Compositional Verification of Multi-Agent Systems: a Formal Analysis of Pro-activeness and Reactiveness

Catholijn M. Jonker, Jan Treur

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

Abstract
A compositional method is presented for the verification of multi-agent systems. The advantages of the method are the well-structuredness of the proofs and the reusability of parts of these proofs in relation to reuse of components. The method is illustrated for an example multi-agent system, consisting of co-operative information gathering agents. This application of the verification method results in a formal analysis of pro-activeness and reactiveness of agents.

1 Introduction

When designing multi-agent systems, it is often hard to guarantee that the specification of a system that has been designed actually fulfils the needs, i.e., whether it satisfies the design requirements. Especially for critical applications, for example in real-time domains, there is a need to prove that the designed system will have certain properties under certain conditions (assumptions). While developing a proof of such properties, the assumptions that define the bounds within which the system will function properly are generated. For nontrivial examples, verification can be a very complex process, both in the conceptual and computational sense. For these reasons, it is a recent trend in the literature on verification in general to study the use of compositionality and abstraction to structure the process of verification; for example, see (Abadi and Lamport, 1993; Hooman, 1994; Dams, Gerth and Kelb, 1996).

The development of structured modelling frameworks and principled design methods tuned to the specific area of multi-agent systems is currently underway; e.g., (Brazier, Dunin-Keplicz, Jennings and Treur, 1995; Fisher and Wooldridge, 1997; Kinny, Georgeff and Rao, 1996). As part of any mature multi-agent system design method, a verification approach is required. For example, in (Fisher and Wooldridge, 1997) verification is addressed within a temporal belief logic. This verification method does not exploit compositionality within the agents. In the current paper, in Section 3, a compositional verification method for multi-agent systems is introduced. Roughly spoken, the requirements of the whole system are formally verified by deriving them from assumptions that themselves are properties of agents, which in their turn may be derived from assumptions on sub-components of agents, and so on.

The compositional verification method introduced here is illustrated for an example multi-agent system, consisting of two cooperative information gathering agents and the world. For this example multi-agent system, requirements are formulated (both the required *static* and*dynamic* properties), including variants of*pro-activeness* and *reactiveness*. These requirements are formalised in terms of temporal semantics. It is shown how they can be derived from properties of agents and how these agent properties in turn can be derived from properties of the agent components. A compositional system specification is introduced in Section 4. The system specification defines how the system is composed of the two agents and the world and how each agent is composed of four agent components: for own process control, world interaction management, agent interaction management, and an agent specific task (which in this case is classification of objects in the world). The compositional specification itself is expressed in the modelling framework DESIRE, shortly introduced in Section 2. The application of the compositional verification method to the example multi-agent system is presented in Section 5 for the top level of the composition. More details on the lower levels can be found in Sections 6 and 7.

2 Compositional Modelling of Multi-Agent Systems

The example task model described in this paper is specified within the compositional modelling framework DESIRE for multi-agent systems (framework for DEsign and Specification of Interacting REasoning components; cf. (Langevelde, Philipsen and Treur, 1992; Brazier, Dunin-Keplicz, Jennings, Treur, 1995)). In DESIRE, a design consist 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.

2.1 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 lower level processes.

2.1.1 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.

Specification of a Process

The identified processes are modelled as *components*. For each process the *types of information* required as input and resulting as output are identified as well. This is modelled as *input and output interfaces* of the components.

Specification of Process Abstraction Levels

The identified levels of process abstraction are modelled as *abstraction/specialisation relations* between components at adjacent levels of abstraction: components may be

composed of other components or they may be *primitive*. Primitive components may be either reasoning components (for example based on a knowledge base), or, alternatively, components capable of performing tasks such as calculation, information retrieval, optimisation, et cetera.

The identification of processes at different abstraction levels results in specification of components that can be used as building blocks, and of a specification of the sub-component relation, defining which components are a sub-component of a which other component. The distinction of different process abstraction levels results in process hiding.

2.1.2 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 the possibilities for *information exchange* between processes (*static view* on the composition), and *task control knowledge* used to control processes and information exchange (*dynamic view* on the composition).

Information Exchange

Knowledge of information exchange defines which types of information can be transferred between components and the *information links* by which this can be achieved. Two types of information links are distinguished: *private* information links and *mediating* information links. For a given parent component, a *private information link* relates output of one of its components to input of another, by specifying which truth value of a specific output atom is linked with which truth value of a specific input atom. Atoms can be renamed: each component can be specified in its own language, independent of other components. In a similar manner *mediating information links* transfer information from the input interface of the parent component to the input interface of one of its components, or from the output interface of one of its components to the output interface of the parent component itself. Mediating links specify the relation between the information at two adjacent abstraction levels in the process composition.

Task Control Knowledge

Components may be activated sequentially or they may be continually capable of processing new input as soon as it arrives (*awake*). The same holds for information links: information links may be explicitly activated or they may be awake. *Task control knowledge* specifies under which conditions which components and information links are active (or made awake). Evaluation criteria, expressed in terms of the evaluation of the results (success or failure), provide a means to guide further processing.

2.2 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.

2.2.1 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. The resulting levels of knowledge abstraction can be distinguished for both information types and knowledge bases.

Information Types

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 as signatures in *order-sorted predicate logic*.

Knowledge Bases

A knowledge base defines a part of the knowledge that is used in one or more of the processes. Knowledge is represented logically by rules in order-sorted predicate logic.

Knowledge bases use ontologies defined in information types. Which information types are used in a knowledge base defines a relation between information types and knowledge bases.

2.2.2 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 that are distinguished, and results in information and knowledge hiding.

2.3 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.

The semantics of the modelling language are based on temporal logic (cf., Brazier, Treur, Wijngaards and Willems, 1996). Design is supported by graphical tools within the DESIRE software environment. Translation into an operational system is straightforward; the software environment includes implementation generators with which specifications can be translated into executable code. DESIRE has been successfully applied to design both single agent and multi-agent systems.

3 Compositional Verification

The purpose of verification is to prove that, under a certain set of assumptions, a system will adhere to a certain set of properties, for example the design requirements. In our approach, this is done by a mathematical proof (i.e., a proof in the form mathematicians are accustomed to do) that the specification of the system together with the assumptions implies the properties that it needs to fulfil. In this sense verification leads to a formal analysis of relations between properties and assumptions.

3.1 The Compositional Verification Method

A compositional multi-agent system can be viewed at different levels of abstraction. Viewed from the top level, denoted by L_0, the complete system is one component S, with interfaces, whereas internal information and processes are hidden (information and process hiding). At the next lower level of abstraction, the system component S can be viewed as a composition of agents and the world, information links between them, and task control. Each agent A is composed of its sub-components, and so on. The compositional verification method takes this compositional structure into account.

For composed components two types of properties are recognised: behavioural and environmental properties. A behavioural property is a property on the output of the component. Behavioural properties can be conditional, or unconditional. A behavioural property is conditional if the statements about the output of the component hold under the assumption that some specific conditions hold for its input. For example, a conditional behavioural property of a diagnostic agent could be *conditional conclusion correctness of an agent* (i.e., if the observation information needed for diagnosis, which is input of the agent, is correct, then all diagnostic output of the agent is correct), whereas the corresponding unconditional property would be *conclusion correctness of an agent* (i.e., all diagnostic output of the agent is correct). An environmental property is a property on the input of the component (possibly referring to certain conditions on the output).

The primitive components can be verified using more traditional verification methods such as described in (Treur and Willems, 1994; Leemans, Treur and Willems, 1995). Verification of a composed component is done using properties of the sub-components it embeds and the task control knowledge, and environmental properties of the component (depending on the rest of the system, including the world). This introduces a form of compositionality in the verification process: given a set of environmental properties the proof that a certain component adheres to a set of behavioural properties depends on the (assumed) properties of its sub-components, properties of the interactions between those sub-components, and the manner in which they are controlled. The assumptions under which the component functions properly, are the properties to be proven for its sub-components. This implies that properties at different levels of abstraction are involved in the verification process.

Often these properties are not given at the start of the verification process. Actually, the process of verification has two main aims:

- to find the properties
- given the properties, to prove the properties

The verification proofs that connect one abstraction level with the other are compositional in the following manner: any proof relating level i to level i+1 can be combined with any proof relating level i-1 to level i, as long as the same properties at level i are involved. This means, for example, that the whole compositional structure beneath level i can be replaced by a completely different design as long as the same properties at level i are achieved. After such a modification the proof from level i to level i+1 can be reused; only the proof from level i-1 to level i has to be adapted. In this sense the verification method supports reuse of verification proofs.

The compositional verification method can be formulated in more detail as follows:

A. Verifying one Abstraction Level Against the Other

For each abstraction level the following procedure for verification is followed:

1. Determine which properties are of interest (for the higher level).
2. Determine which assumptions (at the lower level) are needed to guarantee these properties, and which environment properties.
3. Prove the properties on the basis of these assumptions, and the environment properties.

B. Verifying a Primitive Component

For primitive knowledge-based components a number of techniques exist in literature, see for example (Treur, Willems 1994; Leemans, Treur, Willems 1995). For primitive non-knowledge-based components, such as databases, or neural networks, or optimisation algorithms, verification techniques can be used that are especially tuned for that type of component.

C. The Overall Verification Process

To verify the complete system

1. Determine the properties that are desired for the whole system.
2. Apply the above procedure **A** iteratively until primitive components are reached. In the iteration the desired properties of abstraction level L_i are either:
   - those determined in step **A1**, if i = 0, or
   - the assumptions made for the higher level L_{i-1}, if i > 0
3. Verify the primitive components according to **B**.

The results of verification are:

- Properties and assumptions at the different abstraction levels.
- The logical relations between the properties of different abstraction levels.

Notes:

- both static and dynamic properties and connections between them are covered.
- reuse of verification results is supported (refining an existed verified compositional model by further decomposition, leads to a verification of the refined system in which the verification structure of the original system can be reused).
- process and information hiding limits the complexity of the verification per abstraction level.
- a requirement to apply the compositional verification method described above is the availability of an explicit specification of how the system description at an abstraction level L_i is composed from the descriptions at the lower abstraction level L_{i+1}; the compositional modelling framework DESIRE is an instance of a modelling framework that fulfils this requirement.
- in principle alternative (e.g., bottom-up or mixed) procedures can be formulated as well.

3.2 Semantics Behind the Compositional Verification Method

In principle, verification is always relative to semantics of the system descriptions that are verified. For the compositional verification method, these semantics are based on compositional information states which evolve over time. In this subsection a brief overview of these assumed semantics is given.

An *information state* M of a component D is an assignment of truth values {true, false, unknown} to the set of ground atoms that play a role within D. The compositional structure of D is reflected in the structure of the information state. A formal definition can be found in (Brazier, Treur, Wijngaards and Willems, 1996; Brazier, Eck and Treur, 1996). The set of all possible information states of D is denoted by IS(D).

A *trace* $\mathcal{M}$ of a component D is a sequence of information states $(M^t)_{t \in \mathbf{N}}$ in IS(D). The set of all traces is denoted by $IS(D)^{\mathbf{N}}$, or Traces(D). Given a trace $\mathcal{M}$ of component D, the information state of the input interface of component C at time point t of the component D is denoted by $state_D(\mathcal{M}, t, input(C))$, where C is either D or a sub-component of D. Analogously, $state_D(\mathcal{M}, t, output(C))$, denotes the information state of the output interface of component C at time point t of the component D. Given a trace $\mathcal{M}$ of component D, the task control information state of component C at time point t of the component D is denoted by $state_C(\mathcal{M}, t, tc(C))$, where C is either D or a sub-component of D.

To connect neighbouring levels of abstraction in a verification proof for a DESIRE specification, the following elements can be used:

- the assumptions of the sub-components specified within component D
- the interactions between the sub-components of D and/or the interfaces of D
- the input / output information states of the sub-components of D
- the task control information states of the sub-components of D
- the information states of component D
- the task control information states of component D

4 The Example Multi-Agent Model

The example multi-agent model is composed of three components: two agents A and B and a component W representing the external world, see Figure 1. Each of the agents is able to acquire partial information about the external world (by observation). Each agent's own observations are insufficient to draw conclusions of a desired type, but the combined information of both agents is sufficient. Therefore communication is required to be able to draw conclusions. The agents can communicate their own observation results and requests for observation information of the other agent. This by itself not unrealistic situation is simplified to the following materialised form. The world situation consists of an object that has to be classified. One agent can only observe the bottom view of the object, the other agent the side view. By exchanging and combining observation information they are able to classify the object.

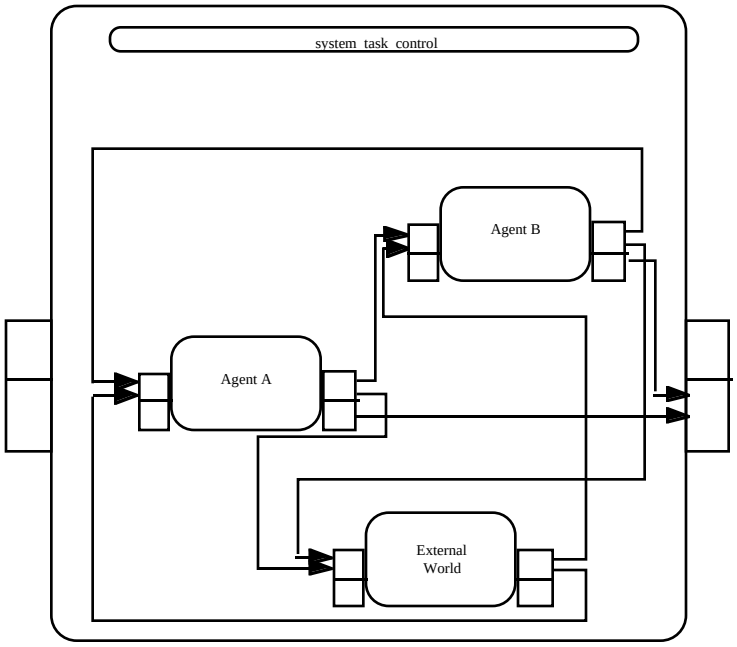

Fig. 1. The example Multi-Agent System

Communication from the agent A to B takes place in the following manner:

- the agent A generates at its output interface a statement of the form:
  to_be_communicated_to(<*type*>, <*atom*>, <*sign*>, B)
- the information is transferred to B; thereby it translated into
  communicated_by(<*type*>, <*atom*>, <*sign*>, A)

In the example <*type*> can be filled with a label request or world_info, <*atom*> is an atom expressing information on the world, and <*sign*>, is one of pos or neg, to indicate truth or falsity.

Interaction between an agent A and the world takes place as follows:

- the agent A generates at its output interface a statement of the form:
  to_be_observed(<*atom*>)
- the information is transferred to W; thereby it is translated into
  to_be_observed_by(<*atom*>, A)
- the world W generates at its output interface a statement of the form:
  observation_result_for(<*atom*>, <*sign*>, A)
- the information is transferred to A; thereby it is translated into
  observation_result(<*atom*>, <*sign*>)

Part of the output of an agent are conclusions about the classification of the object of the form object_type(s); these are transferred to the output of the system.

To be able to perform its tasks, each agent is composed of four components, see Figure 2: three for generic agent tasks (world interaction management, agent interaction management, own proces control), and one for an agent specific task (object classification).

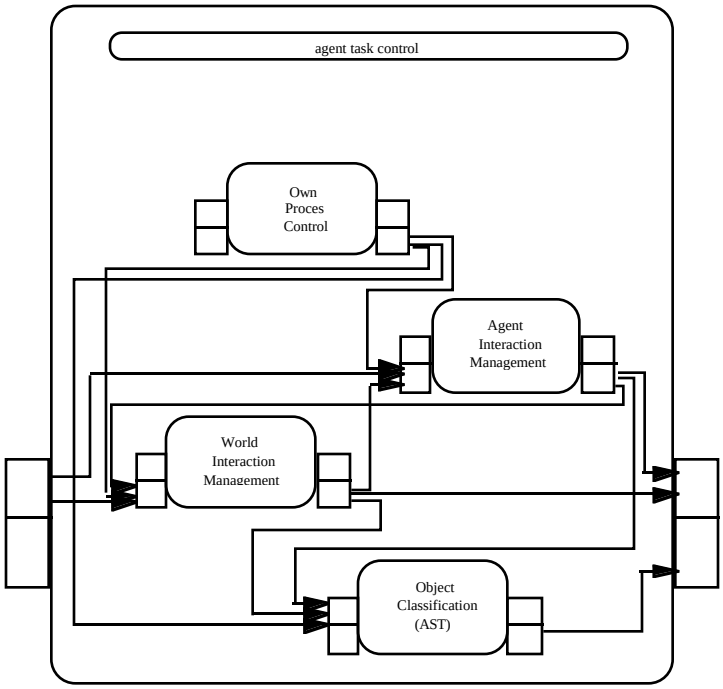

Fig. 2. Composition of an agent

object classification (agent specific task)

This component is able to draw a conclusion if it has input on the two views on the object. The component can reason on the basis of the world knowledge represented in the table depicted in Table 1:

	○	△	□
○	sphere	cone	cylinder
△	cone	tetrahedron	pyramid
□	cylinder	pyramid	cube

Table 1. World knowledge

world interaction management

This component reasons about the manner in which the agent interacts with the world. Here it is decided under which conditions which observations are to be performed in the world.

agent interaction management

This component reasons about the agent's communication with other agents. In this component it is determined when to request which information from the other agent. Another task is to determine when to provide which observation information to the other agent and what to do with the world information received from the other agent.

own process control

This component defines the agent's own characteristics or attitudes. Information on these attitudes can be transferred to other components, to influence the reasoning that takes place there. The agents can differ in their attitudes towards observation and communication: an agent may or may not be *pro-active*, in the sense that it takes the initiative with respect to one or more of:

- performing observations
- communicate its own observation results to the other agent
- ask the other agent for its observation results
- draw conclusions about the classification of the object

Moreover, it may be *reactive* to the other agent in the sense that it responds to a request for observation information:

- by communicating its observation result as soon as they are available
- by starting to observe for the other agent

These agent attitudes are represented explicitly as (meta-)facts in the agent's component own process control. By varying these attitude facts, different variants of agents can be defined. The impact of these explicitly specified characteristics has been specified in the model. For example, if an agent has the attitude that it will always take the initiative to communicate its observation results as soon as they are acquired, then the agent's behaviour should show this, but if the characteristic is not there, then this behaviour should not be present. This requires an adequate interplay between the component own process control and the component agent interaction management within the agent, and adequate knowledge within agent interaction management.

The successfulness of the system depends on the attitudes of the agents. For example, if both agents are pro-active and reactive in all respects, then they can easily come to a conclusion. However, it is also possible that one of the agents is only reactive, and still the other agent comes to a conclusion. Or, an agent that is only pro-active in reasoning and reactive in information acquisition may come to a conclusion due to pro-activeness of the other agent. So, successfulness can be achieved in many ways and depends on subtle interactions between pro-activeness and reactiveness attitudes of both agents. The formal analysis of the example in the following sections provides a detailed picture of these possibilities.

5 Formal Analysis of the Example System: Top Level

In this section the properties of the system as a whole (defined in Section 5.1) are related to properties of the agents and the world (defined in Section 5.2 and 5.3), and their interaction.

5.1 Properties for the Top Level of the System

First, it is determined which properties the system as a whole should satisfy. Considering that the system S is a classification system, it is expected that S produces output of the form object_type(s) for some s. A first requirement is that output generated by the system is correct, i.e., if the system derives object_type(s) for some s, it is true in the world situation. Let the world state (which is assumed static) be denoted by M. In Figure 4 the successfulness property of S is related to other properties of S and assumed properties of the next level. In Figure 3 the correctness property of S is similarly related to other properties. The following property relates the output of the system to the current world state.

Correctness of s

The system S is called *correct* if:

$$\forall \mathcal{M} \in \text{Traces}(S)\ \forall t\ \forall s$$
$$[\ \text{state}_S(\mathcal{M}, t, \text{output}(S)) \models \text{object\_type}(s) \Rightarrow M \models \text{object\_type}(s)\ ]$$
$$\wedge\ [\ \text{state}_S(\mathcal{M}, t, \text{output}(S)) \models \neg\text{object\_type}(s) \Rightarrow M \models \neg\text{object\_type}(s)\ ]$$

Output information of s is only provided by agents

As can be seen in Figure 1 the only information links connected to the output of S are the information link from agent A to S and the information link from agent B to S. Furthermore, the output interface of S cannot spontaneously change its contents. Therefore, the output information of the system is only provided by agents.

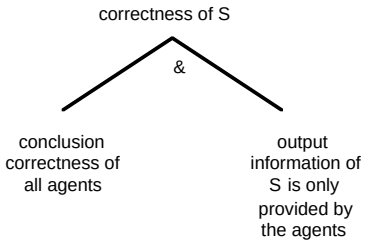

Fig. 3. Correctness of s

Next, the system is required to be successful in generating conclusions: during the process, for each s, at some time point it should either have derived positive output, or negative output:

Successfulness of s

The system S is called *successful* if:

$$\forall \mathcal{M} \in \text{Traces}(S)\ \forall s\ \exists t\ \text{state}_S(\mathcal{M}, t, \text{output}(S)) \models \text{object\_type}(s))$$
$$\vee\ \exists t\ \text{state}_S(\mathcal{M}, t, \text{output}(S)) \models \neg\text{object\_type}(s))$$

To guarantee the adequate information exchange between the different components, the following properties are needed:

Interaction effectiveness
The interaction from agent x to the output interface of system s is called *effective* if, for ϕ either object_type(s) or $\neg$object_type(s):

$\forall \mathcal{M} \in$ Traces(S) $\forall t \forall s$ [state$_S(\mathcal{M}, t,$ output(X)) $\models \phi$
$\Rightarrow \exists t'>t$ state$_S(\mathcal{M}, t',$ output(S)) $\models \phi$]

The interaction from the external world w to agent x is called *effective* if:

$\forall \mathcal{M} \in$ Traces(S) $\forall t \forall r,$sign
[state$_S(\mathcal{M}, t,$ output(W)) $\models$ observation_result_for(view(X,r), sign, X)
$\Rightarrow \exists t'>t$ state$_S(\mathcal{M}, t',$ input(X)) $\models$ observation_result(view(X,r),sign)]

The interaction from the agent x to the external world w is called *effective* if:

$\forall \mathcal{M} \in$ Traces(S) $\forall t \forall r,$sign
[state$_S(\mathcal{M}, t,$ output(X)) $\models$ to_be_observed (view(X,r))
$\Rightarrow \exists t'>t$ state$_S(\mathcal{M}, t',$ input(W)) $\models$ to_be_observed_by (view(X,r), X)]

Interaction effectiveness can be proven from the detailed specification of the information links involved and the timely functioning of those information links as specified in the task control of the component containing the information link given in the detailed specification. This property is needed to prove several environment properties of the agents and also to prove successfulness of s. Sometimes it will not be stated explicitly.

Agent provides output information of s
An agent x provides output information of system s if x is conclusion successful and the interaction from x to s is effective.

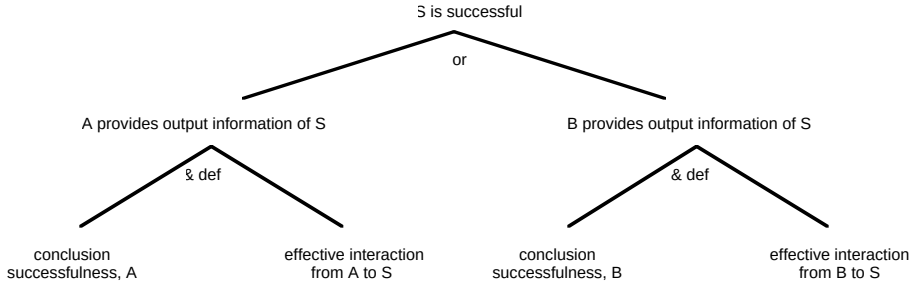

Fig. 4. Successfulness of s

It is undesirable (for a static world situation) that the system changes its mind during the process. Therefore the requirement is chosen that once a conclusion has been derived, this is never revised:

Conservativity of s

The system S is called *conservative* if:

$\forall \mathcal{M} \in$ Traces(S) $\forall t \forall s$
[$state_S(\mathcal{M}, t, output(S)) \models$ object_type(s) $\Rightarrow$
$\forall t'>t$ $state_S(\mathcal{M}, t', output(S)) \models$ object_type(s)]
$\wedge$ [$state_S(\mathcal{M}, t, output(S)) \models \neg$ object_type(s) $\Rightarrow$
$\forall t'>t$ $state_S(\mathcal{M}, t', output(S)) \models \neg$ object_type(s)])

The property of conservativity of S can be proven in a similar way as the other properties of S. The proof has been omitted from this paper.

Communication effectiveness

The communication from agent X to agent Y is called *effective* if:

$\forall \mathcal{M} \in$ Traces(S) $\forall t \forall sign \forall \phi$
[$state_S(\mathcal{M}, t, output(X)) \models$ to_be_communicated_to(q, ϕ, sign, Y)
$\Rightarrow \exists t'>t$ $state_S(\mathcal{M}, t', input(Y)) \models$ communicated_from(q, ϕ, sign, X)]

Communication effectiveness can be proven in the same way as interaction effectiveness. This property is needed to prove environment properties of the agents.

5.2 Properties of the Agents

The required properties of the system have been proven from assumed properties of the components at one level lower. During this proof process these assumptions have been discovered. A number of assumptions are quite straightforward. For example, correctness inherits upward from the agents to the system:

Conclusion correctness of an agent

An agent X is called *conclusion correct* if:

$\forall \mathcal{M} \in$ Traces(X) $\forall t \forall s$
[$state_X(\mathcal{M}, t, output(X)) \models$ object_type(s) $\Rightarrow$ M $\models$ object_type(s)]
$\wedge$ [$state_X(\mathcal{M}, t, output(X)) \models \neg$object_type(s) $\Rightarrow$ M $\models \neg$object_type(s)]

This property logically depends on other properties of the agent, input correctness and conditional conclusion correctness, as can be seen for agent A in Figure 5.

Input correctness of an agent

a) An agent X is called *observation input correct* if

$\forall \mathcal{M} \in$ Traces(X) $\forall t \forall r$
[$state_X(\mathcal{M}, t, input(X)) \models$ observation_result(view(X,r),pos) $\Rightarrow$ M $\models$ view(X,r)]
$\wedge \forall \mathcal{M} \in$ Traces(X) $\forall t \forall r$
[$state_X(\mathcal{M}, t, input(X)) \models$ observation_result(view(X,r),neg) $\Rightarrow$ M $\models \neg$view(X,r)]

b) An agent X is called *communication input correct* if

$\forall \mathcal{M} \in$ Traces(X) $\forall t \forall r$
[$state_X(\mathcal{M}, t, input(X)) \models$ communicated_from(world_info,view(Y,r), pos,Y) $\Rightarrow$ M $\models$ view(Y,r)]
$\wedge \forall \mathcal{M} \in$ Traces(X) $\forall t \forall r$
[$state_X(\mathcal{M}, t, input(X)) \models$ communicated_from(world_info,view(Y,r), neg,Y) $\Rightarrow$ M $\models \neg$view(Y,r)]

c) An agent X is called *input correct* if X is observation input correct and communication input correct.

Conditional conclusion correctness of an agent
An agent X is called *conditionally conclusion correct* if the following holds: if input correctness of X then conclusion correctness of X.

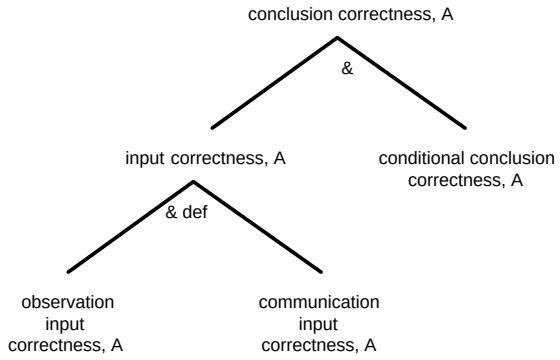

Fig. 5. Conclusion correctness of A

The following property is needed to prove conservativity of S.

Conclusion conservativity of an agent
The agent X is called *conclusion conservative* if:

$$\forall \mathcal{M} \in \text{Traces}(X)\ \forall t\ \forall s$$
$$[\ \text{state}_X(\mathcal{M}, t, \text{output}(X)) \models \text{object\_type}(s) \Rightarrow$$
$$\forall t'>t\ \ \text{state}_X(\mathcal{M}, t', \text{output}(X)) \models \text{object\_type}(s)\ ]$$
$$\wedge\ [\ \text{state}_X(\mathcal{M}, t, \text{output}(X)) \models \neg\, \text{object\_type}(s) \Rightarrow$$
$$\forall t'>t\ \ \text{state}_X(\mathcal{M}, t', \text{output}(X)) \models \neg\, \text{object\_type}(s)\ ]\ )$$

Again, the proof has been omitted from this paper. Successfulness of the system (see Figure 4) depends on successfulness of at least one of the agents.

Conclusion successfulness of an agent
The agent X is called *conclusion successful* if:

$$\forall \mathcal{M} \in \text{Traces}(X)\ \exists t\ \ \text{state}_X(\mathcal{M}, t, \text{output}(X)) \models \text{object\_type}(s))$$
$$\vee\ \exists t\ \ \text{state}_X(\mathcal{M}, t, \text{output}(X)) \models \neg\text{object\_type}(s))$$

This property can be proven in a number of ways. However, in all proofs the properties information saturation of the agent and conclusion pro-activeness is required, see Figure 6 for the logical relations between properties of agent A that are needed to prove conclusion successfulness of agent A.

Information saturation of an agent

a) The agent X is called *observation info saturating* if:

$\forall \mathcal{M} \in$ Traces(X) $\exists$ t $\forall$ r $\exists$ sign

state$_X(\mathcal{M}$, t, input(X)) $\models$ observation_result(view(X,r), sign)

b) The agent X is called *communicated info saturating* if:

$\forall \mathcal{M} \in$ Traces(X) $\exists$ t $\forall$ r $\exists$ sign

state$_X(\mathcal{M}$, t, input(X)) $\models$ communicated_from(world_info, view(Y,r), sign, Y)

c) The agent X is called *request saturating* if for all agents Y different from X:

$\forall \mathcal{M} \in$ Traces(X) $\exists$ t $\forall$ r

state$_X(\mathcal{M}$, t, input(X)) $\models$ communicated_from(request, view(X,r), pos, Y)

d) The agent X is called *information saturating* if X is observation info saturating and communicated info saturating.

Conclusion pro-activeness

The agent X is called *conclusion pro-active* if for all agents Y different from X:

$\forall \mathcal{M} \in$ Traces(X) $\forall$t,t'

[$\forall$r,r' $\exists$ sign,sign'

state$_X(\mathcal{M}$, t, input(X)) $\models$ observation_result(view(X,r), sign) $\wedge$

state$_X(\mathcal{M}$, t', input(X)) $\models$ communicated_from(world_info, view(Y,r'), sign', Y)]]

$\Rightarrow$ $\exists$ t''>t, t''>t' $\forall$s [state$_X(\mathcal{M}$, t'', output(X)) $\models$ object_type(s) $\vee$

state$_X(\mathcal{M}$, t'', output(X)) $\models$ $\neg$object_type(s)]

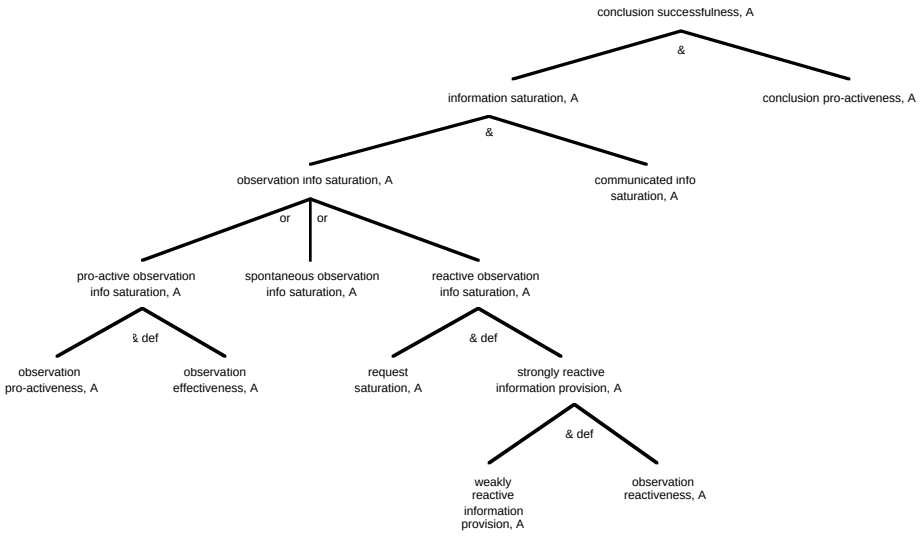

Fig. 6. Conclusion successfulness of A

All properties occurring in Figure 6 are also properties of agent A. The leaves in the tree are either environmental properties of A or behavioural properties of A. To prove environmental properties of a component behavioural properties of other components of the same level (in this case other agents and/or the world) are needed as are properties about interactions between these components (in this case interactions

between agents and between the world and agents). For example, the property communicated information saturation of agent A (see Figure 6) can be proved from interaction effectiveness from agent B to agent A and the property successful information provision of agent B, see Figure 7.

Information provision successfulness

a) The agent X is called *successful information providing* if:

$\forall \mathcal{M} \in$ Traces(X) $\forall$ r, $\exists$ sign $\exists$ t

$\text{state}_X(\mathcal{M}, t, \text{output(X)}) \models$ to_be_communicated_to(world_info, view(X,r), sign, Y)

b) The agent X is called *successful information providing pro-active* if
X is observation effective and strongly information providing pro-active.

c) The agent X is called *successful information providing reactive* if
X is request saturating and reactive observation effective.

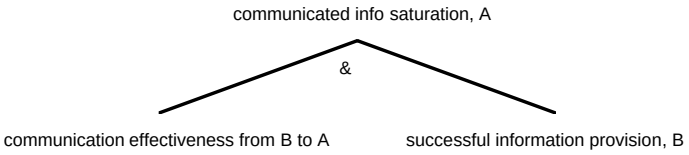

Fig. 7. Communicated info saturation of A

Similarly, the properties observation input correctness and communication input correctness of an agent depend on correct information coming from the other agent or from the world (see Section 5.3 for definitions of properties concerning the world), see Figure 8.

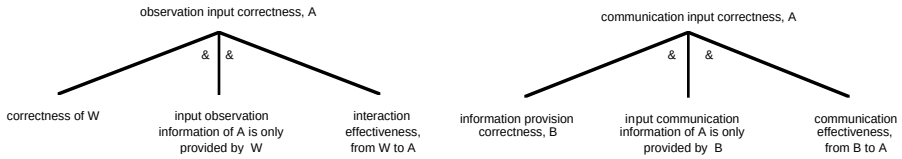

Fig. 8. Information correctness of A

The property **spontaneous observation info saturation of an agent** as used in Figure 6 is defined by the property pro-activeness of the world (see Section 5.3) and interaction effectiveness from the world to that agent, see Figure 9.

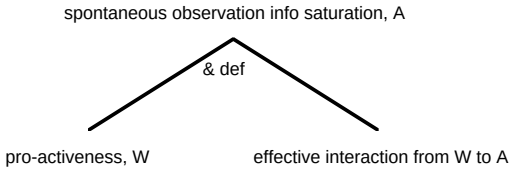

Fig. 9. Spontaneous observation info saturation of A

The property successful information provision of agent B, used in Figure 7, can be proven in several ways. The first division made in Figure 10 is between pro-active and reactive information provision. Also the notions strong and weak are used.

Information provision correctness

The agent X is called *information providing correct* if:

$\forall \mathcal{M} \in$ Traces(X) $\forall r \; \forall t$
[$state_X(\mathcal{M}, t, output(X)) \models$ to_be_communicated_to(world_info, view(X,r), pos, Y)
$\Rightarrow$ M $\models$ view(X,r)]
$\wedge \; \forall \mathcal{M} \in$ Traces(X) $\forall r \; \forall t$
[$state_X(\mathcal{M}, t, output(X)) \models$ to_be_communicated_to(world_info, view(X,r), neg, Y)
$\Rightarrow$ M $\models \neg$view(X,r)]

Information providing pro-activeness

a) The agent X is called *weakly information providing pro-active* if:

$\forall \mathcal{M} \in$ Traces(X) $\forall t, r, sign$
[$state_X(\mathcal{M}, t, input(X)) \models$ observation_result(view(X,r), sign)
$\Rightarrow \exists t'$ $state_X(\mathcal{M}, t', output(X)) \models$ to_be_communicated_to(world_info, view(X,r), sign, Y)]

b) The agent X is called *strongly information providing pro-active* if
X is weakly information providing pro-active and observation pro-active.

Information providing reactiveness

a) The agent X is called *weakly information providing reactive* if:

$\forall \mathcal{M} \in$ Traces(X) $\forall t, t', r, sign$
[$state_X(\mathcal{M}, t, input(X)) \models$ observation_result(view(X,r), sign) $\wedge$
$state_X(\mathcal{M}, t', input(X)) \models$ communicated_from(request,view(X,r), pos,Y)]
$\Rightarrow \exists t''>t, t''>t'$ $state_X(\mathcal{M}, t'', output(X)) \models$
to_be_communicated_to(world_info, view(X,r), sign, Y)

b) The agent X is called *strongly information providing reactive* if
X is weakly information providing reactive and observation reactive.

c) The agent X is called *reactive observation info saturating* if
X is request saturating and strongly information providing reactive.

The tree in Figure 10 consists of logical relations between properties of the agent B. Some of them have been defined above, the others are defined as follows.

Information acquisition pro-activeness of an agent

a) The agent X is called *observation pro-active* if:

$\forall \mathcal{M} \in$ Traces(X) $\forall r \; \exists t$ $state_X(\mathcal{M}, t, output(X)) \models$ to_be_observed (view(X,r))]

b) The agent X is called *request pro-active* if for all agents Y different from X:

$\forall \mathcal{M} \in$ Traces(X) $\forall r \; \exists t$ $state_X(\mathcal{M}, t, output(X)) \models$
to_be_communicated_to(request, view(Y,r), pos, Y)]

c) The agent X is called *information acquisition pro-active* if X is observation pro-active and request pro-active.

Observation reactiveness of an agent

The agent X is called *observation reactive* if:

$\forall \mathcal{M} \in$ Traces(X) $\forall t \; \forall r$ [$state_X(\mathcal{M}, t, input(X)) \models$ communicated_from(request, view(X,r), pos, Y)
$\Rightarrow \exists t'$ $state_X(\mathcal{M}, t, output(X)) \models$ to_be_observed (view(X,r))]

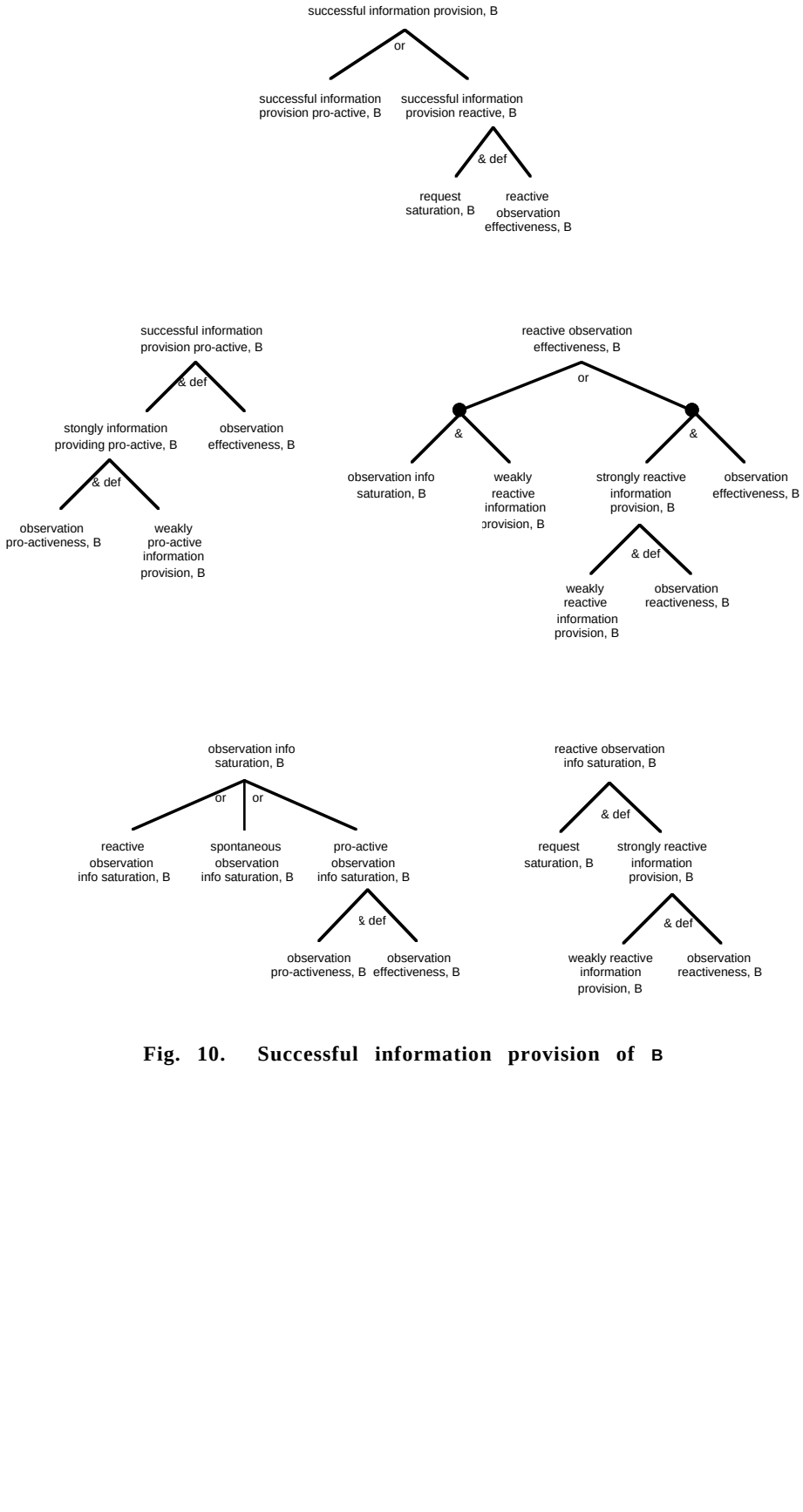

Fig. 10. Successful information provision of B

Information acquisition effectiveness of an agent

a) The agent X is called *observation effective* if:

$\forall \mathcal{M} \in$ Traces(X) $\forall t \; \forall r$ [state$_X(\mathcal{M}, t,$ output(X)) $\models$ to_be_observed (view(X,r))

$\Rightarrow \exists t'>t \; \exists$ sign state$_X(\mathcal{M}, t,$ input(X)) $\models$ observation_result(view(X,r), sign)]

b) The agent X is called *request effective* if:

$\forall \mathcal{M} \in$ Traces(X) $\forall t \; \forall r$

[state$_X(\mathcal{M}, t,$ output(X)) $\models$ to_be_communicated_to(request,view(Y,r),pos,Y)

$\Rightarrow \exists t'>t,$ sign state$_X(\mathcal{M}, t,$ input(X)) $\models$ communicated_from(world_info, view(Y,r), sign, Y)]

c) The agent X is called *information acquisition effective* if
X is observation effective and request effective.

d) The agent X is called *reactive observation effective* if:

$\forall \mathcal{M} \in$ Traces(X) $\forall t \; \forall r$

[state$_X(\mathcal{M}, t,$ input(X)) $\models$ communicated_from(request, view(Y,r), pos, Y)

$\Rightarrow \exists t'>t,$ sign state$_X(\mathcal{M}, t,$ input(X)) $\models$ observation_result(view(X,r), sign)]

e) The agent X is called *pro-active observation info saturating* if
X is observation pro-active and observation effective.

The relations between the environmental properties request saturation of agent A and observation effectiveness of agent A, and properties of the world, of agent B, and of interactions between agents and world can be found in Figure 11.

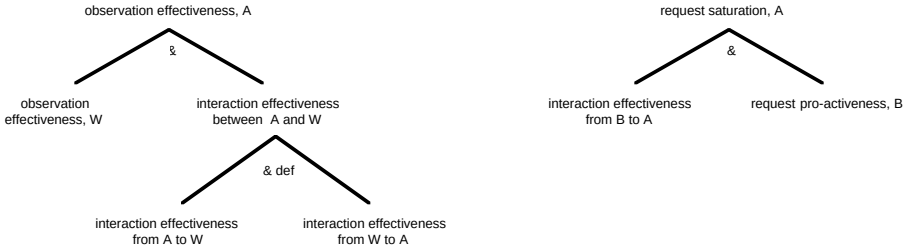

Fig. 11. Observation effectiveness and request saturation of A

5.3 Properties of the World

For the component World assumptions on correctness and conservativity are made.

Correctness of the world

The world W is called *correct* if:

$\forall \mathcal{M} \in$ Traces(W) $\forall t \; \forall v, X$

[state$_W(\mathcal{M}, t,$ output(W)) $\models$ observation_result_for(view(X, r), pos, X) $\Rightarrow$ M $\models$ view(X, r)]

$\wedge \; \forall \mathcal{M} \in$ Traces(W) $\forall t \; \forall v, X$

[state$_W(\mathcal{M}, t,$ output(W)) $\models$ observation_result_for(view(X, r), neg, X) $\Rightarrow$ M $\models \neg$ view(X, r)]

Conservativity of the world

The world W is called *conservative* if:

$\forall \mathcal{M} \in$ Traces(W) $\forall t \; \forall r,$ sign, X

[state$_W(\mathcal{M}, t,$ output(W)) $\models$ observation_result_for(view(X, r), sign, X)

$\Rightarrow \forall t'>t$ state$_W(\mathcal{M}, t',$ output(W)) $\models$ observation_result_for(view(X, r), sign, X)]

Moreover, the world should be effective in providing observation results for observations initiated by the agents.

Observation effectiveness of the world
The component W is called *observation effective* if:

$\forall \mathcal{M} \in$ Traces(W) $\forall t \; \forall r, X$

$[\; state_W(\mathcal{M}, t, input(W)) \models to\_be\_observed\_by(view(X, r), X) \;]$

$\Rightarrow [\; \exists t'>t, sign \;\; state_W(\mathcal{M}, t', output(W)) \models observation\_result\_for(view(X, r), sign, X) \;]$

It is also possible that the world provides observation information without an initiative from the agent (e.g., by automated sensors). In this case the world shows pro-activeness:

Observation pro-activeness of the world
The component W is called *observation pro-active* if:

$\forall \mathcal{M} \in$ Traces(W) $\forall r, X \;\; \exists t, sign$

$state_W(\mathcal{M}, t, output(W)) \models observation\_result\_for(view(X, r), sign, X)$

6 Properties of Agent Components

The properties of the agents needed to prove the properties of the top level of the system were discussed in Section 5.2. The assumed properties of the sub-components of an agent, needed to prove the behavioural properties of that agent, and the logical structure of those proofs in the form of trees are discussed in this section. Properties of the component own process control play a role in the proof of each behavioural agent property.

6.1 Properties of Own Process Control

In the component Own Process Control the agent's attitudes are explicitly represented. The attitudes are represented in the following manner:

attitude	*representation within* OPC
pro-active observation	observation_attitude(pro-active)
weakly pro-active information provision	info_provision_attitude(weakly_pro-active)
strongly pro-active information provision	info_provision_attitude(strongly_pro-active)
pro-active requesting	requesting_attitude(pro-active)
pro-active reasoning	reasoning_attitude(weakly_pro-active)
reactive observation	observation_attitude(reactive)
weakly reactive information provision	info_provision_attitude(weakly_reactive)
strongly reactive information provision	info_provision_attitude(strongly_reactive)

Attitude determination successfulness of OPC
The component OPC is called *attitude determination successful* for the attitude *pro-activeness of observation* if:

$\forall \mathcal{M} \in$ Traces(OPC) $\exists t \; \forall t' \geq t$

$state_{OPC}(\mathcal{M}, t', output(OPC)) \models observation\_attitude(pro\text{-}active)$

In a similar manner attitude determination successfulness for the other attitudes is defined.

In the example the attitudes are assumed to be defined in a static manner, as general facts in OPC. However, it is not difficult to define them dynamically, (i.e., that an agent may change its attitude on the basis of experiences) by specifying a knowledge base that takes into account (dynamic) input for OPC.

Attitude conservativity of OPC
The component OPC is called *attitude conservative* for the attitude *pro-activeness of observation* if:

$\forall \mathcal{M} \in$ Traces(OPC) $\forall t$
$\text{state}_{OPC}(\mathcal{M}, t, \text{output(OPC)}) \models$ observation_attitude(pro-active)
$\Rightarrow \forall t'>t\ \text{state}_{OPC}(\mathcal{M}, t', \text{output(OPC)}) \models$ observation_attitude(pro-active)

In a similar manner attitude conservativity for the other attitudes is defined.

In order to prove observation pro-activeness of agent A not only properties of OPC are needed, but also of the component WIM. The logical relations between these properties can be found in Figure 12.

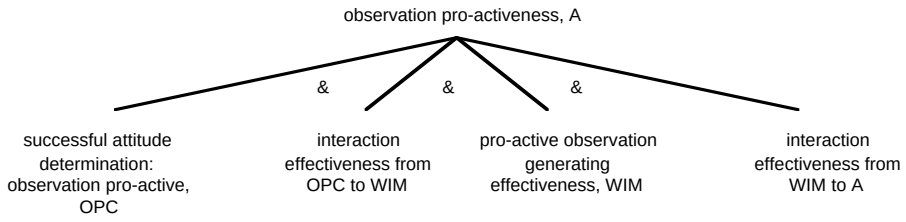

Fig. 12. Observation pro-activeness of A

The properties concerning interaction effectiveness used on this level correspond to the same properties on the top level. Explicit definitions have been omitted in this paper.

6.2 Properties of World Interaction Management

If the agent is pro-active for observation, see Figure 12, then the agent makes sure that every observation is performed at least once. The component world interaction management initiates these observations.

Pro-active observation generation effectiveness of WIM
The component WIM of agent X is called *pro-actively observation generation effective* if:

$\forall \mathcal{M} \in$ Traces(WIM) $\forall t\ \forall r$
$[\ \text{state}_{WIM}(\mathcal{M}, t, \text{input(WIM)}) \models$ observation_attitude(pro-active)
$\Rightarrow \exists t'\ \text{state}_{WIM}(\mathcal{M}, t', \text{output(WIM)}) \models$ to_be_observed (view(X,r))]

Note that no temporal restrictions are put on t': either the observation has been generated in the past (in which case no new observation has to be initiated), or it has to be done now or in the future.
In the reactive case, also the presence of a request is of importance:

Reactive observation generation effectiveness of WIM
The component WIM of agent X is called *reactively observation generation effective* if:

$\forall \mathcal{M} \in$ Traces(WIM) $\forall t \forall r$
[[$state_{WIM}(\mathcal{M}, t, input(WIM)) \models$ observation_attitude(reactive) $\wedge$
$state_{WIM}(\mathcal{M}, t, input(WIM)) \models$ requested(view(X,r))]
$\Rightarrow \exists t'$ $state_{WIM}(\mathcal{M}, t', output(WIM)) \models$ to_be_observed (view(X,r))]

Given this property and the ability of AIM to pass on request information to WIM it is possible to prove observation reativeness of agent A, see Figure 13.

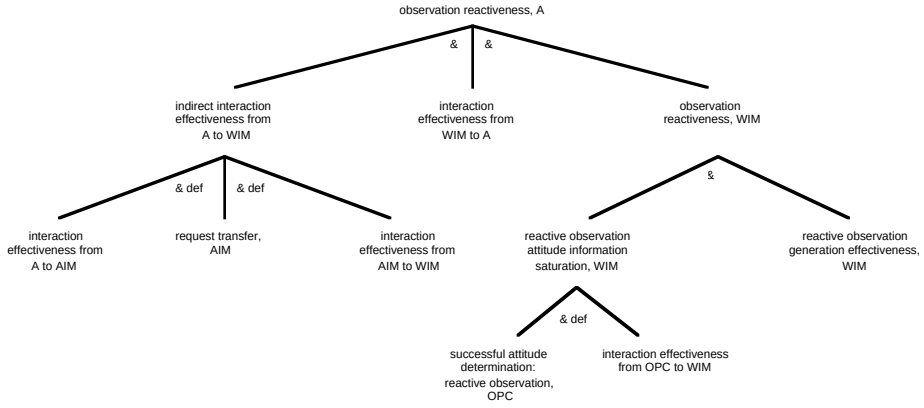

Fig. 13. Observation reactiveness of A

Observation result transfer of WIM
The component WIM of agent X is called *observation result transferring* if:

$\forall \mathcal{M} \in$ Traces(WIM) $\forall t \forall r$, sign
[$state_{WIM}(\mathcal{M}, t, input(WIM)) \models$ observation_result (view(X,r), sign)
$\Rightarrow \exists t' \geq t$ $state_{WIM}(\mathcal{M}, t, output(WIM)) \models$ observation_result (view(X,r), sign)]

This property is used in Figure 14, 15, and 17.

In order to prove weakly pro-active or weakly reactive information provision of A, the properties of the component agent interaction management are of importance.

6.3 Properties of Agent Interaction Management

Pro-active information provision effectiveness of AIM

The component AIM of agent X is called *weakly pro-actively information provision effective* if for every agent Y different from X:

$\forall \mathcal{M} \in$ Traces(AIM) $\forall t \; \forall r$, sign

[$state_{AIM}(\mathcal{M}, t, input(AIM)) \models$ info_provision_attitude(weakly_pro-active) $\wedge$

$state_{AIM}(\mathcal{M}, t, input(AIM)) \models$ observation_result (view(X,r), sign)

$\Rightarrow \exists t' \; state_{AIM}(\mathcal{M}, t, output(AIM)) \models$ to_be_communicated_to(world_info, view(X,r), pos, Y)]

Figure 14 shows how the agent property weakly pro-active information provision depends on other properties of AIM. The component AIM needs observation information, therefore, the observation result transfer property of WIM, and effective interaction from the input of the agent to WIM, and from WIM to AIM should hold. The correct necessary attitude information is provided by OPC.

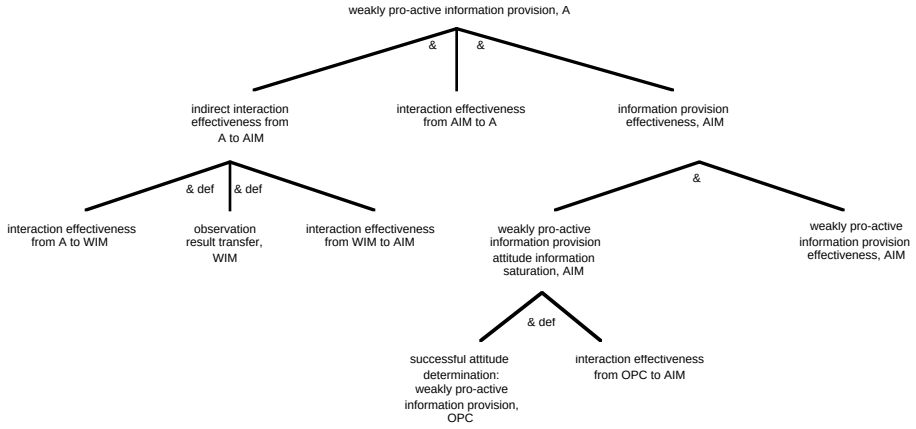

Fig. 14. Weakly pro-active information provision of A

Reactive information provision effectiveness of AIM

The component AIM of agent X is called *weakly reactively information provision effective* if for every agent Y different from X:

$\forall \mathcal{M} \in$ Traces(AIM) $\forall t,t' \; \forall r$, sign

[$state_{AIM}(\mathcal{M}, t, input(AIM)) \models$ info_provision_attitude(weakly_reactive) $\wedge$

$state_{AIM}(\mathcal{M}, t, input(AIM)) \models$ observation_result (view(X,r), sign) $\wedge$

$state_{AIM}(\mathcal{M}, t', input(AIM)) \models$ requested(view(X,r))

$\Rightarrow \exists t'' \geq t' \; state_{AIM}(\mathcal{M}, t'', output(AIM)) \models$

to_be_communicated_to(world_info, view(X,r), pos, Y)]

Similarly, the reactive information provision effectiveness property of AIM is needed to prove the agent property weakly reactive information provision, see Figure 15.

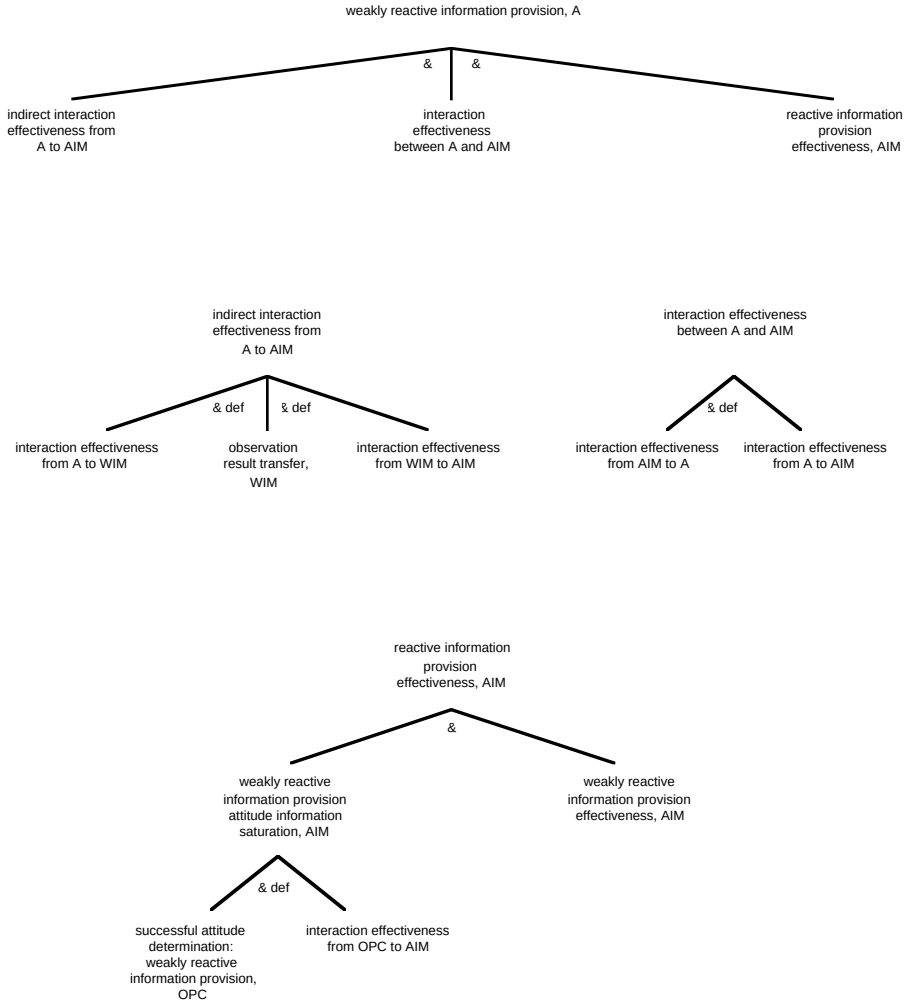

Fig. 15. Weakly reactive information provision of A

Request transfer of AIM

The component AIM of agent X is called *request transferring* if for every agent Y different from X:

$\forall \mathfrak{M} \in$ Traces(AIM) $\forall t \; \forall r$

[$state_{AIM}(\mathfrak{M}, t, input(AIM)) \models$ communicated_by (request, view(X,r), pos, Y)

$\Rightarrow \exists t' \geq t \; state_{AIM}(\mathfrak{M}, t, output(AIM)) \models$ requested(view(X,r))]

This property is used in Figure 13 and in Figure 17. The following property can be used to prove request pro-activeness of agent A, see Figure 16.

Pro-active request generation effectiveness of AIM
The component AIM of agent X is called *pro-actively request generation effective* if for every agent Y different from X:

$\forall \mathcal{M} \in$ Traces(AIM) $\forall t \; \forall r$
[$\text{state}_{AIM}(\mathcal{M}, t, \text{input(AIM)}) \models$ requesting_attitude(pro-active)
$\Rightarrow \exists t' \; \text{state}_{AIM}(\mathcal{M}, t, \text{output(AIM)}) \models$ to_be_communicated_to(request, view(Y,r), pos, Y)]

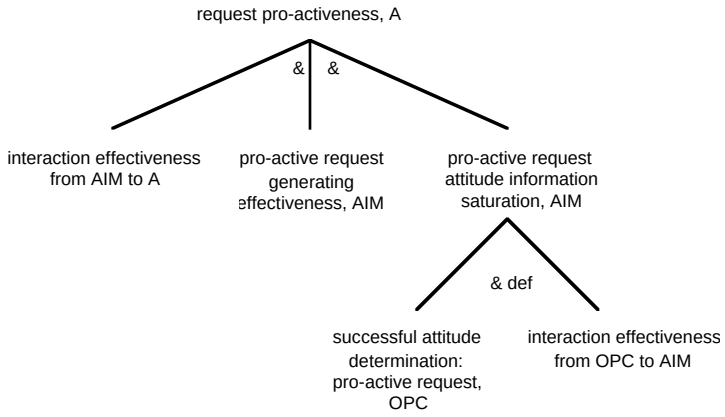

Fig. 16. Request pro-activeness of A

Communicated information transfer of AIM
The component AIM of agent X is called *communicated information transferring* if for every agent Y different from X:

$\forall \mathcal{M} \in$ Traces(AIM) $\forall t \; \forall r, \text{sign}$ [$\text{state}_{AIM}(\mathcal{M}, t, \text{input(AIM)}) \models$
communicated_by (world_info, view(Y,r), sign, Y)
$\Rightarrow \exists t' \geq t \; \text{state}_{AIM}(\mathcal{M}, t, \text{output(AIM)}) \models$ received_world_info(view(Y,r), sign)]

6.4 Properties of the Agent Specific Task: OC

The required properties of the component OC are the following. Conclusiveness defines that the component is able to draw decisive conclusions if sufficient input is provided.

Conclusiveness of OC
The component OC is called *conclusive* if, under the condition that all required input information has been acquired, for every output atom a conclusion is derived.

$\forall \mathcal{M} \in$ Traces(OC) [$\forall Y \; \exists r,t \; \text{state}_{OC}(\mathcal{M}, t, \text{input(OC)}) \models$ view(Y,r)]
$\Rightarrow \forall s$ [$\exists t' \; \text{state}_{OC}(\mathcal{M}, t', \text{output(OC)}) \models$ object_type(s) $\vee$
$\exists t' \; \text{state}_{OC}(\mathcal{M}, t', \text{output(OC)}) \models \neg$object_type(s)]

To allow that OPC controls the reasoning on the basis of its reasoning attitude, the following conditional variant of conclusiveness is needed. This means that only conclusions are drawn if OC has been input (transferred from OPC) the right targets. Conditional conclusiveness is used to prove conclusion pro-activeness of agent A in Figure 17.

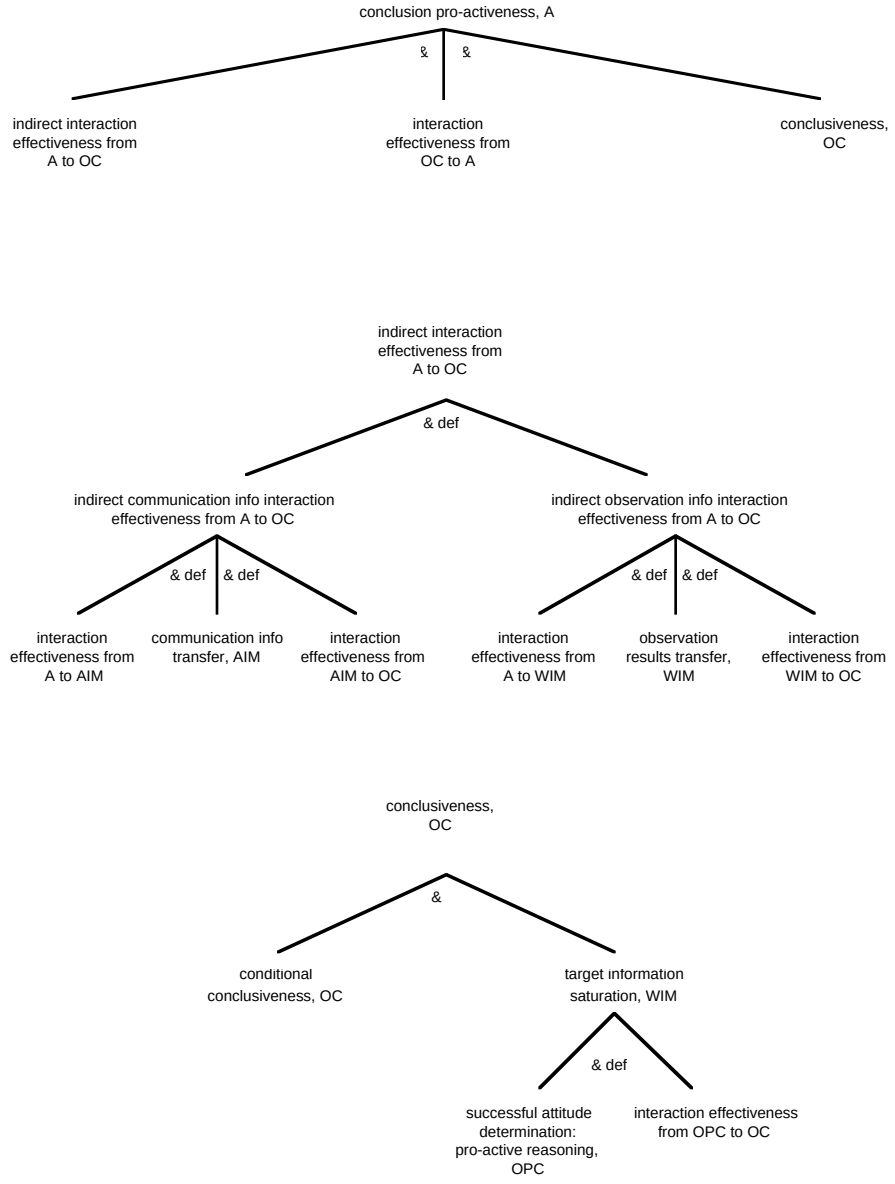

Fig. 17. Conclusion pro-activeness of A

Conditional conclusiveness of oc
The component oc is called *conditionally conclusive* if, under the condition that all required input information has been acquired, for every output atom which is associated to its focus (as a target), a conclusion is derived:

$\forall \mathcal{M} \in$ Traces(OC) $\forall$s
[$\forall$Y $\exists$ r,t $state_{OC}(\mathcal{M}, t, input(OC)) \models view(Y,r)$] $\wedge$
[$state_{OC}(\mathcal{M}, t, input(OC)) \models$ target(OC_focus, object_type(s), determine)
$\Rightarrow$ [$\exists$t' $state_{OC}(\mathcal{M}, t', output(OC)) \models$ object_type(s) $\vee$
$\exists$t' $state_{OC}(\mathcal{M}, t', output(OC)) \models \neg$object_type(s)]

Conclusion correctness means: if a conclusion is derived, then this conclusion corresponds to the world situation.

Conclusion correctness of oc
The component OC is called *conclusion correct* if:

$\forall \mathcal{M} \in$ Traces(OC) $\forall$t $\forall$s
[$state_{OC}(\mathcal{M}, t, output(OC)) \models$ object_type(s) $\Rightarrow$ M $\models$ object_type(s)]
$\wedge$ [$state_{OC}(\mathcal{M}, t, output(OC)) \models \neg$ object_type(s) $\Rightarrow$ M $\models \neg$object_type(s)]

Conservation can be defined by:

Conclusion conservativity of oc
The component OC is called *conclusion conservative* if:

$\forall \mathcal{M} \in$ Traces(X) $\forall$t $\forall$s
[$state_{OC}(\mathcal{M}, t, output(OC)) \models$ object_type(s) $\Rightarrow$
$\forall$t'>t $state_{X}(\mathcal{M}, t', output(OC)) \models$ object_type(s)]
$\wedge$ [$state_{OC}(\mathcal{M}, t, output(OC)) \models \neg$ object_type(s) $\Rightarrow$
$\forall$t'>t $state_{OC}(\mathcal{M}, t', output(OC)) \models \neg$ object_type(s)]

6.5 Domain Assumptions

The properties also need assumptions on the domain knowledge to be used in the model.

Static world
The world state is static during the processing of the system s.

Empirically foundedness
The possible conclusions can be uniquely characterised by means of observations; in other words: if two world situations satisfy exactly the same observations, then they also satisfy exactly the same conclusions (see Treur and Willems, 1994).

7 Verification of Primitive Components

In Sections 5 and 6 verification of the multi-agent model was described, based on assumed properties of the primitive components. The primitive components can be verified making use of the more standard methods introduced in (Treur and Willems, 1994; Leemans, Treur and Willems, 1993). For example, the component Object Classification should satisfy conclusion correctness and conditional conclusiveness.

Actually, these two properties reduce to static properties described in (Treur and Willems, 1994). In fact all properties required for primitive components reduce to static properties that define a constraint on the combined input-output states of the component. Such properties can be verified by the (static) methods described in the references mentioned.

8 Conclusions

The modelling approach DESIRE is based on compositionality of processes and knowledge at different levels of abstraction. The compositional verification method described in this paper fits well to DESIRE, but can also be useful to any other compositional modelling approach. Two main advantages of a compositional approach to modelling are the transparent structure of the design and support for reuse of components and generic models. The compositional verification method extends these main advantages to (1) a well-structured verification process, and (2) the reusability of proofs for properties of components that are reused.

The first advantage entails that both conceptually and computationally the complexity of the verification process can be handled by compositionality at different levels of abstraction. Apart from the work reported here, a generic model for diagnosis has been verified (Cornelissen, Jonker and Treur, 1997) and a multi-agent system with agents negotiating about load-balancing of electricity use. The second advantage entails: if a modified component satisfies the same properties as the previous one, the proof of the properties at the higher levels of abstraction can be reused to show that the new system has the same properties as the original. This has high value for a library of reusable generic models and components. The verification of generic models forces one to find the assumptions under which for the considered domain the generic model is applicable, as is also discussed in (Fensel, 1995; Fensel and Benjamins, 1996). A library of reusable components and task models may consist of both specifications of the components and models, and their design rationale. As part of the design rationale, at least the properties of the components and their logical relations can be documented.

Also due to the compositional nature of the verification method, a distributed approach to verification is facilitated. This implies that several persons can work on the verification of the same system at the same time, once the properties to be verified have been determined. Since the proof of properties of a composed component depends on the properties of its sub-components, it is only necessary to know or to agree on the properties of these sub-components.

The formal analysis of variants of reactiveness and pro-activeness properties deepened our insight in these notions and their logical relationships and interactions. Semantical formalisation of different variants of reactiveness and pro-activeness have been found in the form of conditional temporal statements. The notion of information and process hiding, in DESIRE modelled in terms of components at different abstraction levels, made it possible to distinguish in a natural manner between observable and non-observable variants of pro-activeness and reactiveness: the variants of behaviour that can be observed from outside the agent (at its interface), and the variants of internal behaviour (in its sub-components and interactions between them) that cannot be observed from outside. This formal analysis could be a starting point

for a more general mathematical or logical theory on pro-activeness and reactiveness, and their interaction. Actually, the logical relations, in this paper depicted in the form of AND/OR graphs in the figures, can be viewed as lemmas and theorems in such a theory.

A main difference in comparison to (Fisher and Wooldridge, 1997) is that our approach exploits compositionality. An advantage of their approach is that they can make use of a temporal belief logic. It would be a challenge to extend the approach as referred to a compositional variant of temporal belief logic. A first step in this direction can be found in (Engelfriet, Jonker and Treur, 1997). Also a main difference of the current paper in comparison to the work in (Fensel, 1995, Fensel and Benjamins, 1996; Fensel et al, 1996) is that in our approach compositionality of the verification is addressed; in the work as referred only domain assumptions are taken into account, and no hierarchical relations between properties are defined.

A future continuation of this work will consider the development of tools for verification. At the moment only tools exist for the verification of primitive components; no tools for the verification of composed components exist yet. To support the handwork of verification it would be useful to have tools to assist in the creation of the proof. This could be done by formalising the proofs of a verification process using a first order logic in which time and states are represented explicitly, and an interactive theorem prover to support the proofs. Another option that will be explored is to extend Fisher and Wooldridge's approach to the compositional case. Yet another option to be explored is whether the tool KIV (based on dynamic logic) can be used. Some first, positive experiences with KIV for verification of an example model of a knowledge-based system are reported in (Fensel et al, 1996).

Acknowledgements

Wieke de Vries has read an earlier version of this paper, which has led to a number of improvements in the text.

References

Abadi, M. and L. Lamport (1993). Composing Specifications, ACM Transactions on Programming Languages and Systems, Vol. 15, No. 1, p. 73-132.

Benjamins, R., Fensel, D., Straatman, R. (1996). Assumptions of problem-solving methods and their role in knowledge engineering. In: W. Wahlster (ed.), Proceedings of the 12th European Conference on AI, ECAI'96, John Wiley and Sons, pp. 408-412.

Brazier, F.M.T. , Dunin-Keplicz, B., Jennings, N.R. and Treur, J. (1995). 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, pp. 25-32. Extended version in: International Journal of Cooperative Information Systems, M. Huhns, M. Singh, (Eds.), special issue on Formal Methods in Cooperative Information Systems: Multi-Agent Systems, vol. 6, 1997, pp. 67-94.

Brazier, F.M.T., Treur, J., Wijngaards, N.J.E. and Willems, M. (1996). Temporal semantics of complex reasoning tasks. In: B.R. Gaines, M.A. Musen (eds.), Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-based Systems workshop, KAW'96, Calgary: SRDG Publications, Department of Computer Science, University of Calgary, pp. 15/1-15/17. Extended version to appear in Data and Knowledge Engineering, 1998

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

Dams, D., Gerth, R., Kelb, P. (1996). Practical Symbolic Model Checking of the full μ-calculus using Compositional Abstractions. Report, Eindhoven University of Technology, Department of Mathematics and Computer Science.

Engelfriet, J., Jonker, C.M., Treur, J., (1997). Compositional Verification of Knowledge-based Systems in Temporal Epistemic Logic. In: A. Bossi (ed.), Proceedings of the ILPS'97 Workshop on Verification.

Fensel, D. (1995). Assumptions and limitations of a problem solving method: a case study. In: B.R. Gaines, M.A. Musen (eds.), Proceedings of the 9th Banff Knowledge Acquisition for Knowledge-based Systems workshop, KAW'95, Calgary: SRDG Publications, Department of Computer Science, University of Calgary.

Fensel, D., Benjamins, R. (1996). Assumptions in model-based diagnosis. In: B.R. Gaines, M.A. Musen (eds.), Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-based Systems workshop, KAW'96, Calgary: SRDG Publications, Department of Computer Science, University of Calgary, pp. 5/1-5/18.

Fensel, D., Schonegge, A., Groenboom, R., Wielinga, B. (1996). Specification and verification of knowledge-based systems. In: B.R. Gaines, M.A. Musen (eds.), Proceedings of the 10th Banff Knowledge Acquisition for Knowledge-based Systems workshop, KAW'96, Calgary: SRDG Publications, Department of Computer Science, University of Calgary, pp. 4/1-4/20.

Fisher, M., Wooldridge, M. (1997) On the Formal Specification and Verification of Multi-Agent Systems. International Journal of Cooperative Information Systems, M. Huhns, M. Singh, (eds.), special issue on Formal Methods in Cooperative Information Systems: Multi-Agent Systems, vol. 6, pp. 67-94.

Harmelen, F. van and Fensel, D. (1995). Formal Methods in Knowledge Engineering. Knowledge Engineering Review, Volume 10, Number 4.

Hooman, J. (1994). Compositional Verification of a Distributed Real-Time Arbitration Protocol. Real-Time Systems, vol. 6, pp. 173-206.

Kinny, D., Georgeff, M.P., Rao, A.S. (1996). A Methodology and Technique for Systems of BDI Agents. In: W. van der Velde, J.W. Perram (eds.), Agents Breaking Away, Proceedings 7th European Workshop on Modelling Autonomous Agents in a Multi-Agent World, MAAMAW'96, Lecture Notes in AI, vol. 1038, Springer Verlag, pp. 56-71.

Langevelde, I.A. van, A. Philipsen, and J. Treur (1992). Formal Specification of Compositional Architectures. In B. Neumann (ed.), Proceedings of the 10th European Conference on AI, ECAI'92, Wiley and Sons, pp. 272-276.

Leemans, P., J. Treur, and M. Willems (1993). On the verification of knowledge-based reasoning modules, Report IR-346, Department of Mathematics & Computer Science, AI Group, Vrije Universiteit Amsterdam.

Rao, A.S. and Georgeff, M.P. (1991). Modeling rational agents within a BDI architecture. In: R. Fikes and E. Sandewall (eds.), Proceedings of the Second Conference on Knowledge Representation and Reasoning, Morgan Kaufman, pp. 473-484.

Treur, J., and M. Willems (1994). A logical foundation for verification. In: Proceedings of the 11th European Conference on AI, ECAI'94, A. Cohn (ed.), John Wiley & Sons, Ltd., pp. 745-749.

Wooldridge, M., N.R. Jennings (eds.) (1995), Intelligent Agents, Proceedings of the First International Workshop on Agent Theories, Architectures and Languages, Lecture Notes in AI, vol. 890, Springer Verlag.