Compositional Verification of Knowledge-Based Systems: a Case Study for Diagnostic Reasoning

Frank Cornelissen, 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/~{frankc,jonker,treur}
Email: {frankc,jonker,treur}@cs.vu.nl

Abstract In this paper a compositional verification method for models of knowledge-based systems is introduced. Required properties of the system are formally verified by deriving them from assumptions that themselves are properties of sub-components, which in their turn may be derived from assumptions on sub-sub-components, and so on. The method is based on properties that are formalised in terms of temporal semantics; both static and dynamic properties are covered. The compositional verification method imposes structure on the verification process. By the possibility to focus at one level of abstraction (information and process hiding), compositional verification provides transparency and limits the complexity per level. Since verification proofs are structured in a compositional manner, they can be reused in case of modification of the system. The method is illustrated for a generic model for diagnostic reasoning.

Keywords Compositional verification, knowledge-based systems, diagnostic reasoning model, formal compositional modelling.

1. Introduction

When designing complex knowledge-based systems, it is often hard to guarantee that the specification of a system that has been designed actually fulfills the needs, i.e., whether it satisfies the design requirements. Especially for critical applications, for example in aerospace 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.

In this paper, in Section 3, a structured verification method for complex knowledge-based systems is introduced, called *compositional verification*. Roughly spoken, the requirements of the whole system are formally verified by deriving them from assumptions that themselves are properties of sub-components, which in their turn may be derived from assumptions on sub-sub-components, and so on. This process ends when primitive components are reached: components that are not composed, but specified by means of a knowledge base (or any other means).

The method introduced here is illustrated for a generic task model for diagnostic reasoning. For this example task model, requirements are formulated (both the required static and dynamic properties), and a compositional system specification is introduced in Section 4. The compositional specification is based on a task composition that specifies how the main task is composed of the tasks hypothesis determination and hypothesis validation, and how the sub-task hypothesis validation is composed of the tasks observation determination, observation execution and hypothesis evaluation. The compositional specification itself is expressed in the modelling framework DESIRE, briefly described in Section 2. The application of the compositional verification method to the example task model is presented in Section 5 (top level of the composition), Section 6 (lower level), and Section 7 (primitive components).

2. Compositional modelling of knowledge-based systems

The example task model described in this paper is specified within the compositional modelling framework DESIRE for knowledge-based systems and multi-agent systems (framework for DEsign and Specification of Interacting REasoning components; cf. (Brazier, Treur, Wijngaards and Willems, 1995; Brazier, Dunin-Keplicz, Jennings, Treur, 1995)). A number of generic models for agents and tasks have been developed and used for a number of applications. The architectures upon which compositional specifications are based are the result of analysis of the tasks performed. Task compositions include specifications of interaction between tasks at each level within a task composition. Models specified within DESIRE define the structure of *compositional architectures.*: Components in a compositional architecture are directly related to tasks in a task composition. The hierarchical structures of tasks, interaction and knowledge are fully preserved within compositional architectures. Below the formal compositional framework for modelling multi-agent tasks DESIRE is introduced, in which the following aspects are modelled and specified: (1) a task composition, (2) information exchange, (3) sequencing of tasks, (4) task delegation, (5) knowledge structures.

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 to an operational system is straightforward; the software environment includes implementation generators with which formal specifications can be translated into executable code. DESIRE has been successfully applied to design both single agent and multi-agent knowledge-based systems.

3. Compositional Verification

The purpose of verification is to prove that, under a certain set of assumptions, a system will adhere 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 fulfill.

3.1 The Compositional Verification Method

A compositional system can be viewed at different levels of abstraction. Viewed from the top level, denoted by L_0, the complete system is one component D, with interfaces, whereas internal information and processes are hidden (information and process hiding). At the next lower level of abstraction, the top level component D can be viewed as a composition of sub-components,

information links, and task control. The compositional verification method takes into account this compositional structure. The primitive reasoning components can be verified using more traditional verification methods such as described in (Treur and Willems, 1994; Leemans, Treur and Willems, 1993) . Verification of a composed component is done using properties of the sub-components it embeds and the task control knowledge. This introduces a form of compositionality in the verification process: the proof that a certain component adheres to a set of properties depends on the (assumed) properties of its sub-components. 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. These properties have hierarchical logical relations in the sense that at each level a property is logically implied by (a conjunction of) the lower level properties that relate to it in the hierarchy (see Figure 1).

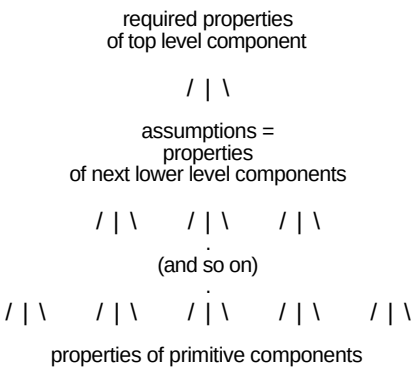

Figure 1 Hierarchical relations between properties in compositional verification

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 higher level properties from lower level 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 method supports reuse of verification.

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 top-down procedure for verification is followed:
1. Determine which properties are of interest (for the higher level).
2. Determine assumptions (at the lower level) that guarantee these properties.
3. Prove the properties on the basis of these assumptions.

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 1993). For primitive non-knowledge-based components, such as data bases, or neural networks, or optimization 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 are 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 (as in Figure 1).

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 fulfills this requirement.
- In principle, a similar, bottom-up procedure, can be formulated as well.
- For any set of assumptions obtained in A., if it is required that it does not contain superfluous elements, for each assumption in the set an example may be constructed in which the assumption does not hold, whereas the other assumptions in the set hold and one or more of the properties fail. If for one of the assumptions no example is possible, then try to eliminate it.

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 and time steps in which transitions from one state to the other occur. In this sub-section 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). 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 C, $state_C(\mathcal{M}, t, input(C'))$ denotes the information state of the input interface of component C' at time point t of the component C, where C' is either C or a sub-component of C. Analogously, $state_C(\mathcal{M}, t, output(C'))$, denotes the information state of the output interface of component C' at time point t of the component C. Given a trace $\mathcal{M}$ of component C, the task control information state of component C' at time point t of the component C is denoted by $state_C(\mathcal{M}, t, tc(C'))$, where C' is either C or a sub-component of C.

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 task model for diagnostic reasoning

The example model described in this section is based on the generic model for diagnostic reasoning analysed in (Treur, 1993). Diagnostic reasoning is the analysis of the cause of a disturbed situation. In most of these situations not all relevant observational facts are known in advance. The process of acquisition of additional (observation) information is an essential part of

most diagnostic processes (Treur, 1993). Therefore, dynamics play an important role in diagnosis. In general diagnostic reasoning consists of a number of sub-tasks such as the determination of hypothesis, the choice of applicable tests, the performance of tests and the interpretation of the test results. Strategic information such as the suitibility of a test, likeliness of a hypothesis being true and the cost and effect of a test play an important role. In this section the model of diagnosis to be verified is described. First the task hierarchy is given, and each component is described, followed by the interaction between the components.

4.1 Task Composition

The task composition of the system is given in Figure 2. The task Hypothesis Determination generates hypotheses that are validated by the task Hypothesis Validation.

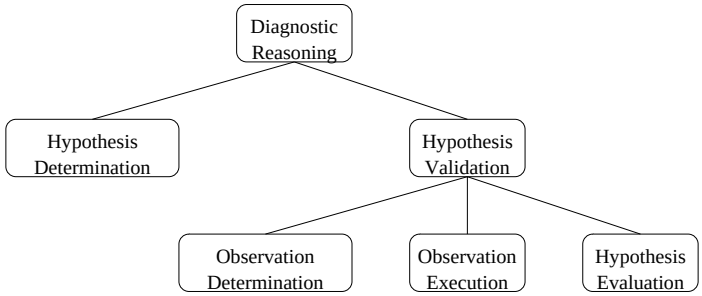

Figure 2 Task composition of the diagnostic reasoning task

These two tasks are described in the subsequent sections. In this section HYPS stands for the set of all (possible) hypotheses and OBS for the set of all (possible) observations.

4.2 Hypothesis Determination

The task Hypothesis Determination suggests hypotheses to be validated. This is done using information on which hypothesis have been rejected so far. The input and output interfaces are defined by

input atoms rejected(h), confirmed(h) ; h ∈ HYPS
output atoms focus(h) ; h ∈ HYPS

Whenever an hypothesis has been rejected or confirmed, it should not be suggested as a focus. The selection of hypotheses for the focus could for example be based on the frequency at which the hypotheses occur. This component should select one or more hypotheses whenever not all hypotheses have been rejected. This task is specified as a primitive component in the example.

4.3 Hypothesis Validation

The main task of Hypothesis Validation is to determine whether the hypotheses of a given focus set are valid. In addition to that it keeps track of hypotheses that have already been validated. The interface and internal atoms for Hypothesis Validation are the following:

input atoms focus(h) ; h ∈ HYPS
internal atoms observed(o); o ∈ OBS
output atoms rejected(h), confirmed(h); h ∈ HYPS

The input is obtained from Hypothesis Determination.

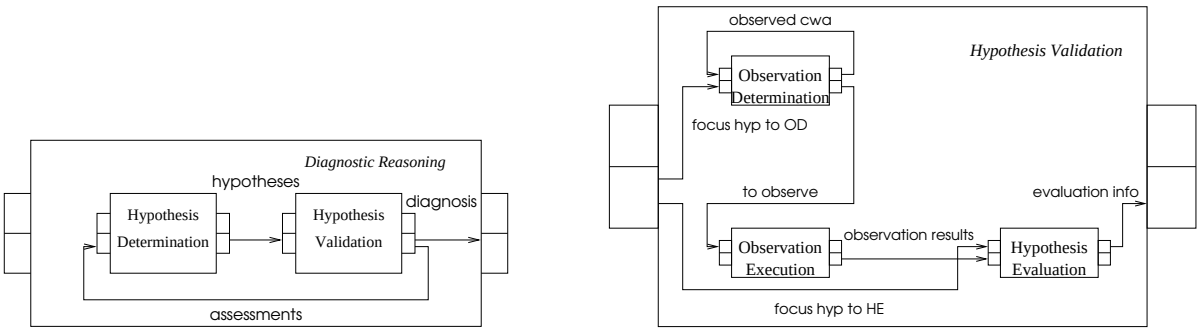

Figure 3 Composition and Information Exchange at two levels

The task Hypothesis Validation is composed of three primitive tasks. Each of these are described shortly in the following paragraphs.

Observation Determination
To validate a hypotheses, observations have to be performed. These observations are selected by the sub-task Observation Determination . The knowledge required for this selection might include cost of doing observations, reliability, and so on. The information required by this task are the hypotheses that are in focus and observations that have already been performed. The interface of this task is

input atoms	focus(h) , observed(o); $h \in HYPS$, $o \in OBS$
output atoms	to_observe(o); $o \in OBS$

Observation Execution
The Observations are made in the sub-task Observation Execution. The information this task requires are the observations it needs to perform. The output consists of the results of those observations. The interface of this component is as follows:

input atoms	target(o) ; $o \in OBS$
output atoms	o ; $o \in OBS$

Hypothesis Evaluation
Given observation results, the task Hypothesis Evaluation derives conclusions about which hypotheses are true. This task has the same level as Observation Execution since it uses the observations made there to derive truth values of hypothesis in focus by means of anti-causal knowledge. The interface of this task is

input atoms	o ; $o \in OBS$
output atoms	h ; $h \in HYPS$

In Figure 3 the interaction within the whole system S is shown. The link hypotheses transfers the hypotheses determined in Hypothesis Determination to Hypothesis Validation. The link assessments transfers the results from the evaluation in Hypothesis Evaluation to Hypothesis Determination so this component knows which hypothesis are rejected. The last link, diagnosis transfers the diagnosis determined by the system to the output interface of the main component.

5. Verification of the system S as a whole

First the manner in which time points are attached to the reasoning process is discussed.

5.1 Time points

For the verification of this system we need to introduce time points to reason about the dynamics of this system as explained in Section 3. For the component S time points are defined as:
• Time point 1 corresponds to the termination time of the first activation of component Hypothesis Validation

• Time point t + 1 is after the subsequent activations of Hypothesis Determination and Hypothesis Validation have been finished.

For the component Hypothesis Validition the time points are defined as:

• Time point 0 corresponds to no activation of the component.
• Time point t + 1 is after Hypothesis Evaluation has been active, or Hypothesis Validation terminates.

The time steps within both components are illustrated by a sample trace of the system in Figure 4.

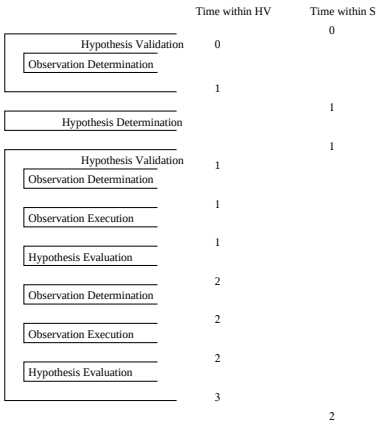

Figure 4 Trace of the diagnostic system with annotated time points

5.2 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 diagnostic reasoning system, it is expected that S produces output of the form confirmed(h) and / or rejected(h) for some hypotheses h. A first requirement is that output generated by the system in terms of assessments of hypotheses is correct, i.e., if the system derives that a hypothesis has been confirmed, it is true in the world situation, and if the system derives it is rejected, it is false in the world situation. Let the current world state be denoted by M. The following property relates the output of the system to the current world state.

Assessment correctness of S
The system S is called *assessment correct* if:

$$(\forall \mathfrak{M} \in \mathrm{Traces}(S)\ \forall t\ \forall h \quad \mathrm{state}_S(\mathfrak{M}, t, \mathrm{output}(S)) \models \mathrm{confirmed}(h) \quad \Rightarrow M \models h) \wedge$$
$$(\forall \mathfrak{M} \in \mathrm{Traces}(S)\ \forall t\ \forall h \quad \mathrm{state}_S(\mathfrak{M}, t, \mathrm{output}(S)) \models \mathrm{rejected}(h) \quad \Rightarrow M \models \neg h)$$

Next, the system is required to be effective in generating assessments: during the process it should derive at least some positive assessment output, except in case all hypotheses are false; then the system should derive that all hypotheses are false:

Assessment effectiveness of S
The system S is called *assessment effective* if:

$$(\exists h\ M \models h \Rightarrow \forall \mathfrak{M} \in \mathrm{Traces}(S)\ \exists t\ \exists h'\ \mathrm{state}_S(\mathfrak{M}, t, \mathrm{output}(S)) \models \mathrm{confirmed}(h')) \wedge$$
$$(\forall h\ M \models \neg h \Rightarrow \forall \mathfrak{M} \in \mathrm{Traces}(S)\ \exists t\ \forall h\ \mathrm{state}_S(\mathfrak{M}, t, \mathrm{output}(S)) \models \mathrm{rejected}(h))$$

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

Assessment conservativity of S
The system S is called *assessment conservative* if:

a) $\forall \mathfrak{M} \in \mathrm{Traces}(S)\ \forall t\ \forall h\ [\ \mathrm{state}_S(\mathfrak{M}, t, \mathrm{output}(S)) \models \mathrm{confirmed}(h) \Rightarrow \mathrm{state}_S(\mathfrak{M}, t+1, \mathrm{output}(S)) \models \mathrm{confirmed}(h)\ ]$

b) $\forall \mathfrak{M} \in \mathrm{Traces}(S)\ \forall t\ \forall h\ [\ \mathrm{state}_S(\mathfrak{M}, t, \mathrm{output}(S)) \models \mathrm{rejected}(h) \Rightarrow \mathrm{state}_S(\mathfrak{M}, t+1, \mathrm{output}(S)) \models \mathrm{rejected}(h)\ ]$

Also termination of the system may be relevant:

Termination of S
The system S *always terminates* if: $\forall \mathfrak{M} \in \mathrm{Traces}(S)\ \exists t\ \mathrm{state}_S(\mathfrak{M}, t, \mathrm{tc}(S)) \models \mathrm{stop}$

5.3 Assumptions needed to prove the properties of the top level

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.

5.3.1 Assumptions on Hypothesis Validation

Some assumptions are quite straightforward. For example, assessment correctness simply inherits upward from Hypothesis Validation:

Assessment correctness of HV

The component hypothesis_validation is called *assessment correct* if:

a) $(\forall \mathcal{M} \in \text{Traces(HV)}\ \forall t\ \forall h\ \text{state}_{HV}(\mathcal{M}, t, \text{output(HV)}) \models \text{confirmed(h)} \Rightarrow M \models h)$

b) $(\forall \mathcal{M} \in \text{Traces(HV)}\ \forall t\ \forall h\ \text{state}_{HV}(\mathcal{M}, t, \text{output(HV)}) \models \text{rejected(h)} \Rightarrow M \models \neg h)$

Similarly, for assessment conservation:

Assessment conservativity of HV

The component hypothesis_validation is called *assessment conservative* if:

a) $\forall \mathcal{M} \in \text{Traces(HV)}\ \forall t\ \forall h\ \text{state}_{HV}(\mathcal{M}, t, \text{output(HV)}) \models \text{rejected(h)} \Rightarrow \text{state}_{HV}(\mathcal{M}, t+1, \text{output(HV)}) \models \text{rejected(h)}$

b) $\forall \mathcal{M} \in \text{Traces(HV)}\ \forall t\ \forall h\ \text{state}_{HV}(\mathcal{M}, t, \text{output(HV)}) \models \text{confirmed(h)} \Rightarrow \text{state}_{HV}(\mathcal{M}, t+1, \text{output(HV)}) \models \text{confirmed(h)}$

For effectiveness, the relationship is not one-to-one as in the case of correctness and conservativity. However, also in this case, at least one (among others) of the required assumptions on Hypothesis Validation is that it is effective in generating assessments, as long as focus hypotheses are provided to it. Here, and in the sequel the abbreviation assessed(h) is used instead of confirmed(h) $\vee$ rejected(h).

Assessment effectiveness of HV

The component hypothesis_validation is called *assessment effective* if:

$(\forall \mathcal{M} \in \text{Traces(HV)}\ \forall t\ [\ \exists h\ \text{state}_{HV}(\mathcal{M}, t, \text{input(HV)}) \models \text{focus(h)}\ ] \Rightarrow$
$[\ \exists h'\ \text{state}_{HV}(\mathcal{M}, t, \text{input(HV)}) \models \text{focus(h')} \wedge \text{state}_{HV}(\mathcal{M}, t, \text{output(HV)}) \models \text{assessed(h')}\ ]\ )$

5.3.2 Assumptions on Hypothesis Determination

For the component Hypothesis Determination the assumption is made that it is efficient and effective in generating focus hypotheses. Focus efficiency means that no hypotheses are chosen in focus that already have been assessed.

Focus efficiency of HD

The component hypothesis_determination is called *focus efficient* if:

$\forall \mathcal{M} \in \text{Traces(HD)}\ \forall t\ \forall h\ [\ \text{state}_{HD}(\mathcal{M}, t, \text{input(HD)}) \models \text{assessed(h)} \Rightarrow \text{state}_{HD}(\mathcal{M}, t, \text{output(HD)}) \not\models \text{focus(h)}\ ]$

Focus effectiveness means that as long as not all hypotheses have been assessed, there will be generated focus hypotheses.

Focus effectiveness of HD

The component hypothesis_determination is called *focus effective* if:

$\forall \mathcal{M} \in \text{Traces(HD)}\ \forall t \quad [\ \exists h\ \text{state}_{HD}(\mathcal{M}, t, \text{input(HD)}) \not\models \text{assessed(h)}\ ] \Rightarrow [\ \exists h'\ \text{state}_{HD}(\mathcal{M}, t, \text{output(HD)}) \models \text{focus(h')}\ ]$

5.3.3 Domain assumptions

The properties at the top level also need assumptions on the (domain) ontology and knowledge to be used by the task model. These are the assumptions of the type considered in (Fensel, 1995; Fensel and Benjamins, 1996).

Finite number of hypotheses: The number of hypotheses is finite.

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

5.4 Proofs of the properties of the top level

For reasons of space limitation, the proofs have been omitted in this paper; see however the longer version (Cornelissen, Jonker and Treur, 1997). In Figure 5 the logical connections between the properties at different levels are depicted. An important notion used in the proofs is the notion of progression:

Definition (progression of S)

Given a trace $\mathcal{M} \in \text{Traces(S)}$ and a timepoint t, system S shows progression from time point t to time point t+1 if $\exists h\ [\ \text{state}_S(\mathcal{M}, t, \text{output(S)}) \not\models \text{assessed(h)} \wedge \text{state}_S(\mathcal{M}, t+1, \text{output(S)}) \models \text{assessed(h)}\ ]$

At each step that the system shows progression, due to assessment conservativity, the set of assessed hypotheses becomes strictly larger. The proofs follow the pattern that the assumptions guarantee that as long as not all hypotheses have been assessed the system will show progression. If the number of hypotheses is finite, say N, then within at most N time steps the set of assessed hypotheses will become the set of all hypotheses, and the system terminates.

It can be noted that for a static world the property of assessment correctness implies assessment conservatism, so in the graph of Figure 5 more logical relationships can be drawn. However, to avoid a complicated graph we did not attempt to give a complete account of all possible logical relationships in Figure 5.

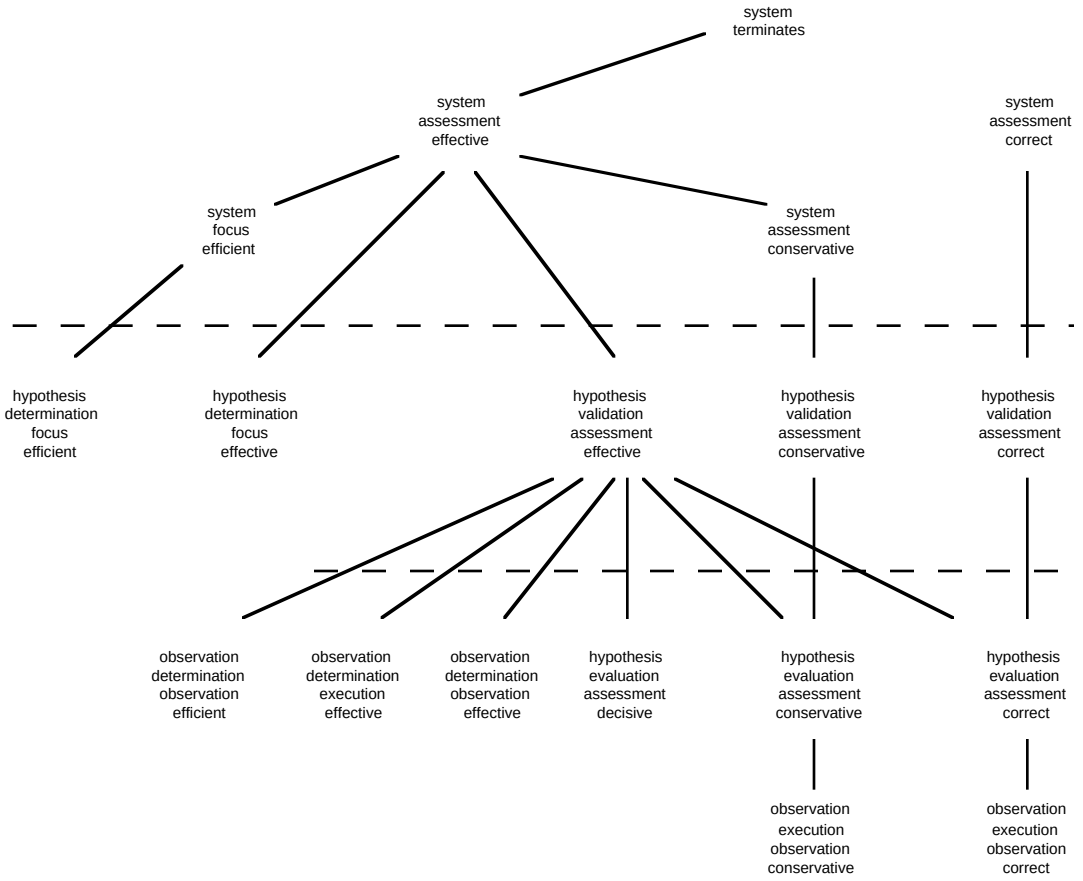

Figure 5 Logical relations between properties at different levels: the diagnostic reasoning case

6. Assumptions to prove the properties of Hypothesis Validation

The properties of Hypothesis Validation needed to prove the properties of the top level of the system were discussed in Section 5.3 (see Figure 5). The assumed properties of the sub-components of Hypothesis Validation, needed to prove these properties can also be found in Figure 5, at the lower level. For shortness, these properties are only explained informally; for formal definitions, see the full report.

The required properties of *Observation Determination* are:

Observation efficiency of OD: No observations are generated that already were performed.

Observation effectiveness of OD: If there exists at least one hypothesis in focus, and not all observations have been performed, then at least one observation is generated.

Execution effectiveness of OD: Every generated observation is performed

The required properties of *Observation Execution* are:

Observation conservativity of OE: Once an observation result has been obtained, it will persist.

Observation correctness of OE: Every observation result that is obtained is true in the world situation.

The required properties of *Hypothesis Evaluation* are:

Assessment conservativity of HE: Once an hypothesis assessment has been derived, it will persist.

Assessment decisiveness of HE: If for all possible observations, observation results have been input, then for every hypothesis an assessment can de derived.

Assessment correctness of HE: If a hypothesis is derived, then it is true in the world situation; if the negation of a hypothesis is derived, then the hypothesis is false in the world situation.

In addition to the *domain assumptions* mentioned in Section 5.3.3, the following are needed:

Empirically foundedness: The hypotheses 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 hypotheses; see (Treur and Willems, 1994).

Finite number of observations: The number of observations needs to be finite because (in the worst case) the system should be able to do all (relevant) observations to assess all hypotheses.

7. Verification of the properties of the primitive components

In Sections 5 and 6 verification of the generic task model was described, based on assumed properties of the primitive components. If the model is to be used, instances are required for the primitive components (e.g., containing domain knowledge), and for these instances it has to be verified whether they satisfy the required properties.

The instances of primitive components can be verified making use of the more standard methods described in (Treur and Willems, 1994). For example, the component Hypothesis Determination should satisfy focus efficiency and focus effectiveness. Actually, these properties reduce to the following static properties of Hypothesis Determination:

For any input model M of HD it holds

$$\forall h\, [\, M \models \text{assessed}(h) \Rightarrow M \not\vdash_{HD} \text{focus}(h)\,] \wedge [\, \exists h\, M \not\models \text{assessed}(h) \Rightarrow \exists h'\ M \vdash_{HD} \text{focus}(h')\,]$$

This type of static properties can be verified automatically for a given knowledge base, by tools as described in (Leemans, Treur and Willems, 1993). In a similar manner the properties observation effectiveness and observation efficiency of the component Observation determination reduce to static properties.

Correctness of Observation Execution and of Hypothesis Evaluation reduce to the (static) property of soundness defined in (Treur and Willems, 1994; Leemans, Treur and Willems, 1993). The property of assessment decisiveness of Hypothesis Evaluation reduces to the static notion decisiveness, which in its turn depends on the domain assumption of empirically foundedness (see Theorem 6.3 of (Treur and Willems, 1994)). The property observation conservativity of the component Observation Execution depends on the static world assumption (and on correctness of observation results).

8. Conclusions

The modelling framework DESIRE is based on compositionality. The compositional verification method described in this paper fits well to DESIRE, but can also be useful to any other compositional modelling approach. The advantage of a compositional approach to modelling is to be able to reuse components and task models easily; the compositional verification method extends this to the reuse of proofs for properties of components that are reused. For example, the diagnostic reasoning system described in this paper could be modified by changing the component Hypothesis Validation. If this changed component has the same properties as the

current, the proof of the top level properties can be reused to show that the new system has the same properties as the original. This has high value for a library of generic task models, where the domain knowledge is not yet known. The verification of generic task models forces one to find the assumptions under which the generic task model is applicable for the considered domain, as is also discussed in (Fensel, 1995, Fensel and Benjamins, 1996). A library of reusable components and task models will be set up, consisting of both specifications of the components and models, and their design rationale. Although the precise contents of the design rationale is currently under study, at least the properties of the components and their logical relations (e.g., as represented in Figure 5) are to be part of it.

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 method proposed in this paper is useful for compositional knowledge-based systems as well as compositional multi-agent systems.

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. Compared to (Fensel and Benjamins, 1996), where also properties of diagnosis are identified, in the current paper the properties are formalised in formal semantical terms (they are expressed in terms of temporal formal semantics), whereas in the paper as referred the properties are not (yet) formalised. For example, the formalisation of the assumption 'heuristic search knowledge' (see Table 3 in Section 4 in the paper as mentioned) in terms of the semantics of the behaviour of the system might turn out far from trivial. Especially the semantical formalisation of such dynamic properties and their logical relationships is a challenge. Furthermore, assumptions on the dynamics of hypothesis determination and the heuristic knowledge involved, as presented in our paper, have been left out of consideration. On the other hand, the value of the paper is that it gives an extensive account on various assumptions for model-based diagnosis; this was left out of consideration in our paper. Besides compositionality, a difference of our approach with (Harmelen and Teije, 1997) is that in the latter approach only static properties of diagnosis are considered, whereas in our approach also dynamic properties are covered, formalised in temporal semantics.

Previous work on verification of compositional knowledge-based systems, described in (Treur and Willems, 1995), was based on the formulation of a compositional verification principle described in (Abadi and Lamport, 1993), applied to knowledge-based systems. This principle lifts properties of the sub-components to the component in which these are embedded. If all sub-components satisfy a certain property, and they are connected in the right manner, then the component as a whole will satisfy that property. The properties that can be verified in this way are the properties such as 'functions properly'. However, for most real-world systems it gives more insight to explicit 'proper functioning' in the form of (task and domain) specific properties, as has been shown above. In this sense the current paper is a further development and refinement of the work described in (Treur and Willems, 1995).

A future continuation of this work will consider the development of tools for verification. At the moment tools exist for the verification of primitive components but not for the verification of composed components. To support the handwork of verification it would be useful to have tools to assist in the creation of the proofs. This could be done by formalizing 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 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

Dieter Fensel provided useful comments on an earlier version of this paper.

References

Abadi, M. and L. Lamport (1993). Composing Specifications, ACM Transactions on Programming Languages and Systems, Vol. 15, No. 1, 1993, pp. 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 Twelfth European Conference on Artificial Intelligence, ECAI'96, John Wiley and Sons, 1996, 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. (1995). Formal Specification of Hierarchically (De)Composed Tasks. 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, 1995, pp. 25/1-15/20.

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, 1996, pp. 15/1-15/17.

Fensel, D. (1995). Assumptions and limitatons 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, 1995.

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, 1996, 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, 1996, pp. 4/1-4/20.

Harmelen, F. van, Teije, A. ten (1997). Validation and verification of diagnostic systems based on their conceptual model. In: Proceedings of the Fourth European Symposium on the Validation and Verification of Knowledge-based Systems, EUROVAV'97, this volume, 1997.

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

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

Treur, J. (1993). Heuristic reasoning and relative incompleteness. International Journal of Approximate Reasoning, vol. 8, 1993, pp. 51-87.

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

Treur, J., and M. Willems (1995). Formal notions for verification of dynamics of knowledge-based systems. In: Proceedings of the Third European Symposium on the Validation and Verification of Knowledge-based Systems, EUROVAV'95, 1995, pp. 189-199.

Treur, J. and Th. Wetter (1993). Formal Specification of Complex Reasoning Systems. Ellis Horwood, 1993.