Formal Notions for Verification of Dynamics of Knowledge-Based Systems[*]

Jan Treur

Mark Willems

Vrije Universiteit Amsterdam
Department of Mathematics and Computer Science
Artificial Intelligence Group
De Boelelaan 1081a, 1081 HV Amsterdam, The Netherlands
tel: +31-20-548 5326/5520
fax: +31-20-6427705
e-mail: `treur,willems@cs.vu.nl`

Abstract

In the literature on validation or verification of knowledge-based systems often a limitation to systems with one knowledge base is chosen, and the focus is on the static properties of these systems. In practice often knowledge-based systems are developed that have some form of compositional structure. Also the reasoning in this structure is controlled, often according to the compositional structure. The task-layer of the KADS-approaches and the task control structure in DESIRE show this. A number of formal specification languages has been developed for this type of knowledge-based systems. Although there are differences, a number of common principles already has been reached (for an overview and comparison, see [23]). Therefore it is natural to attempt to extend the work on verification and validation to systems defined by such languages. Such an extension to compositional systems necessarily leads to the consideration of their temporal behaviour and the dynamic properties. In this paper the logical framework previously developed in [24] is extended for dynamic properties of compositional knowledge-based systems.

---

1.   Introduction

Our ideas on verification, or validation must be viewed in the light of compositional architectures of knowledge based systems. These are systems that are composed from a number of components, each with their own knowledge base and associated inference relation. At each level of composition verification of a component is seen as a comparison between the actual behavior, for instance the set of deductive closures of the knowledge base for all admissible inputs, with respect to a description of the intended behavior, i.e. a (theoretically complete) description of the required output for all admissible inputs.

Moreover, we work at a semantic level by considering behavior as functionality descriptions, which are independent of any particular language of the knowledge base. This becomes apparant in our use of partial models, i.e. three-valued truth assignments to (ground) atoms that are collected in a signature.

In this paper we will extend the formal notions as introduced in [24] (there restricted only to knowledge bases from a static viewpoint). Considering these notions for compositional architectures (more knowledge bases working together) in a natural way challenges one to take the temporal behavior of such systems into account. In this way we introduce formal notions that provide a founcation for the verification of compositional and dynamic aspects of a knowledge based system.

2.   Properties for Verification

First some terminology that will clarify our definitions.

Definition 2.1.(Signature, Model, Input/Output, World Description)
A *signature*   **Ont**   is a sequence of symbols   (for instance sorts, constants, functions, predicates) that is used to generates the set of *(closed) atoms* **At(Ont)**, wherein the *input atoms* **InAt(Ont)**, and *output atoms* **OutAt(Ont)** are distinguished. A *partial model*   **M**   with respect to a signature **Ont** is a truth-assignment to the (closed) atoms in a signature **Ont**, i.e., a mapping $\mathbf{M : At(Ont)} \rightarrow \{0, 1, u\}$. Associated with any (partial) model **M** we can define the input model **In(M)** that copies the input truth-assignment of **M**, but assigns **u** to all other atoms. Similarly the output model **Out(M)** copies the output truth-assignment.

When a proposition **p** *is true / holds* (or not) in some (partial) model **M** following the strong Kleene semantics [2], we write **M** $|=^+$ **p** or **M** $|=^-$ **p**. Moreover, if no confusion is possible we use the more simple notation **M** $|=$ **p** and **M** $|=$ ¬**p**.

An actual situation is described as a *situation model* (or *complete model*) **N** that does not contain any truth-assignment to **u**. A *domain* or *world description* is a set **W** of complete situation models. A model **N** is a *refinement* of a model **M**, denoted by **M** $\leq$ **N**, if for all atoms $\mathbf{a} \in$ **At(Ont)** it holds **M(a)** $\leq$ **N(a).** Then associated with a set of models **V** we define

the set $\mathbf{P(V)}$ of partial models that can be refined to a model in $\mathbf{V}$. We also define the set of all input or output models by $\mathbf{In(P(W))}$ and $\mathbf{Out(P(W))}$. Clearly for any $\mathbf{M}$ in $\mathbf{P(W)}$ we have $\mathbf{In(M)} \leq \mathbf{M}$ and $\mathbf{Out(M)} \leq \mathbf{M}$ as elements of $\mathbf{P(W)}$.

Partial model can used to describe the *information state* of a reasoning module. The set of information states for a system $\mathbf{S}$ will be denoted by $\mathbf{IS(S)}$, and the set of input states by $\mathbf{IS_{in}(S)}$. Given a set of information states $\mathbf{V}$ in $\mathbf{IS(S)}$ the inference relation $\mathbf{M} \vdash_S \mathbf{p}$ is defined for model $\mathbf{M} \in \mathbf{V}$ and proposition $\mathbf{p}$. For consistent systems the notation $\mathbf{dc_S(M)}$ will be used to describe the deductive closure with $\mathbf{S}$ from a model $\mathbf{M}$.

Our view of reasoning is that it is a relation between partial models, that need to be refined: unknown atoms are given a different truth-value. This explains our notation of inference as a relation between a partial model and a proposition. Moreover, it is useful to have an inference which keeps previously derived truth-values (conservativity) and which infers more from larger input (monotonicity).

A knowledge based system can be viewed statically by considering the closure of atoms that can be deduced by the system from a (partial) input model. In the case of a single knowledge base the notation $\mathbf{M} \vdash_{KB} \mathbf{h}$ is allright, but when a system $\mathbf{S}$ contains more sub-knowledge bases the notation $\mathbf{M} \vdash_S \mathbf{h}$ might be considered to be a little strange. Therefore $\mathbf{dc_S(M)}$ seems more natural when considering composition. It assumes that a unique (partial) model can be associated by the function $\mathbf{dc_S}$ to a model $\mathbf{M}$, so the system should be consistent.

Many mechanisms have been proposed for the automated verification of rule-based systems, like pairwise rule comparison [16], [17], [20], the use of meta-knowledge [3], augmented transition networks [15], relational database techniques [14], and some others. Often pairwise rule comparison is used, that only check for problems involving two rules. Here, as in [18], we will check for problems involving an arbitrary number of rules by using chaining.

A question one might ask oneself in light of verification is how one should specify the desired behaviour of a reasoning module. One could imagine the expert to specify the input-output correspondence of the module by means of a table defining the functionality. This is similar to the approach in [6] where a table of atoms is given that specifies what cause leads to which effects.

Some approaches (like COVADIS [18]) use a Fact Base (FB), which is a set of particular cases. It can be built by any user to specify the problem he or she wants to submit to the system. The FB is initialised at each session. We will use situation models (instead of FB's) that represent situations of the real world. A problem with situation models (and FB's too) is, that they are chosen by the designer of the system to cover at best the variety of basic situations. Therefore, it is impossible to guarantee that a knowledge base is completely debugged.

Our idea is to consider a domain description **W**, and try to determine the standards for derivation or the intended behavior from it. With a list of the possible situation models one can define the intended statics of a system. In particular the notion of forcing is interesting for verification, see [24].

Definition 2.2

Given a domain description **W** with respect to a signature **Ont**, a partial model $\mathbf{M} \in \mathbf{P(W)}$, and an output literal $\mathbf{h} \in \mathbf{OutLit(Ont)}$. The model **M** *forces* the literal **h** within **W** if and only if **h** holds in every complete refinement $\mathbf{N} \in \mathbf{W}$ with $\mathbf{N} \geq \mathbf{M}$. We will use the notation $\models_{\mathbf{W}}$ that is defined by:

$$\mathbf{M} \models \mathbf{h} \quad \Leftrightarrow \quad \forall \mathbf{N} \in \mathbf{W} \ [\ \mathbf{N} \geq \mathbf{M} \rightarrow \mathbf{N} \models \mathbf{h}\ ]$$

The alternative functional notation is also possible here because forcing is consistent by definition. Thus we can define the semantic closure by $\mathbf{sc_W(M)} \models \mathbf{h} \Leftrightarrow \mathbf{M} \models_{\mathbf{W}} \mathbf{h}.$

The notion of forcing is one of the essences of our theory of verification. It sets a standard for the knowledge base, what are the hypotheses it should derive given some partial model. In later sections we will elaborate on this notion from a dynamic, temporal view.

It is important to realise that forcing is essentially different from the semantic consequence relation in a logical theory, $\mathbf{T} \models \mathbf{h}$. A world description is a *random* set of models not necessarily associated (yet) with a logical theory, so the notion of forcing is more general and better suited for the purposes of verification. In fact the whole idea behind verification is that we are building a theory, i.e. the knowledge base or system.

In general, for verification we want the properties for forcing and inference to be closely related, and what is natural for forcing can be enforced on inference. The main theme of our notions is that this is what verification should check: the equivalence between forcing $\models$ and inference $\vdash$-.

The properties that we have defined in [24] are listed below, each time followed by the same notion in the functional notation. Some notions can be formulated more concisely in this different notation for static systems as well.

Definition 2.4

Assume a domain description **W**, and a system **S** both with respect to a signature **Ont**. We define the following properties:

A system **S** is *weakly sound* with respect to **W** if

$$\forall \mathbf{N} \in \mathbf{W} \qquad \mathbf{dc_S(In(N))} \leq \mathbf{sc_W(In(N))}$$

A system **S** is *correct* with respect to **W** if

$$\forall \mathbf{N} \in \mathbf{W} \qquad \mathbf{dc_S(In(N))} \leq \mathbf{N}$$

A system **S** is *strongly sound* with respect to **W** if

$$\forall M \in P(W) \qquad dc_S(M) \leq sc_W(M)$$

A system **S** is *weakly complete* with respect to **W** if

$$\forall N \in W \qquad dc_S(In(N)) \geq sc_W(In(N))$$

A system **S** is *decisive* with respect to **W** if

$$\forall N \in W \qquad dc_S(In(N)) = N$$

A system **S** is *strongly complete* with respect to **W** if

$$\forall M \in P(W) \qquad dc_S(M) \geq sc_W(M)$$

In [24] we prove that for monotonic inference relations the three notions of soundness are equivalent (see Appendix). For the notions of completeness it can be proven that if soundness holds, the property of empirical foundness ($\forall N \in W \ \ sc_W(In(N)) = N$) distinguishes weak completeness from decisiveness, and that the property of well-informedness ($\forall M \in P(W)$ $dc_S(M) = \cap N \geq M \in A \ \ dc_S(N)$) distinguishes weak completeness from strong completeness. Note that all equivalences of completeness only hold for sound and monotonic systems. Thus monotonicity is a useful property for verification, and one may obviously consider it for substructures, i.e. any set of cluster of components, of a compositional architecture.

In the following section we will describe compositional verification by assuming an arbitrary property (either soundness, weak/strong completeness, etc), but the ultimate property is that all properties hold, i.e. soundness, decisiveness, and strong completeness; we will call this covering.

3.   Compositionality of verification properties

In Section 2 we defined properties in such a general manner that they are applicable not only to systems with one knowledge base but also to systems that are a combination of a number of components, e.g. according to a compositional architecture.

3.1 The compositional verification principle

The natural question arises whether the properties to be verified are in some sense compositional, i.e., whether the following *compositional verification principle* holds:

  if *all components* **C** *satisfy property* **P**

    and *they are composed in the right manner to the system* **S**

  then *the system* **S** *satisfies property* **P**

In this subsection we will analyse this question and improve its formulation. A similar approach to verification of conventional systems is given in [1], but the theory we propose is tuned to knowledge based systems in a non-trivial way. We make the following four observations.

* admissibility of inputs
A first important observation is that both for a system as a whole and for the components such a property **P** is always relative to some given input from the environment. For example, a system in principle may be inconsistent with respect to some possible input information state, but if this state will never occur in the environment (and as a consequence has been declared not to belong to the set of admissible input states) then the system is not inconsistent. So, the statement above needs some refinement involving sets of admissible input states for both the whole system and the components:

> *if all components   **C** satisfy property **P** w.r.t. their sets of admissible input states AI_C*
>
> >   *and   they are composed in the right manner to the system **S***
> *then   the system **S** satisfies property **P** w.r.t. its set of admissible input states AI_S*

* well-composedness
A second observation is that the condition "are composed in the right manner" is crucial but far from trivial. One aspect is that this criterion of "well-composedness" will have to guarantee that, given the set of admissible inputs AI_S for the whole system, the resulting internally generated input for each of the components **C** falls within its set of admissible input AI_C. Of course in the first place the connections between the components should be made in the proper manner.

> *if all components   **C** satisfy the property **P** w.r.t. their sets of admissible input states AI_C*
>
> >   *and   they are connected in the right manner to the system **S***
> >   *and   for each system input M_S falling within the set of admissible inputs AI_S*
> > >   *each component   **C** receives input M_C within its set of admissible inputs AI_C*
> *then   the system **S** satisfies property **P** w.r.t. its set of admissible input states AI_S*

* admissibility of inputs depends on proper functioning of other components
However, we are almost forced to make our third observation: it is clear that, even with proper connections, if a component **C** receives input from another component **C'** that is unpredictable because it does not function in a proper manner, a guarantee on inputs generated for components being admissible still cannot be given. Therefore in the whole

argumentation to prove that a system satisfies some property **P** (that in general will be related to some aspect of proper functioning) another (or the same ?) notion of proper functioning **Q** of a component **C'** delivering input for **C** comes in (for simplicity we assume here the same set of admissible inputs $AI_{C'}$); for simplicity we take **P = Q**:

> *if*   *(i) all components* **C** *satisfy the property* **P** *w.r.t. their sets of admissible input states* AI_C
>
>   *(ii) they are connected in the right manner to the system* **S**
>
>   *(iii) for each system input* M_S *falling within the set of admissible inputs* AI_S
>
>   *each component* **C** *receives input* M_C *within its set of admissible inputs* AI_C
>
>   *under the assumption that each component* **C'** *with output-input connection to* **C**
>
>   *satisfies property* **P** *w.r.t. its input* $M_{C'}$
>
> *then*   *the system* **S** *satisfies property* **P** *w.r.t. its set of admissible input states* AI_S

* induction principle or circularity

Formulated in this more detailed manner we make a fourth observation. The constructive form of this observation is that an induction argument occurs in the argumentation. To prove that the generated inputs for **C** are admissible, one has to prove that all components **C'** with output-input connection to **C** function properly, which requires that within the system they in their turn receive only admissible input, and so on.

   In essence, for the static case, this leads to a condition on the graph of connections. This graph of connection must not contain cycles, if the induction argument is to be accomplished in a constructive manner. Moreover, two connections coming from different components but offering a truth-value for the same atom, may lead to unwanted inconsistencies.

   In case of an output-input connection graph with cycles the argumentation as pointed out is not constructive anymore: it becomes circular. For example, two components **C** and **C'** can have mutual (bidirectional) output-input connections. To prove proper functioning of **C** within the context of the system one has to assume proper functioning of **C'** within the context of the system and conversely.

There are some further considerations to be made:

- Is it undesirable if the set of admissible inputs for a component **C** is chosen too large, i.e., if there are admissible inputs that never occur in the system ? In that case the condition on **C** satisfying property **P** or **Q** is stronger than needed.
- The subcondition of condition (iii) that **C'** functions properly with respect to the input $M_{C'}$ could be strengthened to the subcondition that **C'** functions properly for all its admissible inputs and $M_{C'}$ is admissible.

- An alternative (non-equivalent) formulation is that $\mathbf{C'}$ functions properly for all its admissible inputs (and leave out that the input $\mathbf{M_{C'}}$ is admissible). In that case by applying condition (iii) on $\mathbf{C'}$ instead of $\mathbf{C}$ it can be established in addition that the input of $\mathbf{C'}$ within the system is admissible.

3.2 Formalization of the compositional verification principle

To be able to make a valid formalization we need to make some assumptions about the connectivity of components in a system.

The connectivity is formalized by a global connection relation $\mathbf{connection(C', C)}$ between components. Such connections will map signatures onto signatures by pairing the atoms in these signatures. Moreover, we assume that each component's input state functionally depends on (is uniquely determined by) the system's input state, which we will represent by extending the same connect relation to $\mathbf{connection(S, C)}$ and $\mathbf{connection(C, S)}$.

In the following formalization we have taken strong soundness and completeness as the property that is verified. We have taken the abstract properties $\mathbf{P}$ and $\mathbf{Q}$ both equal to strong soundness and completeness

Under these assumptions the compositional verification principle can be formalized in the following manner. Admissible inputs are captured by assigning a subset to each component $\mathbf{C}$ and to the whole system $\mathbf{S}$:

$$\mathbf{AI_C \subseteq IS_{in}(C), \ AI_S \subseteq IS_{in}(S)}$$

Thus the following formalization is obtained.

Definition 3.2 (Formalized compositional verification principle; static variant)
The *(static) compositional verification principle* is the following statement.
Suppose the following three conditions hold:

(i) For all components $\mathbf{C}$ and input models $\mathbf{M \in AI_C}$ it holds that $\mathbf{dc_C(M) = sc_{W_C}(M)}$

(ii) There exists a relation $\mathbf{connection(C', C)}$ describing that $\mathbf{C}$ receives input from $\mathbf{C'}$. Using these connections each input model $\mathbf{M_S \in AI_S}$ determines in a unique manner for each component $\mathbf{C}$ an input model $\mathbf{M_C \in AI_C}$.

(iii) For each input model $\mathbf{M_S \in AI_S}$ and each component $\mathbf{C}$ such that
for all $\mathbf{C'}$ with $\mathbf{connection(C', C)}$ it holds $\mathbf{dc_{C'}(M_{C'}) = sc_{W_{C'}}(M_{C'})}$
it holds $\mathbf{M_C \in AI_C}$

Then for all input models $\mathbf{M \in AI_S}$ it holds $\mathbf{dc_S(M) = sc_{W_S}(M)}$.

As remarked before, it is neccesary to formulate a condition for the connectivity before we can prove such a system:

Definition 3.1 (Connection Graph, (static) well-connected)

The *connection graph* for a compositional system is defined by a global relation:
connection(C', C) for components **C'** or **C** together with **connection(S, C)** and **connection(C, S)** for component **C** and the system **S**.

Any such connection connects the signatures associated with both components:

- **connection(C', C)** is a relation on **InAt(Ó$_{C'}$)** x **OutAt(Ó$_C$).**
- **connection(S, C)** is a relation on **InAt(Ó$_S$)** x **InAt(Ó$_C$).**
- **connection(C, S)** is a relation on **OutAt(Ó$_C$)** x **OutAt(Ó$_S$).**

A system is *well-connected* if

- the connection graph is acyclic,
- no two connections **connection(C', C)** and **connection(C'', C)** exist such that **(a',a)** $\in$ **connection(C', C)** and **(a'',a)** $\in$ **connection(C'', C).**

Now in the case of a finite number of components for well-connected systems, one can build a proof of the compositional verification principle by induction on the length of a maximal path in the connection graph. For such a proof we need a basic lemma for two specific situations:

Lemma 3.3

The compositional verification principle is valid for

a) systems composed from components that have no mutual connections.

b) systems that consist of two sequential components

Theorem 3.4

For well-connected systems with a finite number of components the static compositional verification principle is valid.

Proof By induction of the length **n** of a maximal path in the connection graph of S. If **n =1** we have the case of the Lemma 3.2a). If **n > 1**, take **S''** the set of all components at end points of paths with (maximal) length **n**. These components form a system of the type of Lemma 3.2a). The remaining components form a system **S'** with length of maximal path length less than **n**. By the induction hypothesis and Lemma 3.2 for both **S'** and **S''** the compositional verification principle already holds. The system **S** can composed from them by sequential composition, so Lemma 3.2b) can be applied.

4.   Dynamic properties

Essential in the theory above is the notion of forcing or semantic closure. This notion must be called static because it is based on a world description of static situation models. A straightforward generalisation of this notion to dynamic behavior is available if we consider temporal models. In [4] such temporal models were introduced as linear sequences of partial models. 'Time' in such a model has a starting point, is countable and non-branching. Therefore the natural numbers can be taken as the basis:

Definition 4.1.
A temporal model for a system **S** is a sequence of elements of **IS(S)**, i.e., a function of the form:

$$\gamma \in \mathbf{IS(S)}^{\mathbb{N}}$$

Such temporal models can describe the behavior of a knowledge based system by showing the trace of reasoning. Time steps in such a trace may consider time at different levels. Changes may correspond to individual firing of rules in a knowledge base, or at a different level to the determination of the deductive closure of separate chains of rules. Other changes may correspond to change in input for the knowledge base.

   In a compositional system, different parts of a model will correspond to the information states of different components. Time steps will then correspond to the effects of (alternating) activations of these components. This view will become important later when we consider such compositionality.

   The trace of a system given some input can be described by a temporal model, if the system is deterministic. This holds for the case that some initial input is given, but also for input that changes over time. Therefore input is given by a temporal input model:

Definition 4.2.
A *temporal input model* is a temporal model restricted to input models
$$\gamma \in \mathbf{IS_{in}(S)}^{\mathbb{N}}$$

Associated with any temporal model γ we define the input model **In(γ)** that copies the truth-assignment of the input atom and assigns **u** to all others. If any temporal input model deterministically specifies a trace for a system, we can determine one temporal model that describes this. We will use the following terminology: temporal deductive closure.

Definition 4.3.
Given a temporal input model γ the deterministic trace of the system is a temporal model
   tdc$_\mathbf{S}$(γ) called the *temporal deductive closure*.

A naive way to define forcing for such temporal models is to consider a set of them which describes all possible correct and required behavior of a system.

Definition 4.4

A behavior description is a set of temporal models, **B**. A temporal model $\gamma 1$ is refined by a model $\gamma 2$, denoted $\gamma 1 \leq \gamma 2$ if for all times **t** in $\mathbb{N}$

$$\gamma 1(t) \leq \gamma 2(t).$$

Now we can define forcing in the usual way, by taking a temporal model that shows the common part of all refinements for some model:

Definition 4.5

Given a temporal model γ, the *temporal semantic closure* forced in a behavior **B** by γ is a temporal model **tsc$_B$(γ)** such that

- for all γ' in **B** s.t. $\gamma \leq \gamma'$ it holds that **tsc$_B$(γ)** $\leq \gamma'$, and
- for any other model γ'' satisfying this property $\gamma'' \leq$ **tsc$_B$(γ)**

Equivalently stated, **tsc$_B$(γ)** is the greatest common temporal model of all refinements in **B** of γ.

This type of forcing is useful to determine the required behavior from a temporal input model. This is interesting in two ways. First of all, we may want to parallel the intuition behind the temporal deductive closure of a system by determining what temporal model is forced by some temporal input model. Secondly, we may want to know what is the trace that must follow some given prefix:

Definition 4.6

The prefix up to a certain time-point **t** of a temporal model γ is a temporal model **Pre(t,γ)** defined by for all **t'** in **N** and atom **a**:

$$\mathbf{Pre(t,\gamma)(t',a)} = \gamma(t',a) \quad \text{if } t' \leq t,$$
$$= \mathbf{u} \quad \text{otherwise}$$

There are two notions that are interesting in this respect: a temporal model can be determined by its input, and the model at a certain time may be determined by only preceding models. Together they form a natural generalisation of empirical foundedness: no two (distinct) temporal models have the same input, nor identical prefixes.

Definition 4.7

A behavior description **B** is *deterministic* if for all **γ** in **B**

$$tsc_B(In(\gamma)) = \gamma$$

A behavior description **B** is *simulatable* if for all **γ** in **B** and **t** in **N**:

$$tsc_B(Prec(t,\gamma)) = \gamma$$

Obviously, the task of dynamic verification is to compare the actual behavior of a system to the required behavior. Thus we can define, similar to the static case, the following dynamic verification properties:

Definition 4.8

A system **S** is *behaviour-sound* for a temporal input model **γ** if

$$tdc_S(\gamma) \leq tsc_B(\gamma)$$

A system S is *behaviour-complete* for a temporal input model **γ** if

$$tdc_S(\gamma) \geq tsc_B(\gamma)$$

A question we naturally ask ourselves is whether there are other notions for dynamic verification that together form a coherent theory as is the case for static verification presented before.

The above notion of forcing does not take into account that some changes may be sequential in some particular temporal model, but that this sequentiality is not necessary: i.e. there is another temporal model that contains the same behavior except for a subtrace that switches the order of changes. For this we will define the notion of permutation equivalence:

Definition 4.9

Two temporal models **γ1** and **γ2** are *permutation equivalent* $\gamma1 \cong \gamma2$ if there are two intervals **[0,t1]** and **[t2,inf)** with **t1 < t2** such that for all **t** in these intervals $\gamma1(t)=\gamma2(t)$. A temporal model **γ2** is a *permutation refinement* of **γ1** if there exists a model $\gamma3 \cong \gamma2$ and $\gamma3 \geq \gamma1$.

A more sophisticated notion of forcing will take such permutations into consideration. A possible way to formulate this is given below:

Definition 4.10

A *permutation temporal semantic closure* $ptsc_B(\gamma)$ is forced in a behavior **B** by a model **γ** if it is the largest common temporal model of all permutation refinements in **B** of **γ.**

5.   Compositionality of dynamic properties

Compositionality implies that each information state of the whole system S is composed from the information states of its components C. The set of them can be described by the cartesian product

$$IS(S) \quad = \pi_{C \in S} IS(C)$$

Behaviours of the whole system can be described by sequences (indexed by the natural numbers) of these composed information states:

$$IS(S)^{\mathbb{N}} \quad = \pi_{C \in S} IS(C)^{\mathbb{N}}$$

We consider the formulation and formalization of a dynamic variant of the compositional verification principle (as in Section 3). First we need to interpret the term property in a dynamic manner: a *property of behaviour*: BP. Some of these properies have been defined in Section 4.

   We assume all components in principle are active all the time and react on their (dynamic) inputs. Among these inputs *control-atoms* can play an important role; e.g., an input atom that serves as a kind of "on-off" button with the effect that the component only has trivial behaviour (not changing anything) as long as this atom is false (cf. the meta-information states in compositional architectures).

   Admissible inputs are replaced by *admissible input behaviours*: for each component and for the system as a whole a given subset

$$AIB_C \subseteq IS_{in}(C)$$

 Thus we obtain the following *dynamic compositional verification principle.*

> if (i) all components C satisfy (behaviour) property BP w.r.t. their sets of admissible input
>   behaviours AIB_C
>   (ii) they are connected in the right manner to the system S
>   (iii) for each system input behaviour M_S falling within the set of admissible input behaviours AIB_S
>   each component C receives input behaviour M_C within its set of admissible input behaviours AIB_C
>   under the assumption that each component C' with output-input connection to C
>   satisfies (behaviour) property BP w.r.t. its input behaviour $M_{C'}$
>   then   the system S satisfies (behaviour) property BP w.r.t. its set of admissible input behaviours AIB_S

About this dynamic variant of the principle similar considerations can be made as in Section 3. We assume deterministic behaviour of the system and all of its components. This means

that all local (component) behaviour functionally depends on the input behaviour. A difference is that in a dynamic setting cyclic connection graphs are sensible.

In a formal definition we can state the principle as follows.

Definition 5.1 (Formalized dynamic compositional verification principle)
The *(dynamic) compositional verification principle* is the following statement.
Suppose the following three conditions hold:

(i) For all components **C** and input behaviours $\mathbf{M} \in \mathbf{AIB_C}$ the property **BP** holds.

(ii) There exists a relation **connection(C', C)** describing that **C** receives input from **C'**. Using these connections each input behaviour $\mathbf{M_S} \in \mathbf{AIB_S}$ determines in a unique manner for each component **C** an input behaviour $\mathbf{M_C} \in \mathbf{AIB_C}$.

(iii) For each input behaviour $\mathbf{M_S} \in \mathbf{AIB_S}$ and each component **C**
such that for all **C'** with **connection(C', C)** it holds **BP**
it holds $\mathbf{M_C} \in \mathbf{AIB_C}$

Then for all input models $\mathbf{M} \in \mathbf{AIB_S}$ it holds **BP**.

For the behaviour property **BP** one can substitute one of the behaviour properties formulated in the previous section.

6. Conclusions

In our paper [24] we presented a logical foundation for verification and validation for the restricted case of knowledge-based systems consisting of just one knowledge base. Moreover we only treated static properties to be verified. In the current paper we investigated how this framework can be generalized to the case of compositional knowledge-based systems and also to dynamic properties. First, we found out that it is not hard to generalize them to more complex systems (see the results in Section 2). Moreover, it turned out that generalizations to the dynamic case are possible as well (see Section 4). Given these generalizations of our framework we were able to formulate a compositional verification principle both for static and dynamic properties (Sections 3 and 5). We consider this principle as one of the main research topics in the practice and foundation of verification of compositional knowledge-based systems. One of the most challenging problems to tackle is the case of verification of the behaviour of compositional knowledge-based systems. The generalized logical framework presented in the current paper can be a first step in this direction; it enables one to formalize the various issues that can be identified.

7. References

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

[2]   Blamey, S., "Partial Logic", in: D. Gabbay and F. Guenthner (eds.), *Handbook of Philosophical Logic,* Vol. III, 1-70, Reidel, Dordrecht, 1986

[3]   R. Davis, "Use of Meta-Level Knowledge in the Construction of Large Knowledge Bases," Ph.D dissertation, Computer Science Department, Stanford University, Stanford, California, 1976.

[4]   Engelfriet, J. and Treur, J., A Temporal Model Theory for Default Logic, in: *Proc. 2nd European Conference on Symbolic and Quantitative Approaches to Reasoning and Uncertainty, ECSQARU '93* M. Clarke, R. Kruse, S. Moral (Eds.), , Springer Verlag, 1993, pp. 91-96. Extended version: Report IR-334, Vrije Universiteit Amsterdam, Department of Mathematics and Computer Science, 1993, 38 pp.

[5]   Engelfriet, J. and Treur, J., Temporal Theories of Reasoning, in: *Proc. Fourth European Workshop on Logics in AI, JELIA'94,* L.M. Pereira and D. Pearce (eds.), , Springer Verlag, 1994.

[6]   V. Guibert, A. Beauvieux, and M. Haziza, "Consistencey, Soundness and Completeness of a Diagnostic System," in *Validation, Verification and test of knowledge-based systems*, by M. Ayel and J-P. Laurent (Eds), Wiley & Sons, 1991.

[7]   H. Herre, "Semantical completeness of model-based diagnosis", *EUROVAV-93*, Univ. of Leipzig, 1993.

[8]   I.A. van Langevelde and J. Treur, "Tackling the incompleteness of chaining", Report IR-274, Department of Mathematics & Computer Science, AI-Section, Vrije Universiteit, Amsterdam, 1991.

[9]   I.A. van Langevelde, A. Philipsen, and J. Treur, "Formal Specification of Compositional Architectures", in B. Neumann (ed.), *Proc. of ECAI-92*, Wiley & Sons, 1992., pp. 272-276

[10]   H.L. Larsen, and H. Nonfjall, "Modeling in the design of a KBS validation system," in *Int. J. Intelligent Systems,* 6, p 759-775, 1991.

[11]   H.L. Larsen, and H. Nonfjall, "Detection of Potential Inconsistencies in Knowledge Bases," in *Int. J. Intelligent Systems,* 7, p 81-96, 1992.

[12]   P. Leemans, "*On the Verification of Knowledge in Expert System Modules*", Masters Thesis, Department of Mathematics & Computer Science, AI-Section, Vrije Universiteit, Amsterdam, 1993.

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

[14]   W. Marek, "Completeness and consistency in knowledge based systems," *Proc. of the First Int. Conf. on Expert Database Systems*, Charleston, South Carolina, p. 119-126, 1986.

[15]   P.L. Miller et al., "Expert system knowledge acquisition for domains of medical workup: An augmented transition network model," *Proc. of the 10th Annual Symposium on Computer Applications in Medical Care*, Washington D.C., p 30-35, 1986.

[16]   T.A. Nguyen, "Verifying consistency of production systems," *Proc. of the third IEEE Conference on AI Applications*, Orlando, Florida p 4-8, 1987.

[17]   T.A. Nguyen, W.A. Perkins, T.J. Laffey, and D. Pecora, "Checking an expert system knowledge base for consistency and completeness," in *Proc. of IJCAI-85*, Los Angeles, p 375-378, 1985.

[18]   M.-C. Rousset, "On the consistency of knowledge bases: the COVADIS system," in *Proc. of ECAI-88*, Pitman Publishing, p 79-84, 1988.

[19]   M. Suwa, A. Garlisle Scott, and E.H. Shortliffe, "Completeness and consistency in a rule based system," in B.G. Buchanan and E.H. Shortliffe, Rule-based Expert Systems, Addison-Wesley, 1985 p 159-170.

[20]   M. Suwa, A. Scott, and E.H. Shortliffe, "An approach to verifying completeness and consistency in a rule based expert system," *AI Magazine*, 3(4), p 16-21, 1982.

[21]   J. Treur, "Completeness and definability in diagnostic expert systems", in *Proc. of ECAI-88*, Munich, Germany, Pitman Publishing, p 619-624, 1988.

[22]   J. Treur, "Declarative Functionality Descriptions of Interacting Reasoning Modules", in *Proc. of Int. Workshop on Declarative Knowledge*, *PDK-91,* Boley & Richter (Eds.), Springer Verlag, p. 221-236, 1991.

[23]   J. Treur and Th. Wetter (eds.), *Formal Specification of Complex Reasoning System*s, Ellis Horwood, 284 pp, 1993.

[24]   J. Treur and M. Willems, A logical foundation for Verification, in *Proc. of ECAI-94*, Amsterdam, The Netherlands, A. Cohn (Ed.), John Wiley & Sons, Ltd., 1994, pp. 745-749.