The following full text is a publisher's version.

For additional information about this publication click this link.
http://hdl.handle.net/2066/63995

Please be advised that this information was generated on 2017-11-19 and may be subject to change.
Compositional Design and Verification of a Multi-Agent System for One-to-Many Negotiation

Frances Brazier, Frank Cornelissen, Rune Gustavsson, Catholijn M. Jonker, Olle Lindeberg, Bianca Polak, Jan Treur

1Vrije 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 /{frances, frankc, jonker, treur} Email: {frances, frankc, jonker, treur}@cs.vu.nl

2University of Karlskrona/Ronneby (HK/R)
Department of Computer Science and Business Administration, Research Laboratory SIKT
URL: http://www.sikt.hk-r.se Email: {Rune.Gustavsson, Olle.Lindeberg}@ide.hk-r.se

Abstract
A compositional verification method for multi-agent systems is presented and applied to a multi-agent system for one-to-many negotiation in the domain of load balancing of electricity use. Advantages of the method are that the complexity of the verification process is managed by compositionality, and that parts of the proofs can be reused in relation to reuse of components.

1. Introduction
When designing multi-agent systems, it is often difficult to guarantee that the specification of a system actually fulfills 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 has 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, a recent trend in the literature on verification is to exploit compositionality and abstraction to structure the process of verification; cf. [5], [10], [11], [12].

The development of structured modeling frameworks and principled design methods tuned to the specific area of multi-agent systems is currently underway; e.g., [5], [9], [13]. Mature multi-agent system design methods should include a verification approach. For example, in [9] verification is addressed using a temporal belief logic. In the approach presented below, a compositional verification method for multi-agent systems (cf. [12]) is used for formal analysis of a multi-agent system for one-to-many negotiation, in particular for load balancing of electricity use; see [4]. In short, the properties of the whole system are established by derivation from assumptions that themselves are properties of agents, which in turn may be derived from assumptions on sub-components of agents, and so on. The properties are formalised in terms of temporal semantics. The multi-agent system described and verified in this paper has been designed using the compositional development method for multi-agent systems DESIRE; cf. [5].

2. Compositional Verification
The purpose of verification is to prove that, under a certain set of conditions (assumed properties), a system will adhere to a certain set of desired properties, for example the design requirements. In the compositional verification approach presented in this paper, 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 assumed properties implies the properties that it needs to fulfill.

2.1. The Compositional Verification Method
A compositional multi-agent system can be viewed at different levels of process abstraction. Viewed from the top level, denoted by $L_0$, the complete system is one component $s$; internal information and processes are hidden. At the next, lower level of abstraction, the system component $s$ can be viewed as a composition of agents and the world. Each agent is composed of its sub-components, and so on. The compositional verification method takes this compositional structure into account. Verification of a composed component is done using:
- properties of the sub-components it embeds,
- the way in which the component is composed of its sub-components (the composition relation),
- environmental properties of the component (depending on the rest of the system, including the world).

Given the specification of the composition relation, the assumptions under which the component functions properly are the environmental properties and the properties to be proven for its sub-components. This implies that properties at different levels of process...
abstraction are involved in the verification process. The primitive components (those that are not composed of other components) can be verified using more traditional verification methods. Often the properties involved are not given at the start: to find them is one of the aims of the verification process.

The verification proofs that connect properties of one process abstraction level with properties of the other level 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 only the proof for the new component has to be provided. In this sense the verification method supports reuse of verification proofs. The compositional verification method can be formulated as follows:

A. Verifying one 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 assumed properties (at the lower level) are needed to guarantee the properties of the higher level, and which environment properties.
3. Prove the properties of the higher level on the basis of these assumed properties, and the environment properties.

B. Verifying a Primitive Component
For primitive components, verification techniques can be used that are especially tuned to the type of component; both for primitive knowledge-based components and non-knowledge-based components (such as databases or optimisation algorithms) techniques (and tools) can be found in the literature.

C. The Overall Verification Process
To verify the entire system
1. Determine the properties that are desired for the whole system.
2. Apply A iteratively. In the iteration the desired properties of each abstraction level i are the assumed properties for the higher level.
3. Verify the primitive components according to B.

Notes:
- The results of verification are two-fold:
  1. Properties at the different abstraction levels.
  2. The logical relations between the properties of adjacent abstraction levels.
- 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 li is composed from the descriptions at the lower abstraction level li,1; the compositional development method for multi-agent systems DESIRE fulfils this requirement.
- in principle different procedures can be followed (e.g., top-down, bottom-up or mixed).

2.2. Semantics behind Compositional Verification
Verification is always relative to semantics of the system descriptions to be 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 more detailed formal definition can be found in [6]. The set of all possible information states of D is denoted by IS(D).

A trace  may of a component D is a sequence of information states  in IS(D). The set of all traces (i.e., ) is denoted by Traces(D). Given a trace 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, C, t) = IS(D, C, t), where C is either D or a sub-component of D. Analogously, state(D, C, t, output(C)) denotes the information state of the output interface of component C at time point t of the component D.

3. One-to-many Negotiation Processes
In this section the application domain is briefly sketched, and the one-to-many negotiation process devised within this domain is presented.

3.1. Load Balancing of Electricity Use
The purpose of load management of electricity use is to smoothen peak load by managing a more appropriate distribution of the electricity use among consumers. Flexible pricing schemes can be an effective means to influence consumer behaviour; cf. [10]. The assumption behind the model presented in this paper is that, to acquire a more even distribution of electricity usage in time, consumer behaviour can be influenced by financial gain. Consumers are autonomous in the process of negotiation: each individual consumer determines which price/risk he/she is willing to take and when. As consumers are all individuals with their own characteristics and needs (partially defined by the type of equipment they use within their homes), that vary over time, models of consumers used to design systems to support the consumer, need to be adaptive and flexible (cf. [2]). Utility companies negotiate price in a one-to-many negotiation process with each and every individual separately, unaware of the specific models behind such systems for individuals. In the model discussed in this paper the negotiation process is modelled for one utility company and a number of consumers, each with their own respective agent to support them in the negotiation process: one Utility Agent and a number of Customer Agents.

3.2. Modelling the Negotiation Process
In [14, 15] a number of mechanisms for negotiation are described. A protocol with well-defined properties, called the monotonic concession protocol, is described: during a negotiation process all proposed deals must be equally or
more acceptable to the counter party than all previous
deals proposed. The strength of this protocol is that the
negotiation process always converges. The monotonic
concession protocol has been applied to the load
management problem, to obtain a model for the one-to-
many negotiation process between one Utility Agent and a
(in principle large) number of Customer Agents.

In this model, the Utility Agent always initiates the
negotiation process, as soon as a coming peak in the
electricity consumption is predicted. In the method used
the Utility Agent constructs a so-called reward table and
communicates this table to all Customer Agents
(announcement). A reward table (for a given time interval)
consists of a list of possible cut-down values, and a reward
value assigned to each cut-down value. The cut-down value
specifies an amount of electricity that can be saved
(expressed in percentages) and the reward value specifies
the amount of reward the Customer Agent will receive
from the Utility Agent if it lowers its electricity
consumption by the cut-down value. A Customer Agent
examines and evaluates the rewards for the different cut-
down values in the reward tables. If the reward value
offered for the specific cut-down is acceptable to the
Customer Agent, it informs the Utility Agent (bid) that it
is prepared to make a cut-down $x$, which may be zero to
express that no cut-down is offered.

As soon as the Customer Agents have responded to the
announcement of a reward table, the Utility Agent predicts
the new balance between consumption and production of
electricity for the stated time interval. The Utility Agent is
satisfied by the responses if a peak can be avoided if all
Customer Agents implement their bids. If the Utility
Agent is not satisfied by the responses communicated by
the Customer Agents, it announces a new reward table
(according to the monotonic concession protocol
mentioned above) to the Customer Agents in which the
reward values are at least as high, and for some cut-down
values higher than in the former reward table (determined
on the basis of, for example, the formulæ described in
Section 4.2 below). The Customer Agents react to this
new announcement by responding with a new bid or the
same bid again (in line with the rules of the monotonic
concession protocol). This process continues until (1) the
peak is satisfactorily low for the Utility Agent (at most
the capacity of the utility company), or (2) the reward
values in the new reward table have (almost) reached the
maximum value the Utility Agent can offer. This value
has been determined in advance. For more details on this
negotiation method, see [4].

4. Compositional Design of the System

The prototype Multi-Agent System has been fully
specified and (automatically) implemented in the DESIRE
software environment. The top level composition of the
system consists of a Utility Agent, two Customer Agents,
and an External World.

4.1. Top Level Composition of the Utility Agent

The first level composition within the Utility Agent is
depicted in Figure 1 (taken from the graphical design tool
within the DESIRE software environment). This picture
shows part of the graphical interface of the DESIRE
software environment; in addition, interfaces to the agents
have been implemented which are specific for this
prototype (see [4]).

4.2. Knowledge used within the Utility Agent

In this prototype system the Utility Agent communicates
the same announcements to all Customer Agents, in
compliance with Swedish law. The predicted balance
between the consumption and the production of electricity,
is determined by the following formulæ (here CA is a
variable ranging over the set of Customer Agents):

\[
predicted\_use\_with\_cutdown(CA) = \begin{cases}
predicted\_use(CA) & \text{if } (1 - cutdown(CA)) \cdot allowed\_use(CA) \geq predicted\_use(CA) \\
(1 - cutdown(CA)) \cdot allowed\_use(CA) & \text{otherwise}
\end{cases}
\]

\[
predicted\_overuse = \sum_{CA} predicted\_use\_with\_cutdown(CA) - normal\_use\_overuse
\]

\[
new\_reward = reward + \beta \cdot overuse \cdot (1 - reward/max\_reward).
\]

Figure 1 Process composition at the first level within the Utility Agent
In the prototype system, the factor beta determines how steeply the reward values increase; in the current system it has a constant value. The reward value increases more when the predicted overuse is higher (in the beginning of the negotiation process) and less if the predicted overuse is lower. It never exceeds the maximal reward, due to the logistic factor (1 - reward/max_reward).

5. Verification at the Top Level

Two important assumptions behind the system are: energy use is (statistically) predictable at a global level, and consumer behaviour can be influenced by financial gain. These assumptions imply that if the financial rewards (calculated on the basis of statistical information) offered by a Utility Agent are well chosen, Customer Agents will respond to such offers and decrease their use.

The most important properties to prove for the load balancing system S as a whole are that (1) the negotiation process satisfies the monotonic concession protocol, (2) at some point in time the negotiation process will terminate, and (3) the agents make rational decisions during the negotiation process. These properties are formally defined in Section 5.1. An important property for the Utility Agent, in particular, is that after the negotiation process the predicted overuse has decreased to such an extent that it is at most the maximal overuse the utility company considers acceptable. To prove these properties several other properties of the participating agents (and the external world) are assumed. These properties of agents and the external world are defined in Section 5.2. Some of the proofs of properties are briefly presented in Section 5.3.

Next, Section 6 shows how these assumed properties can be proven from properties assumed for the subcomponents of the agents.

5.1 Properties of the System as a Whole

The properties defined at the level of the entire system are based on combinations of properties of the agents.

S1. Monotonicity of negotiation

The system S satisfies monotonicity of negotiation if the Utility Agent satisfies monotonicity of announcements and each Customer Agent satisfies monotonicity of bids. This is formally defined as the conjunction of the Utility Agent announce monotonicity property U7 and for each Customer Agent the bid monotonicity property C5 (see below).

S2. Termination of negotiation

The system S satisfies termination of negotiation (on a given time interval) if a time point exists after which no announcements or bids (referring to the given time interval) are generated by the agents. This is formally defined by: for all Customer Agents CA it holds

\[
\forall \alpha \in \text{Traces}(S) \exists t \forall \forall t', \forall t' \leq t, \forall C \in \alpha \text{ announce} \Rightarrow \text{termination}(C, t, \alpha)
\]

S3. Rationality of negotiation

The system S satisfies rationality of negotiation if the Utility Agent satisfies announcement rationality and each Customer Agent satisfies bid rationality. This is formally defined as the conjunction of the Utility Agent rationality property U9 and for each Customer Agent the Customer Agent rationality property C4 defined below.

5.4 Required reward limitation

The system S satisfies required reward limitation if for each Customer Agent and each cut-down percentage, the required reward of the Customer Agent is at most the maximal reward that can be offered by the Utility Agent.

\[
\forall \alpha \in \text{CA} \forall CD \forall \alpha(CA) \leq \alpha(UA)
\]

5.2 Properties of the Agents and the World

The properties of the Utility Agent, the Customer Agents, and the External World are defined in this section. Note that each of the properties is presented as a temporal statement either about all traces of the system S or about all traces of an agent. In the latter case the truth of the property does not depend on the environment of the agent. Section 5.3 discusses how the various properties are logically related.

5.2.1 Properties of the Utility Agent

U1. Successfulness of negotiation

The Utility Agent satisfies successfulness of negotiation if at some point in time t and for some negotiation round n the predicted overuse is less than or equal to the constant max_overuse.

\[
\forall \alpha \in \text{Traces}(S) \exists t \forall U \leq \max_overuse \rightleftharpoons \text{predicted_overuse}(U, N)
\]

U2. Negotiation round generation effectiveness

The Utility Agent satisfies negotiation round generation effectiveness if the following holds: if and when predicted overuse is higher than the maximal overuse, a next negotiation round is initiated.

\[
\forall \alpha \in \text{Traces}(U) \forall t, U, \forall C, \forall R
\]

U3. Negotiation round generation groundedness

The Utility Agent satisfies negotiation round generation groundedness if the following holds: if the predicted overuse is at most the maximal overuse, then no new negotiation round is initiated.

\[
\forall \alpha \in \text{Traces}(U) \forall t, U \leftarrow \max_overuse \rightleftharpoons \text{predicted_overuse}(U, N)
\]
U4. Announcement generation effectiveness
The Utility Agent satisfies announcement generation effectiveness if for each initiated negotiation round at least one announcement is generated.
\[ \forall \alpha \in Traces(UA) \exists \tau, N \exists CD, R, R' \]
\[ \forall \tau, N \exists CD, R, R' \]
\[ stateUA(M, t, output(UA)) \Leftrightarrow announcement(CD, R, N) \]
\[ \Rightarrow \exists \tau t, CD, R, R' \]
\[ stateUA(M, t, output(UA)) \Leftrightarrow announcement(CD, R, N) \]
\[ \Rightarrow \exists \tau t, CD, R, R' \]

U5. Announcement uniqueness
The Utility Agent satisfies announcement uniqueness if for each initiated negotiation round at most one announcement is generated.
\[ \forall \alpha \in Traces(UA) \forall \tau, N \forall CD, R, R' \]
\[ stateUA(M, t, output(UA)) \Leftrightarrow announcement(CD, R, N) \]
\[ \Rightarrow \exists ! \tau t, CD, R, R' \]

U6. Announcement generation groundedness
The Utility Agent satisfies announcement generation groundedness if an announcement is only generated for initiated negotiation rounds.
\[ \forall \alpha \in Traces(UA) \forall \tau, N \exists CD, R, R' \]
\[ stateUA(M, t, output(UA)) \Leftrightarrow announcement(CD, R, N) \]
\[ \Rightarrow \exists \tau t, CD, R, R' \]

U7. Monotonicity of announcement
The Utility Agent satisfies monotonicity of announcement if for each announcement and each cut-down percentage the offered reward is at least the reward for the same cut-down percentage offered in the previous announcements.
\[ \forall \alpha \in Traces(UA) \forall \tau, N \exists CD, R, R' \]
\[ stateUA(M, t, output(UA)) \Leftrightarrow announcement(CD, R, N) \]
\[ \Rightarrow \exists \tau t, CD, R, R' \]

U8. Progress in announcement
The Utility Agent satisfies progress in announcement if for at least one cut-down percentage the difference between the currently announced reward and the previously announced reward is at least the positive constant m (announce margin).
\[ \forall \alpha \in Traces(UA) \forall \tau, N \exists CD, R, R' \]
\[ stateUA(M, t, output(UA)) \Leftrightarrow announcement(CD, R, N) \]
\[ \Rightarrow \exists \tau t, CD, R, R' \]

U9. Announcement rationality
The Utility Agent satisfies announcement rationality if no announced reward is higher than the maximal reward plus the announce margin.
\[ \forall \alpha \in Traces(UA) \forall \tau, N \exists CD, R, R' \]
\[ stateUA(M, t, output(UA)) \Leftrightarrow announcement(CD, R, N) \]
\[ \Rightarrow \exists \tau t, CD, R, R' \]

U10. Finite termination of negotiation by UA
The Utility Agent satisfies finite termination of negotiation if a time point exists such that UA does not negotiate anymore after this time point.
\[ \forall \alpha \in Traces(UA) \exists ! \tau t, CD, R, N \]
\[ stateUA(M, t, output(UA)) \Leftrightarrow announcement(CD, R, N) \]

5.2.2 Properties of each Customer Agent
C1. Bid generation effectiveness
A Customer Agent CA satisfies bid generation effectiveness if for each announced negotiation round at least one bid is generated (possibly a bid for reduction zero).
\[ \forall \alpha \in Traces(CA) \forall \tau, N \exists CD, R, CD' \]
\[ stateCA(M, t, output(CA)) \Leftrightarrow announcement(CD, R, N) \]
\[ \Rightarrow \exists \tau t, CD, R, CD' \]

C2. Bid uniqueness
A Customer Agent CA satisfies bid uniqueness if for each negotiation round at most one bid is generated.
\[ \forall \alpha \in Traces(CA) \forall \tau, N, CD, CD' \]
\[ stateCA(M, t, output(CA)) \Leftrightarrow cutdown(CD, N) \]
\[ \Rightarrow \exists \tau t, CD, CD' \]

C3. Bid generation groundedness
A Customer Agent CA satisfies bid generation groundedness if a bid is only generated once a negotiation round is announced.
\[ \forall \alpha \in Traces(CA) \forall \tau, N, CD \]
\[ stateCA(M, t, output(CA)) \Leftrightarrow cutdown(CD, N) \]
\[ \Rightarrow \exists \tau t, CD \]

C4. Bid rationality
A Customer Agent CA satisfies bid rationality if for each bid the required reward for the offered cut-down is at most the reward announced in the same round, and the offered cut-down is the highest with this property.
\[ \forall \alpha \in Traces(CA) \forall \tau, N, CD \]
\[ [ stateCA(M, t, output(CA)) \Leftrightarrow cutdown(CD, N) \]
\[ \Rightarrow \exists \tau t, CD \]

C5. Monotonicity of bids
A Customer Agent CA satisfies monotonicity of bids if each bid is at least as high (a cut-down percentage) as the bids for the previous rounds.
\[ \forall \alpha \in Traces(S) \forall \tau, N, CD, CD' \]
\[ stateS(M, t, output(S)) \Leftrightarrow cutdown(CD, N) \]
\[ \Rightarrow \exists ! \tau t, CD, CD' \]

C6. Finite termination of negotiation by CA
A Customer Agent CA satisfies finite termination of negotiation by if a time point exists such that CA does not negotiate anymore after this time point.
\[ \forall \alpha \in Traces(S) \exists \tau t, CD, N \]
\[ stateS(M, t, output(S)) \Leftrightarrow cutdown(CD, N) \]

A successfulness property of a Customer Agent could be defined on the basis of some balance between discomfort and financial gains.
5.2.3 Properties of the External World
The External World satisfies information provision effectiveness if it provides information about the predicted use of energy, the maximum energy level allocated to each Customer Agent, and the maximal overuse of the Utility Agent. The External World satisfies static world if the information provided by the external world does not change during a negotiation process.

5.3 Proving Properties
To structure proofs of properties, the compositional structure of the system is followed. For the level of the whole system, system properties are proved from agent properties, which are defined at one process abstraction level lower.

5.3.1 Proofs of the System Properties
Property S4 is an assumption on the system, which is used in the proofs of other properties. The other top level properties can be proven from the agent properties in a relatively simple manner. For example, by definition monotonicity of negotiation (S1) can be proven from the properties monotonicity of announcement (U7) and monotonicity of bids (C5) for all Customer Agents. Also S2 (termination) can be proven directly from U10 and C6, and S3 (rationality) immediately follows from U9 and C4.

5.3.2 Proofs of Agent Properties
Less trivial relationships can be found between agent properties. As an example, the termination property for the Utility Agent (U10) can be proven from the properties U1, U3, and U6. The termination property of a Customer Agent depends on the Utility Agent, since the Customer Agents are reactive: the proof of C6 makes use of C3, and the Utility Agent properties U1 and U3, and the assumption that the communication between UA and CA functions properly (CA should not receive round information that was not generated by UA). In the proofs of an agent property, also properties of sub-components of the agent can be used: the proof can be made at one process abstraction level lower. This will be discussed for the Utility Agent in Section 6.

6. Verification within the Utility Agent
To illustrate the next level in the compositional verification process, in this section it is discussed how properties of the Utility Agent can be related to properties of components within the Utility Agent. First some of the properties of the components Agent Interaction Management and Determine Balance are defined.

6.1 Properties of Components within UA
Properties are defined for the components Agent Interaction Management (AIM), Determine Balance (DB), Cooperation Management (CM), and Own Process Control (OPC) of the Utility Agent (see Figure 1).

6.1.1 Properties of AIM
The following two properties express that the component Agent Interaction Management (1) distributes the relevant information from incoming communication, and (2) generates outgoing communication if required.

AIM1. Cut-down provision effectiveness
The component Agent Interaction Management satisfies cut-down provision effectiveness if AIM is effective in the analysis of incoming communication: the cut-down information received by AIM of the form received(cutdown from(CD, CA, N)) is interpreted and translated into cut-down information required by other components of the form offered bid(cutdown(CD, CA, N)) and made available in AIM’s output interface.

AIM2. Communication generation effectiveness
The component Agent Interaction Management satisfies communication generation effectiveness if AIM is effective in generation of outgoing communication on the basis of the analysis of input information received from other components of the form next communication(announcement(CD, R, N)) which is made available in statements own communication(announcement(CD, R, N)).

6.1.2 Properties of Determine Balance
The following two properties express that the component Determine Balance calculates predictions in a reasonable manner.

DB1. Overuse prediction generation effectiveness
The component Determine Balance satisfies overuse prediction generation effectiveness if the predicted overuse is determined if and when normal capacity, predicted use and cut-downs are known.

DB2. Overuse prediction monotonicity
The component Determine Balance satisfies overuse prediction monotonicity if the following holds: if based on received cut-downs CDca for each Customer Agent CA, a predicted overuse u is generated by DB, and based on received cut-downs CDca for each Customer Agent CA, a predicted overuse u' is generated by DB, then CDca ≤ CDca for all CA implies u' ≤ u.
DB3. Overuse prediction decrease effectiveness

The component Determine Balance satisfies overuse prediction decrease effectiveness if the following holds: cut-down values exist such that, if the Utility Agent receives them, the predicted overuse will be at most the maximal overuse. Formally, a collection of numbers $CD_{CA}$ for each Customer Agent $CA$ exists such that:

$$\forall t',\; \forall U \leq \text{max\_overuse}, \; \text{predicted\_overuse}(U, N)$$

6.1.3 Properties of Cooperation Management

Cooperation Management fulfills a number of properties, for example on properly generation announcements: announcement generation effectiveness, announcement uniqueness, and announcement generation groundedness. These are defined similarly to the corresponding properties of the Utility Agent. In this paper only the property that guarantees that new rounds are initiated is explicitly stated.

CM1. Round generation effectiveness

The component Determine Balance satisfies round generation effectiveness if CM determines the value of the next round and makes this information available to other components in its output interface.

$$\forall \text{Traces}(CM) \forall t, N$$

$$\text{state}_{CM}(N, t, \text{input}(CM)) \Rightarrow \text{round}(N)$$

6.1.4 Properties of Own Process Control

One of the properties of the component Own Process Control guarantees that decisions about continuation of a negotiation process are made:

OPC1. New announce decision effectiveness

If the predicted overuse is still more than the maximum overuse, then a new announcement is warranted.

$$\forall \text{Traces}(OPC) \forall t, N, U$$

$$\text{state}_{OPC}(N, t, \text{input}(OPC)) \Rightarrow \text{current\_negotiation\_state}(\text{predicted\_overuse}(U, N))$$

$$\Rightarrow \exists \text{Tr}\Rightarrow \text{state}_{OPC}(N, t, \text{output}(OPC)) \rightarrow \text{new\_announce}$$

6.2 Proofs within the Utility Agent

To verify the UA property U2 (negotiation round generation effectiveness), a number of properties of sub-components, are of importance, and also the interaction between the components through the information links (the arrows in Figure 1) should function properly. The following gives a brief sketch of the proof of the UA property negotiation round generation effectiveness.

The round number itself is determined by CM; to guarantee this, CM needs to satisfy the property of round generation effectiveness (CM1). This round value is transferred to the component AIM. The component AIM must fulfill the property of communication generation effectiveness (AIM2) to enable this value to be placed in the Utility Agent's output interface, once the relevant link has been activated. Activation of the link to the Utility Agent's output interface depends (via task control) on whether the component OPC derives the need for a new announcement. To guarantee this, the property new announce decision effectiveness (OPC1), is needed.

Based on the properties mentioned, the proof runs as follows. Whenever the component AIM has received all the cut-downs for the current round, the link from AIM to DB1 is activated. This component then derives the current predicted overuse (assuming predicted use, normal capacity and round are known). It can be assumed that the overuse for this round is above $\text{max\_overuse}$ (otherwise the conditions for U2 are not satisfied). The component OPC is then activated (by task control) and, given property OPC1 (new announce decision effectiveness) this component will derive the atom new_announce. Then Cooperation Management is activated and given property CM1, this component will derive a new round. Given property AIM2, this new round information will be available on the output interface of AIM; the link outgoing communications transfers the desired result: round(N+1) at the output of UA. This proves Utility Agent property U2.

7. Discussion

To come to a clearer understanding of strengths and weaknesses of a compositional approach to verification it is important to address real world problems where size and/or complexity are characteristic. The load balancing problem of electricity use, as addressed in this paper, belongs to the class of real world problems. This paper focuses on one-to-many negotiation between a Utility Agent and its Customer Agents, using a (monotonic) negotiation strategy based on announcing reward tables.

The compositional verification method used in this paper is part of the compositional development method for multi-agent systems DESIRE, based on compositionality of processes and knowledge at different levels of abstraction, but can also be useful to other compositional approaches. 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) the complexity of the verification process is managed by compositionality, and (2) the proofs of properties of components that are reused can be 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. The second advantage entails: if a modified component satisfies the same properties as the one it replaces, 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 system. This increases the value for a documented library of reusable generic and instantiated components.

Also due to the compositional nature of the verification method, a distributed approach to verification is facilitated: several persons can work on the verification of the same system at the same time. It is only necessary to know or to agree on the properties of these sub-components with interfaces in common.

A main difference in comparison to [9] is that our approach exploits compositionality. An advantage of their...
approach is that it uses a temporal belief logic. A first step to extend our approach a compositional variant of temporal logic can be found in [7]. A main difference to the work described in [3] and [8] 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 address both the embedding of verification proofs in a suitable proof system for temporal logic (for some first results, see [7]), and 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 in a suitable proof system.

References


