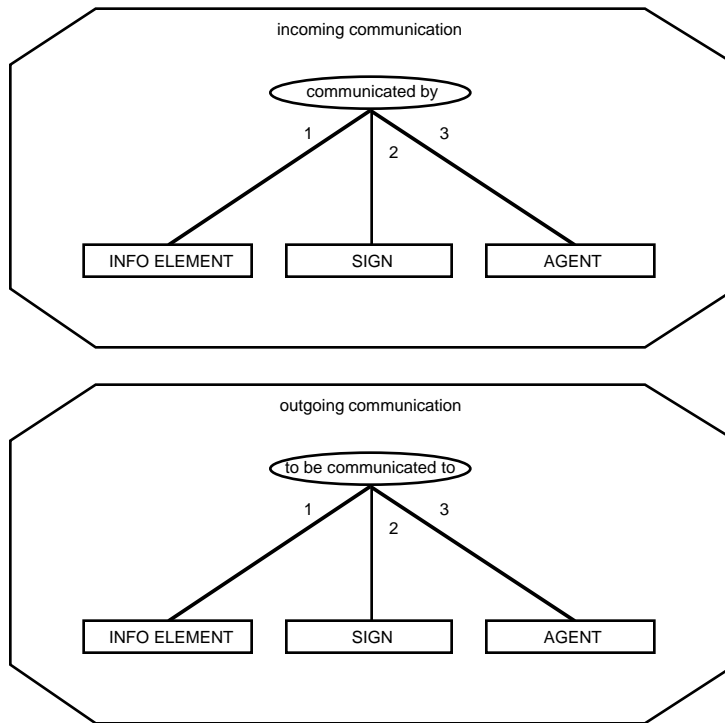

Figure 14: Generic information types on communication

The information communicated to the agent may be used to extend or update an agent's beliefs both on the world or on other agents. The information received is analysed, selected and prepared to be stored as information either on the world or on other agents; the related information types are depicted in Figure 15.
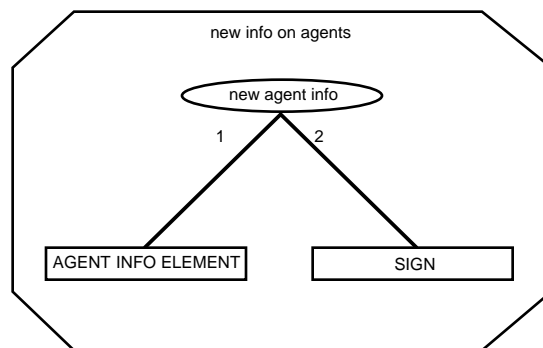

Figure 15: Generic information type maintenance on agents

In the component **agent interaction management** knowledge is specified that identifies new communicated knowledge about other agents:

if	communicated_by(I_want_food, pos, A:AGENT)
then	new_agent_info(wants_food(A:AGENT))

Here, the statement communicated_by(I_want_food, pos, A:AGENT) expresses that the information I_want_food *has been communicated* (positively) by the agent A:AGENT. This *new agent information* (expressed using the relation new_agent_info) is stored in the component maintenance of agent information: the knowledge used in maintenance of agent information specifies the hierarchy between different animals, and whether another animal that is present wants the food. For example, if an1 is an animal which is present and is *higher in the hierarchy*, then this can be specified within the knowledge base used in maintenance of agent information of the agent self as higher_than(an1, self). It is also possible to model this information in a dynamic form, as an outcome of earlier experiences (fights). If the other animal wants the food, within the component maintenance of world information of the agent self it is derived that the food is protected, using the knowledge

if	wants_food(A:AGENT)
and	higher_than(A:AGENT, self)
then	food_protected

Within the knowledge elements used in the component world interaction management an additional condition **not** belief(food_protected, pos) is specified. For more details of the behaviour of agent type D, see Appendix D.

7. AN AGENT MODEL WITH ADAPTIVE BEHAVIOUR

The agent models presented in the previous sections showed different types of behaviour, but they are not able to adapt their behaviour. In this section an agent model for adaptive animal behaviour is presented. To be able to learn from experiences it is important that for a given situation more than one option for an action is generated. By making decisions about the action that is selected from the options the animal can experiment with its behaviour, and, after eveluation, adapt its decision making process.

7.1. *Process composition of the adaptive agent model*

To be able to address and evaluate the action selection process explicitly, in this agent model the following more specialised processes are distinguished:
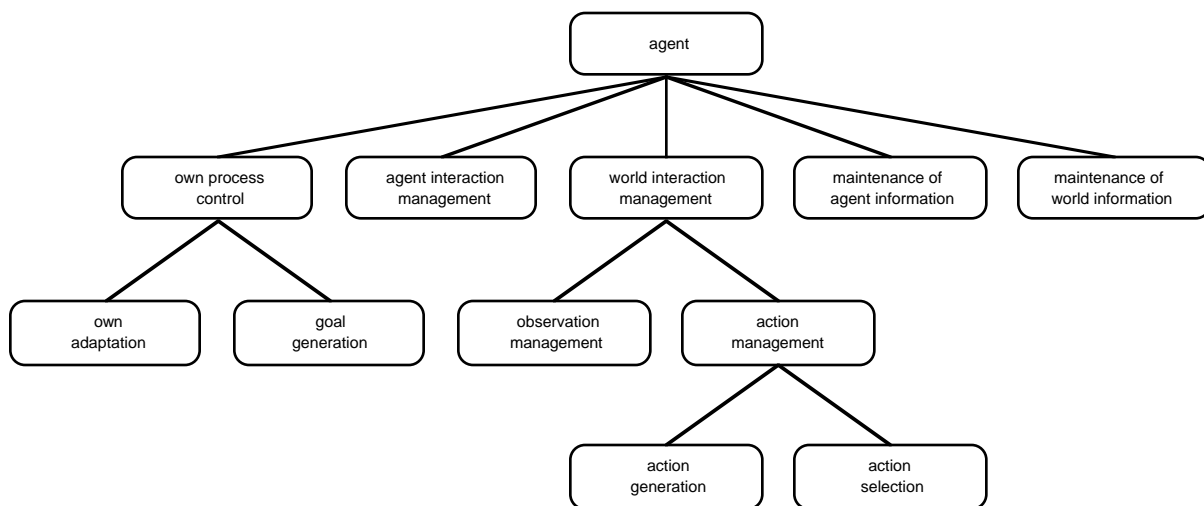

Figure 16: Processes and their abstraction levels for the adaptive agent model

- *action generation*
  This process generates a number of alternative behaviours for a given (observed or believed) situation.
- *action selection*
  This process selects one option from the set of alternative behaviours, using selection knowledge based on earlier experiences.
- *own adaptation*
  This process evaluates the selected action upon its successfulness, and adapts the knowledge used to make a selection. In particular, if the action was successful, in the future it will be chosen more often (under comparable circumstances), and the other options less often. If the action was not successful, it will be chosen less often.

The first two of these processes are modelled as a specialisation of world interaction management, whereas the latter process is modelled as a subprocess of own process control. The processes and their four abstraction levels are shown in Figure 16.

The process composition relations are straighforward. The relevance ordering relations are transferred from the component own adaptation via the input interface of own process control to world interaction management and within this component to action selection. Moreover, the actions generated by action generation are transferred to action selection as well.

7.2. *Knowledge composition of the adaptive agent model*

In this section the information types and knowledge bases for the adaptive agent model are discussed.

7.2.1. *Information types*

For the adaptive agent, information types are required to specify which options for actions are generated in a given situation, how the decision is made to choose between the options, and how this choice is inflenced by experiences. The options for actions to achieve a given goal are expressed by the generic information type action alternatives; see Figure 17.
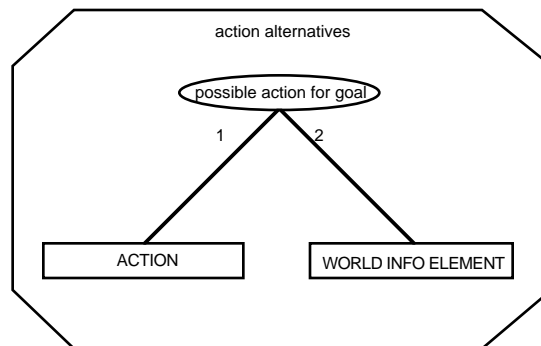

Figure 17: The information type action alternatives

The selection process first determines which actions are to be ignored; the remaining actions are considered appropriate: the generic information type action selection info; see Figure 18. Within this selection process the domain specific information types food relevance info and action relevance are used; see Figure 19. The agent's experiences are formalised by a (dynamic) linear ordering of the colours: the higher a colour in this ordering, the more successful experiences the agent has in finding food at positions of that colour.
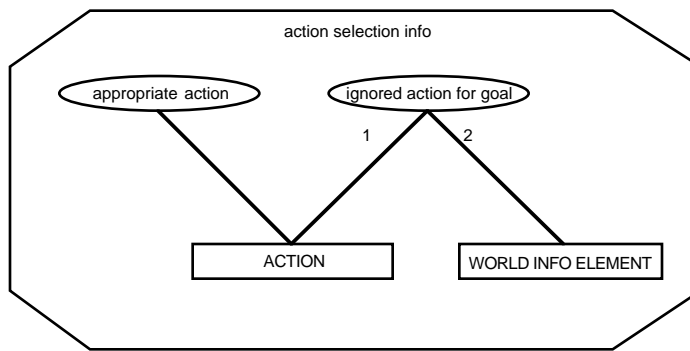
Figure 18: The information type action selection info

This linear ordering is represented by

has_higher_food_relevance(C1:COLOUR, C2:COLOUR)

which means that colour C1:COLOUR is higher than colour C2:COLOUR in the linear ordering. The linear ordering is maintained and updated in the component own adaptation within own process control. It is only used in the component action selection.
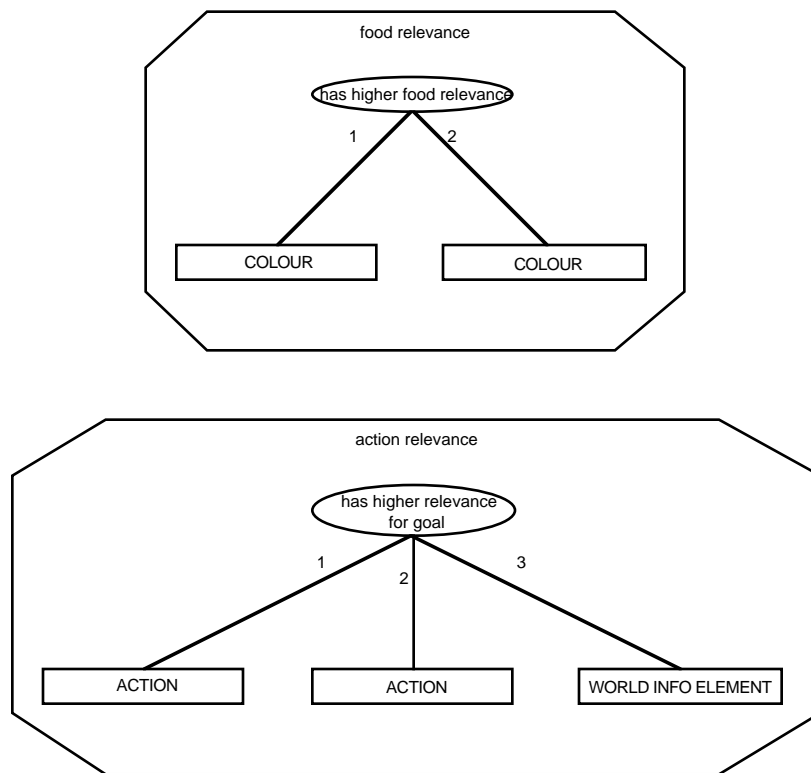

Figure 19: The information types food relevance and action relevance

The update of the relevance information makes use of the generic information type reinforcement info (see Figure 20), and the domain specific information type food precedence info (see Figure 21) are used. The food precedence relation expresses the immediate neighbours in the linear ordering on food relevance.
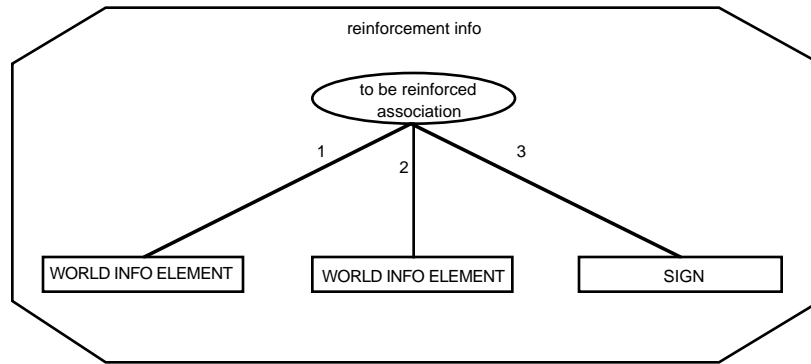
Figure 20: The information type reinforcement info

Besides the information types introduced above, the domain information is extended by the sort COLOUR and the relation is_colour_at_position between COLOUR and POSITION. For more interesting experiments, also the number of positions was extended.

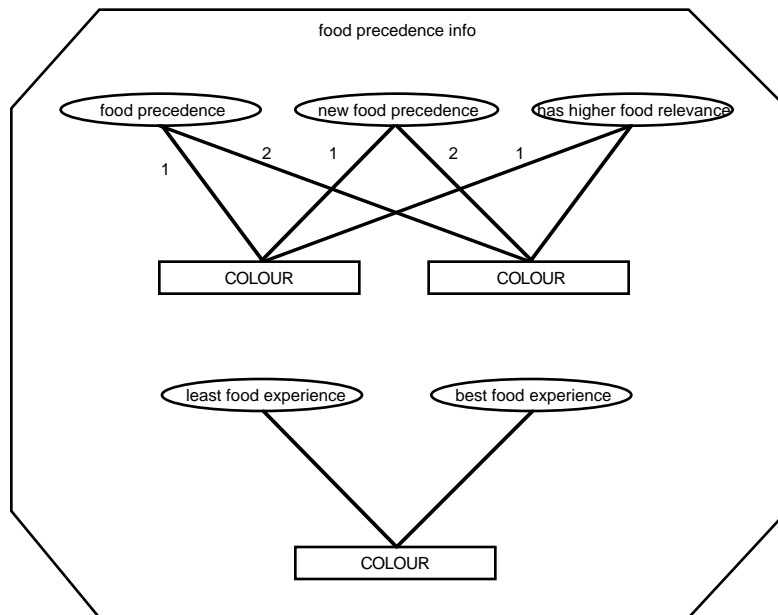Figure 21: The information type food precedence info

7.2.2. Knowledge used

The following knowledge bases are used in the different components.

Knowledge used within observation management
The knowledge used within observation management is simple. For the goal find food, if it is unknown whether food is present at the own position, the observation to explore this position is always selected:

if	selected_goal(find_food)
and	belief(at_position(self, P1:POSITION), pos)
and	**not** belief(at_position(food, P1:POSITION), neg)
and	**not** belief(at_position(food, P1:POSITION), pos)
then	to_be_observed(at_position(food, P1:POSITION))

Knowledge used within action generation

The knowledge used within action generation is kept simple. For the goal find food
(1) if it is known that there is no food at the own position, all actions that are generated, are those of the form goto(P:POSITION) for the positions P:POSITION for which the agent has no belief on whether there is food present, and
(2) if it is known that food is present at the own position, the action eat is generated.

if	selected_goal(find_food)
and	belief(at_position(self, P1:POSITION), pos)
and	belief(at_position(food, P1:POSITION), neg)
and	**not** belief(at_position(food, P2:POSITION), neg)
and	**not** belief(at_position(food, P2:POSITION), pos)
then	possible_action_for_goal(goto(P2:POSITION), find_food)

if	selected_goal(get_food_inside)
and	belief(at_position(self, P:POSITION), pos)
and	belief(at_position(food, P:POSITION), pos)
then	possible_action_for_goal(eat, get_food_inside)

Knowledge used within action selection

For the action eat, the generated action is always selected:

if	selected_goal(find_food)
and	possible_action_for_goal(explore_position, find_food)
then	appropriate_action(explore_position)

if	selected_goal(get_food_inside)
and	possible_action_for_goal(eat, get_food_inside)
then	appropriate_action(eat)

For the other case the knowledge used within action selection is based on what the agent has experienced. From the colour relevancy ordering relations, within action selection action relevance ordering relations (relative for a given goal) are derived:

if	belief(is_colour_at_position(C1:COLOUR, P1:POSITION), pos)
and	belief(is_colour_at_position(C2:COLOUR, P2:POSITION), pos)
and	has_higher_food_relevance(C1:COLOUR, C2:COLOUR)
then	has_higher_relevance_for_goal(goto(P1:POSITION), goto(P2:POSITION), find_food))

Using this total ordering of relevancy of actions the decision is made to ignore all actions for which at least two more relevant actions exist:

if	possible_action_for_goal(A1:ACTION, G:WORLD_INFO_ELEMENT)
and	possible_action_for_goal(A2:ACTION, G:WORLD_INFO_ELEMENT)
and	possible_action_for_goal(A3:ACTION, G:WORLD_INFO_ELEMENT)
and	has_higher_relevance_for_goal(A1:ACTION, A2:ACTION, G:WORLD_INFO_ELEMENT))
and	has_higher_relevance_for_goal(A2:ACTION, A3:ACTION, G:WORLD_INFO_ELEMENT))
then	ignored_action_for_goal(A3:ACTION, G:WORLD_INFO_ELEMENT)

Finally, all actions relevant for the selected goal, which were not ignored, are derived as appropriate:

  if   selected_goal(G:WORLD_INFO_ELEMENT)
   and possible_action_for_goal(A:ACTION, G:WORLD_INFO_ELEMENT)
   and **not** ignored_action_for_goal(A:ACTION, G:WORLD_INFO_ELEMENT)
  then   appropriate_action(A:ACTION)

From these appropriate actions only one is derived, at random.

The action selection approach discussed here is just one possible approach. Another, more simple approach would be to just take the highest action in the relevance ordering. However, this more simple decision model allows the agent less space for experimentation, and implies slower adaptation in certain circumstances. Also numerical approaches are possible; for an example, see Section 7.3 below.

Knowledge used within own adaptation
Within the component own adaptation the linear order of relevancy of colours for food is maintained and updated, on the basis of experiences. First experiences are evaluated: if there was food found at a position with a certain colour, a positive reinforcement has to be implemented for this colour, otherwise a negative reinforcement:

  if   observation_result(at_position(food, P:POSITION), S:SIGN)
   and belief(is_colour_at_position(C:COLOUR, P:POSITION), pos)
  then   to_be_reinforced_association(is_colour_at_position(C:COLOUR, P:POSITION),
             at_position(food, P:POSITION), S:SIGN)

The positive reinforcement is modelled by movingthe colour upwards in the linear ordering one place, as long as it is not the highest colour. If it already was the highest colour, then nothing changes. If it was second highest, then it becomes the highest. The relation food_precedence(C1:COLOUR, C2:COLOUR) represents the immediate succession in the linear ordering.

  if   to_be_reinforced_association(is_colour_at_position(C:COLOUR, P:POSITION),
               at_position(food, P:POSITION), pos)
   and food_precedence(C0:COLOUR, C1:COLOUR)
   and food_precedence(C1:COLOUR, C:COLOUR)
   and food_precedence(C:COLOUR, C2:COLOUR)
  then   new_food_precedence(C0:COLOUR, C:COLOUR)
   and new_food_precedence(C:COLOUR, C1:COLOUR)
   and new_food_precedence(C1:COLOUR, C2:COLOUR)

  if   to_be_reinforced_association(is_colour_at_position(C:COLOUR, P:POSITION),
               at_position(food, P:POSITION), pos)
   and best_food_experience(C0:COLOUR)
   and food_precedence(C0:COLOUR, C:COLOUR)
   and food_precedence(C:COLOUR, C1:COLOUR)
  then   new_best_food_experience(C:COLOUR)
   and new_food_precedence(C:COLOUR, C0:COLOUR)
   and new_food_precedence(C0:COLOUR, C1:COLOUR)

If the colour was the lowest, the second lowest becomes the lowest:

if	to_be_reinforced_association(is_colour_at_position(C:COLOUR, P:POSITION),
	at_position(food, P:POSITION), pos)
and	least_food_experience(C:COLOUR)
and	food_precedence(C2:COLOUR, C:COLOUR)
and	food_precedence(C1:COLOUR, C2:COLOUR)
then	new_least_food_experience(C2:COLOUR)
and	new_food_precedence(C:COLOUR, C2:COLOUR)
and	new_food_precedence(C1:COLOUR, C:COLOUR)

For negative reinforcement, the opposite of the above takes place. The colour is moved downwards in the linear ordering, as long as it is not the lowest colour. If it already was the lowest colour, then nothing changes. If it was second lowest, then it becomes the lowest. If the colour was the highest, the second highest becomes the highest:

If the colour was the lowest, the second lowest becomes the lowest:

if	to_be_reinforced_association(is_colour_at_position(C:COLOUR, P:POSITION),
	at_position(food, P:POSITION), neg)
and	best_food_experience(C:COLOUR)
and	food_precedence(C:COLOUR, C1:COLOUR)
and	food_precedence(C1:COLOUR, C2:COLOUR)
then	new_least_food_experience(C1:COLOUR)
and	new_food_precedence(C1:COLOUR, C:COLOUR)
and	new_food_precedence(C:COLOUR, C2:COLOUR)

if	to_be_reinforced_association(is_colour_at_position(C:COLOUR, P:POSITION),
	at_position(food, P:POSITION), neg)
and	food_precedence(C1:COLOUR, C2:COLOUR)
and	food_precedence(C:COLOUR, C1:COLOUR)
and	food_precedence(C0:COLOUR, C:COLOUR)
then	new_food_precedence(C:COLOUR, C2:COLOUR)
and	new_food_precedence(C1:COLOUR, C:COLOUR)
and	new_food_precedence(C0:COLOUR, C1:COLOUR)

if	to_be_reinforced_association(is_colour_at_position(C:COLOUR, P:POSITION),
	at_position(food, P:POSITION), neg)
and	least_food_experience(C1:COLOUR)
and	food_precedence(C:COLOUR, C1:COLOUR)
and	food_precedence(C0:COLOUR, C:COLOUR)
then	new_least_food_experience(C:COLOUR)
and	new_food_precedence(C1:COLOUR, C:COLOUR)
and	new_food_precedence(C0:COLOUR, C1:COLOUR)

After updating the precedence relations by the new ones, by taking the transitive closure a total ordering is derived:

if	food_precedence(C1:COLOUR, C2:COLOUR)
then	has_higher_food_relevance(C1:COLOUR, C2:COLOUR)

if	food_precedence(C1:COLOUR, C2:COLOUR)
and	has_higher_food_relevance(C2:COLOUR, C3:COLOUR)
then	has_higher_food_relevance(C1:COLOUR, C3:COLOUR)

For more details of the behaviour of agent type E, see Appendix E.

7.3. A numerical approach based on weight values

An alternative approach that can be built in the same agent model makes use of numerical weight values $w_E(a)$ for actions under given environmental circumstances E. As a special case, the weight values can be normalised to overall sum 1:

$$\Sigma_b \; w_E(b) = 1$$

Within the component action selection the selection can be made by making a random choice based on the probability distribution

$$w_E(a) \, / \, \Sigma_b \; w_E(b)$$

over the actions a (in the case the overall sum is 1, the denominator can be left out).

Within the component own adaptation, the update for a successful experience for action a can be done by (here λ is a factor between 0 and 1, for example 0.9):

$$
\begin{aligned}
1 - w'_E(a) &= \lambda \, (1 - w_E(a)) \\
w'_E(b) &= \lambda \, w_E(b) \quad \text{for } b \neq a
\end{aligned}
$$

The update for an unsuccessful experience for action a can be done by, for example:

$$
\begin{aligned}
w'_E(a) &= \lambda \, w_E(a) \\
1 - w'_E(b) &= \lambda \, (1 - w_E(b)) \qquad \text{for } b \neq a
\end{aligned}
$$

These formulae guarantee that, if the sum is 1 (in this, most simple, case the weights themselves can be used as a probability distribution) it will remain 1 after updates. For example, when $\lambda = 0.9$ and

$$
\begin{aligned}
w_E(a_1) &= 0.3 \\
w_E(a_2) &= 0.3 \\
w_E(a_3) &= 0.4
\end{aligned}
$$

then after a successful experience on a_1, the update results in:

$$
\begin{aligned}
w_E(a_1) &= 0.37 \\
w_E(a_2) &= 0.27 \\
w_E(a_3) &= 0.36
\end{aligned}
$$

8. OVERVIEW OF THE BEHAVIOUR OF THE DIFFERENT AGENT MODELS

In this section the different types of behaviour of the agent models are presented and compared.

8.1. Global differences in behaviour

In Table 1 below the global differences in behaviour of the agent models are summarised. The different variants of behaviour depicted in this table indeed satisfiy the requirements expressed in Section 2.

8.2. More specific traces

In the Appendices A to E for each of the agent models a more detailed trace is shown. The information shown follows the graphical intereface that has been made for demonstration purposes.

First, in Appendix A a behaviour trace of agent model A is presented for Situation 3. In this trace the rows represent states (states of the world and mental states of the animal), and all steps in the process are depicted as transitions from one of the rows to the next row; for example: observation (from row 0 to 1), an event in the world (appearance of the cup, from 1 to 2), observation (making the mental model of the world state up to date by removing the food and adding the cup, from 2 to 3), an event in the world (disappearance of the screen, from 3

to 4), observation (taking out the screen of the mental model, from 4 to 5). It is shown that at each time point, the animal only has an image of the current world situation, not of the past. Therefore the animal is not successful in reaching the food.

	agent A	*agent B*	*agent C*	*agent D*	*agent E*
situation 1 (*no food*)	do nothing	do nothing	look for food at random	look for food at positions without higher competitors	look for food at places with relevant colours
situation 2 (*visible food*)	go to food	go to food	go to food	go to food at positions without higher competitors	go to food
situation 3 (*visible food made invisible*)	do nothing	go to food	go to food	go to food at positions without higher competitors	go to food
situation 4 (*invisible food*)	do nothing	do nothing	look for food at random	look for food at positions without higher competitors	look for food at places with relevant colours; learn from experience

Table 1: Global overview of the different behaviours

Next, in Appendix B behaviour of agent model B is shown for the same Situation 3. Also in this trace all steps in the process are depicted as transitions from one of the rows to the next row; for example: observation (from row 0 to 1), an event in the world (appearance of the cup, from 1 to 2), observation (this time the food remains in the mental model and the cup is added, from 2 to 3), action in the world (disappearance of the screen, from 3 to 4), observation (from 4 to 5), generating an action (the action go to p2, from 5 to 6), execution of the generated action (from 6 to 7). In this case the animal maintains beliefs about the world, and therefore still has the information about the position of the food, after the food was covered. It is shown that this animal is successful in reaching the food.

In Appendix C the behaviour of agent model C is shown for Situation 4. For shortness, in this trace (and in the traces for agent models D and E) a more condensed presentation is used. As in the previous traces changes of the world state are represented by a transition from one row to the next row. However, the observations following changes of the world state and also the mental steps following the observations are depicted in the same row as the new world state occurs. This trace shows that the animal first examiones position p1, without success, and next position p2, with success. It is shown how not only beliefs but also goals play a role in the determination of actions.

Appendix D describes behaviour of agent model D. First the agent visits p1, as in the case of model C. Next (in row 3) it waits, because at the other position, p2, a competitor is observed which is higher in the hierarchy. It is shown that the agent is not going to this position as long as a higher competitor is present there, but as soon as this other animal leaves (depicted in row 4), it is visiting that position. Note that the short phrase 'higher competitor at p2' actually refers to a combination of two types of information: (1) another animal is at position p2, (2) this other animal is higher than me. The first type of information is about the world, whereas the second type of information is about other agents.

Appendix E shows adaptive behaviour based on reinforcement in Situation 4. In the initial relevance ordering of colours red-green-blue, red is the highest, then green, and blue is the lowest in relevance (see row 0). If nothing is known about food at the positions p1, p2 and p3, then going to any of these positions is an option. The relevance ordering of actions is just copied from the relevance ordering of colours. From the highest two,

one is selected at random: p1, with a green colour (see row 1). From exploring this position (see row 2) a negative experience results. This leads to adaptation of the colour relevance ordering: green is pushed one position down, and blue pops up to the second position (see row 3, middle column). On the basis of this new ordering a decision is made about the next position to visit. Only the two candidates p2, p3 remain because it is known that at p1 no food is available. From these two, although going to p3 has the highest relevance, going to position p2, with the blue colour, is chosen at random (see row 3, rightmost column). Here a positive experience results. This again leads to adaptation of the relevance ordering of colours: blue and red are interchanged (see row 4). After eating (row 5), which makes the food disappear, at last p3 (with colour red) is visited (see row 6) which leads to a negative experience. Based on this experience the relevance ordering of colours is adapted: red and green are interchanged (not depicted). At this point the relevance ordering is blue-red-green. On the basis of this new relevance ordering a next session can be started.

9. CONCLUSIONS

In this paper it is shown how different types of animal behaviour can be modelled and simulated at a conceptual level on the basis of the compositional multi-agent development method DESIRE (cf. [4]). Different (variants of) reusable compositional agent models were used to model the different required behaviours. The advantage of this approach is that the models are designed at a high conceptual level, in terms of the processes, information and knowledge that is needed, and abstracting from implementation details. Nevertheless they can be executed by the DESIRE software environment. Besides the simulation of animal behaviour discussed in this paper, a variety of other applications have been developed using DESIRE. Some recent multi-agent applications can be found in [3] (negotiation between agents), [5] (simulation of a society of agents), [6] (distributed work flow and agenda scheduling), and [15] (agents in brokering processes).

Simulation of animal behaviour is an interesting type of application for multi-agent systems. Both areas can benefit from a more extensive study of this type of application. The area of multi-agent systems can benefit from the more detailed analyses and distinctions that have been made for different types of animal behaviour (see the Introduction). The study of animal behaviour can benefit from software tools for agent modelling at a conceptual level that support simulation. Moreover, formal techniques in the area of verification can be used to analyse and formalise behaviour properties of animals and their logical relations; e.g., [9], [14]. These formalisations can be a basis for the development of a theory of animal behaviour.

ACKNOWLEDGEMENTS

REFERENCES

1. Agre, P.E., and Chapman, D., Pengi: an Implementation of a Theory of Activity. In: *Proceedings of the sixth National Conference of the American Association for Artificial Intelligence (AAAI-87)*, Morgan Kaufmann, 1987, pp. 268-272.
2. Allen, C., and Bekoff, M., *Species of Mind: the philosophy and biology of cognitive ethology*. MIT Press, 1997.
3. Brazier, F.M.T., Cornelissen, F., Gustavsson, R., Jonker, C.M., Lindeberg, O., Polak, B., and Treur, J., Agents Negotiating for Load Balancing of Electricity Use. In: M.P. Papazoglou, M. Takizawa, B. Krämer, and S. Chanson (eds.), *Proceedings of the 18th International Conference on Distributed Computing Systems, ICDCS'98*, IEEE Computer Society Press, 1998, pp. 622-629.
4. Brazier, F.M.T., Dunin-Keplicz, B., Jennings, N.R. and Treur, J., Formal specification of Multi-Agent Systems: a real-world case. In: V. Lesser (Ed.), *Proceedings of the First International Conference on Multi-Agent Systems, ICMAS'95*, MIT Press, Cambridge, MA, 1995, pp. 25-32. Extended version in: *International Journal of Cooperative Information Systems*, M. Huhns, and M. Singh, (eds.), special issue on Formal Methods in Cooperative Information Systems: Multi-Agent Systems, vol. 6, 1997, pp. 67-94.
5. Brazier, F.M.T., Eck, P.A.T. van, and Treur, J., Modelling a society of simple agents: from conceptual specification to experimentation. In: R. Conte, R. Hegselmann, and P. Terna, (eds.), *Simulating Social Phenomena, Proceedings of the International Conference on Computer Simulation and Social Sciences,*

ICCS&SS'97, Lecture Notes in Economics and Mathematical Systems, vol. 456, Springer Verlag, 1997, pp. 103-107.

6. Brazier, F.M.T., Jonker, C.M., Jüngen, F.J., and Treur, J., Distributed Scheduling to Support a Call Centre: a Co-operative Multi-Agent Approach. In: *Proceedings of the Third International Conference on Practical Applications of Agents and Multi-Agent Systems, PAAM'98*, The Practical Application Company Ltd, 1998, pp. 555-576.

7. Brazier, F.M.T., Jonker, C.M., and Treur, J., Formalisation of a cooperation model based on joint intentions. In: J.P. Müller, M.J. Wooldridge, N.R. Jennings (eds.), *Intelligent Agents III (Proceedings of the Third International Workshop on Agent Theories, Architectures and Languages, ATAL'96)*, Lecture Notes in AI, volume 1193, Springer Verlag, 1997, pp. 141-155.

8. Brooks, R.A., A robust layered control system for a mobile robot. In: *IEEE Journal of Robotics and Automation*, vol. RA-2 (1), 1986, pp. 14-23.

9. Cornelissen, F., Jonker, C.M., and Treur, J., Compositional verification of knowledge-based systems: a case study in diagnostic reasoning. In: E. Plaza, R. Benjamins (eds.), *Knowledge Acquisition, Modelling and Management, Proceedings of the 10th European Workshop on Knowledge Acquisition, Modelling and Management, EKAW'97*, Lecture Notes in AI, vol. 1319, Springer Verlag, 1997, pp. 65-80.

10. Dennett, D.C., *The Intentional Stance.* MIT Press, Cambridge, 1987.

11. Dijkstra, E.W., *A discipline of programming.* Prentice Hall, 1976.

12. Gibson, J.J., The concept of the stimulus in psychology. In: *American Psychology* 15, 1960, pp. 694-703.

13. Hunter, W.S., The delayed reaction in animals. *Behavioral Monographs*, 2, 1912, pp. 1-85.

14. Jonker, C.M., and Treur, J., Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness. In: W.P. de Roever, A. Pnueli et al. (eds.), *Proceedings of the International Workshop on Compositionality, COMPOS'97*, Lecture Notes in Computer Science, vol. 1536, Springer Verlag, 1998.

15. Jonker, C.M., and Treur, J., A Generic Architecture for Broker Agents. In: *Proceedings of the Third International Conference on Practical Applications of Agents and Multi-Agent Systems, PAAM'98*, The Practical Application Company Ltd, 1998, pp. 623-624.

16. Kaelbling, L.P., An archictecture for intelligent reactive systems. In: Allen, J., and J. Hendler, and A. Tate (eds.), *Readings in Planning*, Morgan Kaufmann, 1990, pp. 713-728.

17. Maes, P. (ed.), *Designing Autonomous Agents: Theory and Practice from Biology to Engineering and Back*. MIT / Elsevier, 1990.

18. Morgan, C.L., *An introduction to comparative psychology*. London: Scott, 1894.

19. Müller, J. P., *The design of intelligent agents: a layered approach.* Lecture Notes in Artificial Intelligence, Vol. 1177, 1996.

20. Pavlov, I.P., *Conditioned reflexes*. London: Oxford, 1927.

21. Skinner, B.F., The generic nature of the concepts of stimulus and response. *Journal of Gen. Psychology* 12, 1935, pp. 40-65.

22. Tinklepaugh, O.L., Multiple delayed reaction with chimpanzees and monkeys. *Journal of Comparative Psychology*, 13, 1932, pp. 207-243.

23. Vauclair, J., *Animal Cognition*. Harvard Univerity Press, Cambridge, Massachusetts, 1996.

24. Watson, J.P., *Psychology from the standpoint of a behaviourist*. Philadelphia: Lippincott, 1919.

25. Wooldridge, M.J., and Jennings, N.R., Intelligent Agents: Theory and practice. In: *Knowledge Engineering Review*, 10(2), 1995, pp. 115-152.
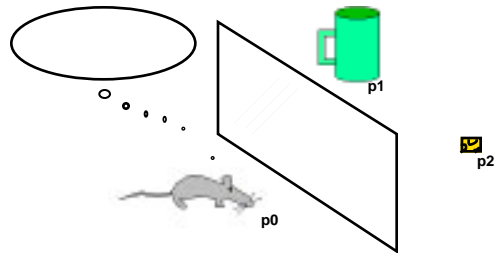
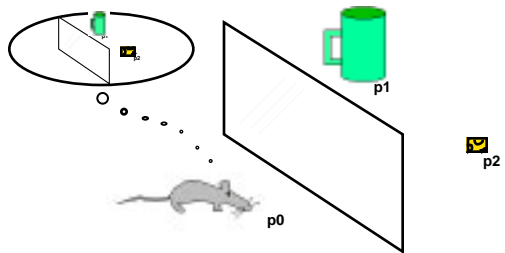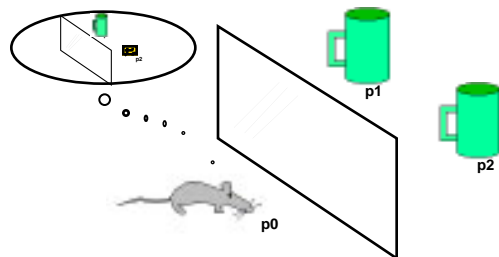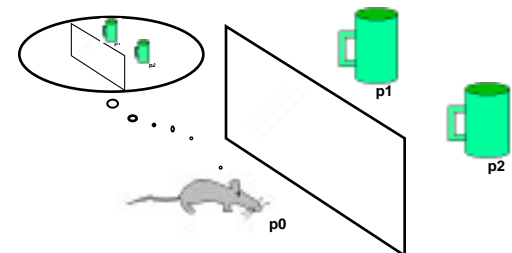|	world state	observation info of A	actions of A
0	screen at p0 cup1 at p1 food at p2 no cup at p2 self at p0		
1	screen at p0 cup1 at p1 food at p2 no cup at p2 self at p0	screen at p0 cup1 at p1 food at p2 no cup at p2 self at p0	
2	screen at p0 cup1 at p1 food at p2 cup2 at p2 self at p0	screen at p0 cup1 at p1 food at p2 no cup at p2 self at p0	
3	screen at p0 cup1 at p1 food at p2 cup2 at p2 self at p0	screen at p0 cup1 at p1 cup2 at p2 self at p0	

Trace A 3rd situation: 0
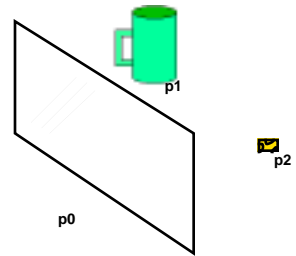
Trace A 3rd situation: 1

Trace A 3rd situation: 2

Trace A 3rd situation: 3

Trace A 3rd situation: 4

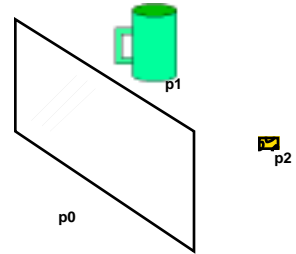|	world state	observation info of A	actions of A
4	no screen at p0 cup1 at p1 food at p2 cup2 at p2 self at p0	screen at p0 cup1 at p1 cup2 at p2 self at p0	
5	no screen at p0 cup1 at p1 food at p2 cup2 at p2 self at p0	no screen at p0 cup1 at p1 cup2 at p2 self at p0	

Trace A 3rd situation: 5
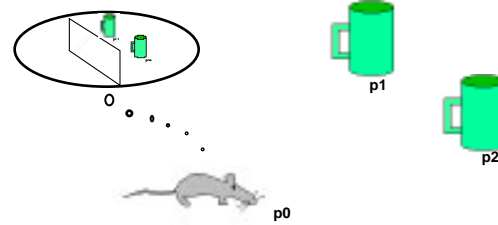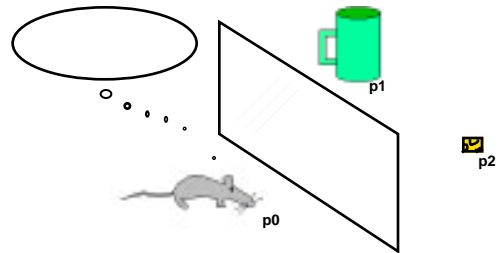

Trace A 3rd situation: 6

Trace B 3rd situation: 0
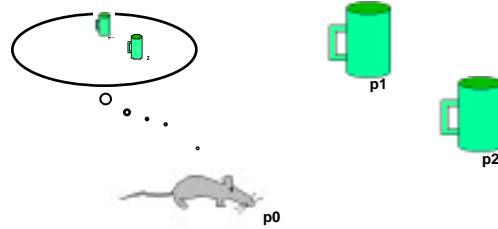
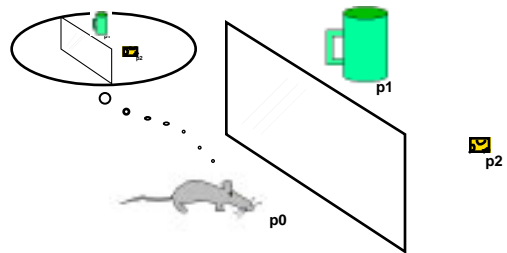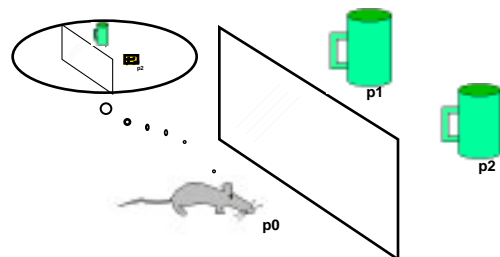|	world state	beliefs B	actions B
0	screen at p0 cup1 at p1 food at p2 no cup at p2		
1	screen at p0 cup1 at p1 food at p2 no cup at p2	screen at p0 cup1 at p1 food at p2 no cup at p2	
2	screen at p0 cup1 at p1 food at p2 cup2 at p2	screen at p0 cup1 at p1 food at p2 no cup at p2	
3	screen at p0 cup1 at p1 food at p2 cup2 at p2	screen at p0 cup1 at p1 food at p2 cup2 at p2	

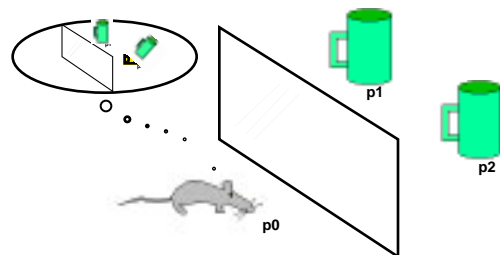Trace B 3rd situation: 1

Trace B 3rd situation: 2

Trace B 3rd situation: 3

Trace B 3rd situation: 4

	world state	beliefs B	actions B
4	no screen at p0 cup1 at p1 food at p2 cup2 at p2	screen at p0 cup1 at p1 food at p2 cup2 at p2	
5	no screen at p0 cup1 at p1 food at p2 cup2 at p2	no screen at p0 cup1 at p1 food at p2 cup2 at p2	
6	no screen at p0 cup1 at p1 food at p2 cup2 at p2	no screen at p0 cup1 at p1 food at p2 cup2 at p2	go to p2
7	no screen at p0 cup1 at p1 food at p2 cup2 at p2 self at p2	no screen at p0 cup1 at p1 food at p2 cup2 at p2	go to p2

Trace B 3rd situation: 5
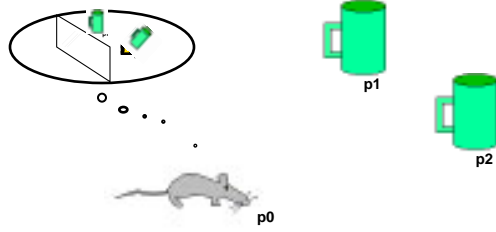
Trace B 3rd situation: 6
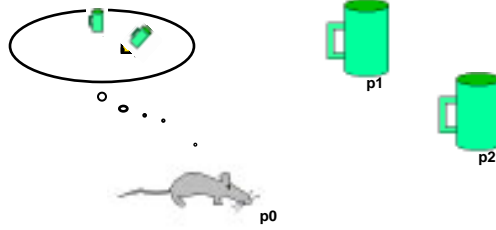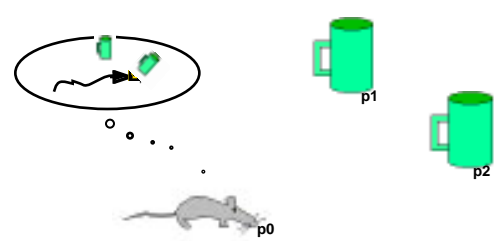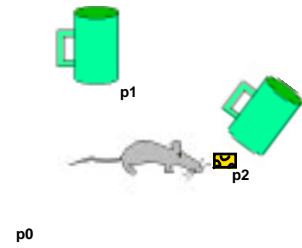

Trace B 3rd situation: 7

Trace B 3rd situation: 8

	world state	*beliefs C*	*goals C*	*actions and observations C*
0	screen at p0 cups at p1, p2 food at p2 self at p0			
1	no screen at p0 cups at p1, p2 food at p2 self at p0	no screen at p0 cups at p1, p2 self at p0	be fed find food	go to p1
2	no screen at p0 cups at p1, p2 food at p2 self at p1	no screen at p0 cups at p1, p2 self at p1	be fed find food	to be observed food at p1
3	no screen at p0 cups at p1, p2 food at p2 self at p1	no screen at p0 cups at p1, p2 self at p1 no food at p1	be fed find food	go to p2
4	no screen at p0 cups at p1, p2 food at p2 self at p2	no screen at p0 cups at p1, p2 self at p2 no food at p1	be fed find food	to be observed food at p2
5	no screen at p0 cups at p1, p2 food at p2 self at p2	no screen at p0 cups at p1, p2 self at p2 no food at p1 food at p2	be fed get food inside	eat
6	no screen at p0 cups at p1, p2 no food at p2 self at p2	no screen at p0 cups at p1, p2 self at p2 no food at p1 no food at p2	be fed find food	

APPENDIX D   BEHAVIOUR OF AGENT MODEL D IN SITUATION 4

	world state	beliefs D	goals D	actions and observations D
0	screen at p0 cups at p1, p2 food at p2 higher competitor at p2 self at p0			
1	no screen at p0 cups at p1, p2 food at p2 higher competitor at p2 self at p0	no screen at p0 cups at p1, p2 self at p0 higher competitor at p2	be fed find food	go to p1
2	no screen at p0 cups at p1, p2 food at p2 higher competitor at p2 self at p1	no screen at p0 cups at p1, p2 self at p1 higher competitor at p2	be fed find food	to be observed food at p1
3	no screen at p0 cups at p1, p2 food at p2 higher competitor at p2 self at p1	no screen at p0 cups at p1, p2 self at p1 no food at p1 higher competitor at p2	be fed find food	
4	no screen at p0 cups at p1, p2 food at p2 no higher competitor at p2 self at p2	no screen at p0 cups at p1, p2 self at p1 no food at p1 no higher competitor at p2	be fed find food	go to p2
5	no screen at p0 cups at p1, p2 food at p2 no higher competitor at p2 self at p2	no screen at p0 cups at p1, p2 self at p2 no food at p1 no higher competitor at p2	be fed find food	to be observed food at p2
6	no screen at p0 cups at p1, p2 food at p2 no higher competitor at p2 self at p2	no screen at p0 cups at p1, p2 self at p2 no food at p1 food at p2 no higher competitor at p2	be fed get food inside	eat
7	no screen at p0 cups at p1, p2 no food at p2 no higher competitor at p2 self at p2	no screen at p0 cups at p1, p2 self at p2 no food at p1 no food at p2 no higher competitor at p2	be fed find food	

	world state	beliefs E	experience E	goals E	actions and observations E
0	cups at p1, p2, p3 green at p1 blue at p2 red at p3 food at p2 self at p0		food relevance: red-green-blue	be fed find food	

1	cups at p1, p2, p3
	green at p1
	blue at p2
	red at p3
	food at p2
	self at p0
		cups at p1, p2, p3
		green at p1
		blue at p2
		red at p3
		self at p0
			food relevance:
			    red-green-blue
				be fed
				find food
					relevancy of actions:
					    go to p3
					    go to p1
					    go to p2
					selected:
					    go to p1

2	cups at p1, p2, p3
	green at p1
	blue at p2
	red at p3
	food at p2
	self at p1
		cups at p1, p2, p3
		green at p1
		blue at p2
		red at p3
		self at p1
			food relevance:
			    red-green-blue
				be fed
				find food
					to be observed
					    food at p1

3	cups at p1, p2, p3
	green at p1
	blue at p2
	red at p3
	food at p2
	self at p1
		cups at p1, p2, p3
		green at p1
		blue at p2
		red at p3
		self at p1
		no food at p1
			food relevance:
			    red-blue-green
				be fed
				find food
					relevancy of actions:
					    go to p3
					    go to p2
					selected:
					    go to p2

4	cups at p1, p2, p3
	green at p1
	blue at p2
	red at p3
	food at p2
	self at p2
		cups at p1, p2, p3
		green at p1
		blue at p2
		red at p3
		self at p2
		no food at p3
			food relevance:
			    red-blue-green
				be fed
				find food
					to be observed
					    food at p2

5	cups at p1, p2, p3
	green at p1
	blue at p2
	red at p3
	food at p2
	self at p2
		cups at p1, p2, p3
		green at p1
		blue at p2
		red at p3
		self at p2
		no food at p1
		food at p2
			food relevance:
			    blue-red-green
				be fed
				get food inside
					relevancy of actions:
					    eat
					selected:
					    eat

6	cups at p1, p2, p3
	green at p1
	blue at p2
	red at p3
	no food at p2
	self at p2
		cups at p1, p2, p3
		green at p1
		blue at p2
		red at p3
		self at p2
		no food at p1, p2
			food relevance:
			    blue-red-green
				be fed
				find food
					relevancy of actions:
					    go to p3
					selected:
					    go to p3