A Semantical Perspective on Verification of Knowledge

Paul Leemans, Jan Treur, Mark Willems
Vrije Universiteit Amsterdam, Department of Artificial Intelligence
De Boelelaan 1081a, 1081 HV Amsterdam The Netherlands
Email: treur@cs.vu.nl URL: http://www.cs.vu.nl/~treur

Abstract
The usual approaches to verification of knowledge in the literature have a rather practical nature, and are sometimes ad hoc and depending on the syntactical structure of a particular knowledge representation language in the definitions of the properties that are tested. In this paper a more principled investigation provides a well-defined semantical basis, independent of any particular syntactical representation, of the various properties that can be tested, and their logical relationships. Central in this unifying semantical framework is the notion of domain description and the forcing relation that is used as a standard for the inferences. Properties covered include consistency, three variants of soundness, three variants of completeness, empirically foundedness, and well-informedness.

Keywords: Verification, knowledge, semantics, consistency, soundness, completeness, empirical foundation, well-informedness.

1 Introduction

With the widespread use of knowledge in some form of representation, the need for verification of knowledge has become more important. In situations where knowledge-based systems support or even replace people, reliability and correct performance are of major importance; e.g., [20], [37]. An automatic verifier can serve as a very useful tool during the development of a (knowledge-based) system. It makes it possible to check the knowledge, prior to testing by identifying, for example, inconsistency, incorrectness, redundancy, and missing knowledge, thus supporting the knowledge engineer in the knowledge elicitation and modelling process; e.g., [29].

1.1 Syntactic versus Semantic Perspective on Verification

Knowledge and domain specific information are commonly described by specific representation languages, such as rules, decision tables, frames or objects. In most literature on verification and validation (V&V) of knowledge systems a rule-based representation language is considered. The reason for this is that they are the most often practically used representation of knowledge implemented.

A number of existing approaches to knowledge base verification can be found in [1], [5], [9], [14], [15], [16], [17], [18], [19], [21], [22], [23], [24], [25], [26], [28], [29], [30], [31], [32]; in [37] a comparative overview can be found for some of the existing approaches. As argued in [37], one of the problems in almost all of the existing approaches to knowledge base verification is that both the properties addressed (e.g., consistency, correctness, completeness) and the techniques used for verification strongly depend on the knowledge

representation language in which the knowledge base is expressed. Most references address the case of rule-based knowledge representation; e.g., [22], [23], [24], [26], [30], [31], [32]. Other contributions focus on decision tables (e.g., [5]), frames (e.g., [6]), or other representation languages.

This dependency on a specific knowledge representation language leads to ad hoc syntactical definitions of the properties, to problems in the transfer of the approaches from one knowledge representation formalism to another one, or problems in combing them in hybrid knowledge bases, e.g., knowledge bases combining rules and objects with inheritance (cf. [37]).

Most of the anomalies are checked, in practice, for the case of rule-based knowledge representation by considering the syntactic form of the rules: syntactic verification. Such properties are not directly interpretable independent of the knowledge representation format used: they are not semantic notions. The main aim of this paper is to propose semantic notions, and to relate syntactic notions from the literature to these semantic notions. In [37] one of the research issues identified as to be addressed in more depth is theoretical foundations, in the sense of formal semantics for knowledge base anomalies. This paper indeed contributes such semantical foundations of a major part of the notions known from the literature on verification of knowledge. Moreover, to define the semantics of these various notions, one unifying semantical framework is offered.

1.2 The Role of Situation-Specific Information

Reasoning processes as performed by human expert consultants or knowledge-based reasoning systems often involve an interaction with the outside world (e.g., a user or sensor) to obtain the situation-specific information needed in the reasoning. Therefore two collaborating roles for providing knowledge may be distinguished: the role of the reasoner (or reasoning system) is to provide (domain-specific but) situation-independent domain knowledge, while the role of the client or user is to provide situation-specific domain knowledge (information). The *interactive reasoning*, which results from this collaboration, combines these two types of knowledge. From the viewpoint of the reasoner, the knowledge acquired from the client or user may be viewed as domain information "from outside", in contrast to the reasoner's own (internal) domain knowledge. In [38], which inspired our view on interactive reasoning, the situation-specific information is represented by a so-called simulation structure, in contrast to the knowledge base, which is represented as a logical theory. From the logical perspective a simulation structure is a model of the theory defined by the knowledge base.

To obtain adequate criteria for such an interactive reasoning process to draw correct conclusions, suitable definitions are needed of notions of soundness and completeness with respect to the given domain. Basically verification can be described as a comparison between

the behaviour of a knowledge base (**KB**) and the requirements. But for more complex knowledge bases it is not so clear how one should state these requirements. One could imagine the expert to specify the input-output correspondence of the component by means of a table defining the functionality. This is similar to the approach in [9] where a table of atoms is given that specifies what cause leads to which effects. In this paper we will introduce a particular view of verification by considering these requirements essentially as a domain description defining the possible situations (**W**), and determine the standards for derivation from it. Theoretically this domain description should be seen as the set of all possible situations that can occur (e.g., at different times). Obviously, listing this is not possible in a practical setting especially if the number of possible situations is very large. A useful idea for verification, in this case, is either to consider a smaller set of typical (crucial) situations, which can be used as a test-set, or define the set of world situations by a set of constraints.

We will introduce a number of properties that can be attributed to a **KB** *in relation to* **W**, and are interesting for verification. We will show that the usual approach to verification is to check for special (usually syntactically defined) cases of these more general semantic notions. Verification, in our view, is essentially a comparison between the domain description **W** and the knowledge base **KB**.

1.3 The Potential Problems

Errors in a knowledge base can arise through a variety of reasons. The simplest source of errors is, of course, the typographical mistake. An other, often occurring source of error is the alteration of the system knowledge in order to accommodate wider application coverage. Adding, changing and refining knowledge base elements can easily lead to problems. Yet another source of error is ignorance or limited knowledge of the expert or the knowledge engineer, which can very well lead to a system with incorrect or non-specific recommendations. The larger the knowledge base, the more difficult it is to see whether there are errors in the knowledge base. Knowledge base properties addressed in this paper are in the first place:

- *Consistency properties*
  No conflicting conclusions can be made from valid information (see Section 3),
- *Soundness properties*
  No invalid inferences can be made from valid information. In Section 3 three variants of definitions of soundness are put forward, which, however, (as shown in Section 3), if the inference relation is monotonic, semantically are equivalent.
- *Completeness properties*
  No deficiency: no valid information fails to produce desired conclusions. A more in-depth analysis reveals that at least three semantically different variants of completeness can be identified (Sections 4 to 6).

A Prolog programme, DES-CHECK has been developed to check a knowledge base on these notions. In addition to these properties some other useful and related properties are identified such as implicit definability, explicit definability, and empirically foundedness of a domain (Section 5), and well-informedness of a knowledge base (Section 6). Moreover, a number of theorems established the logical connections between the notions. In Section 7 it is shown how the perspective put forward here can be used to verify more complex knowledge-based systems. It is addressed what kinds of properties are needed in example compositional model: a generic process model for diagnosis. In Section 2 the basic concepts needed for the rest of the paper are presented. The paper concludes with a discussion in Section 8.

2 Basic Semantic Concepts

Knowledge bases and situation-specific information are commonly described by representation languages, such as rules, decision tables, frames, or objects. However, qua representation the expressive power of a large class of languages essentially does not exceed that of propositional logic, especially taking into account that within representations variables often (like in logic programming) can be eliminated by replacing them by all possible instances that are relevant in a certain context. Of course for the computational aspects of inferencing it may make a lot of difference whether and how variables or other abstractions are used; however, these inference process aspects are not addressed here. For a discussion on how object- , frame like or object-attribute-value representations as often used in knowledge system development environments can be related to the semantical propositional formalisation used in this paper, see Section 2.2, between Definition 2.2 and 2.3.

Therefore, as we are interested in semantical properties of a system, rather than computational or visual properties, a propositional approach suffices here: we will focus on the sets of (ground) atoms that can be built in a given language rather then on the language itself. In this manner we can abstract from the specific language that is dealt with.

Logical means are used to define two types of entities for a system, the knowledge and the facts. In logic it is not common to distinguish input facts versus output facts. Also a set of possible situations is hardly ever considered explicitly (rather the model theoretic semantics of a knowledge base defines such a set implicitly). We will show that both these distinctions are important for dedicated foundations of verification.

As discussed above our approach exploits a description of the world. Such a description is given by a set of situation models, each describing a possible or typical situation of the world. The whole set describes either the set of all possible situations or a set of typical test cases. In short, verification should check whether the output that is derived from an input is desirable with respect to this world.

2.1 Situation Models

In our language situations are described by a finite *signature* Σ, a sequence of (ground) atoms based on any given language**.** In a reasoning component one can distinguish three kinds of atoms according to the three subsets of the set of ground atoms $\mathbf{At}(\Sigma)$ based on Σ.

$$\mathbf{At}(\Sigma) = \mathbf{InAt}(\Sigma) \cup \mathbf{InternalAt}(\Sigma) \cup \mathbf{OutAt}(\Sigma)$$

They are called the *input* atoms, the *private* or *internal* atoms, and the *output* atoms. One should note here that in the literature this is a less common distinction as usually it is assumed that any atom of a signature can be given (input) or queried (output). However, in accordance with software engineering principles on information hiding, in practice this is a very useful distinction. The sets of input and output atoms are disjoint from the set of internal atoms, but they are not necessarily disjoint from each other.

Here, the input atoms describe facts that are influenced or observed from outside the system, the output atoms are those that one is interested in, and the internal or private atoms are those that can be used to determine the relation between input and output. The latter function as a sub goal or intermediate step in the derivations. In an inference they are derived from inputs (or other intermediary atoms) and are then used to continue the inference so as to reach the outputs (i.e. they subsequently serve as conditions in other rules).

A useful notation is an ordered tuple

$$\Sigma = < \mathbf{s}_1, \mathbf{s}_2, \ldots ; \mathbf{i}_1, \mathbf{i}_2, \ldots ; \mathbf{h}_1, \mathbf{h}_2, \ldots >$$

that groups together the input, internal, and output atoms, respectively. Obviously, the **s**'s, the **i**'s and the **h**'s can be arbitrary words, like **flu**, **defect(cd_dirty)** and **temperature(dog, high)**. The propositional equivalent of an atom with a variable is the set of all instantiations of the variables; for instance **defect(X: OBSERVATIONS)** is equivalent to the set of atoms **defect(cd_dirty)**, **defect(cd_upside_down)**, etc. where **X** is instantiated by all objects that have been declared for the sort **OBSERVATIONS**.

Definition 2.1

An actual situation is described as a (*complete*) *situation model* (or *complete model*) **N** that is a truth-assignment to the atoms in a signature Σ, i.e., a mapping $\mathbf{N : At}(\Sigma) \rightarrow \mathbf{\{0, 1\}}$. The set of complete models for signature Σ is denoted by $\mathbf{CMod}(\Sigma)$. The concise way to show the truth assignments is to give an ordered tuple of truth-values **0** or **1** that is in agreement with the signature Σ (e.g., see the example world description below).
A *domain* (or *world) description* is a set **W** of situation models, i.e., any subset $\mathbf{W} \subseteq \mathbf{CMod}(\Sigma)$.

As an example consider the following world description

$$\mathbf{W} = \{ \mathbf{N_1}, \mathbf{N_2}, \mathbf{N_3} \}$$

with respect to the signature $\Sigma = < \mathbf{s_1}, \mathbf{s_2}, \mathbf{s_3}, \mathbf{s_4}, \mathbf{s_5} ; \mathbf{i_1} ; \mathbf{h_1}, \mathbf{h_2} >$ where

$$\mathbf{N_1} = < \mathbf{1, 0, 1, 0, 0 ; 1 ; 1, 0} >$$
$$\mathbf{N_2} = < \mathbf{1, 0, 0, 1, 0 ; 1 ; 0, 1} >$$
$$\mathbf{N_3} = < \mathbf{0, 1, 0, 0, 1 ; 0 ; 1, 0} >$$

In principle the world could be described extensionally, by listing explicitly all possible (or interesting) situations. If the set of ground atoms is finite, as often is the case in practice, the set of all possible world situations will be finite. During knowledge engineering this set can often only be acquired in the form of a sample of (the most characteristic) situation models. Another way to describe these situations is to give a set of axioms that serve as constraints for the situations, see also [14], [15]. In this case a world description is the set of all possible situations that fulfil these constraints, which often can be chosen in such a way that a finite set remains (e.g., by giving bounds to numerical variables). However, when one wants to define foundations of verification, one can abstract from this issue and just assume that a set of situations is defined, leaving open the manner in which this set of situations is described. In this paper, sometimes the set is assumed finite, but then this will explicitly be stated.

In order to express facts about the domain more conveniently it is possible to build up propositions from the atoms in our signature using the logical signs $\wedge$ (conjunction: and), $\vee$ (disjunction: or), $\rightarrow$ (implication: if ... then ...) and $\neg$ (negation: not). When an arbitrary proposition $\mathbf{p}$ *is true / holds* in some situation model $\mathbf{N}$ we will use the truth relation symbol $\models$ to write $\mathbf{N} \models \mathbf{p}$.

A *literal* is a proposition that is either an atom or a negation of an atom, and for an arbitrary signature Σ we distinguish the sets of *input literals* (sometimes also called *observables*) $\mathbf{InLit(\Sigma)}$, and the set of *output literals* (sometimes also called *hypotheses*) $\mathbf{OutLit(\Sigma)}$. Furthermore $\mathbf{\sim c}$ is defined as $\neg c$ if $\mathbf{c}$ is an atom and as $\mathbf{a}$ if $\mathbf{c} = \neg \mathbf{a}$.

2.2 Information States and Knowledge Bases

It is appropriate to distinguish the models that reflect situation specific information of a domain from the knowledge base, **KB**, that gives the general information about a domain.

The goal of such a knowledge base is to establish a relationship between the input and output facts of a signature, so when given some input facts the knowledge base should be able to derive some output facts.

In general a system does not have full access to the actual situations. Indeed, it is the (unobservable) output part of these situations that one is interested in. Moreover, at a certain moment in time only some of the observations may be available, and only some of the

conclusions that may be drawn have been derived. For example, when a reasoning component starts to reason, not all input facts of a situation need to be known at that moment, and no conclusions have been derived yet. Therefore the following is defined (cf. 13]):

Definition 2.2

A *partial model* **M** is a mapping $\mathbf{M : At(\Sigma) \rightarrow \{0, 1, u\}}$

For a many-sorted predicate logic variant, see [12]. Such a partial model describes the (current) *information state* of the reasoning component. We will sometimes omit the word "partial" and simply use the word "model" for a partial model. A partial model that has no unknown atoms (a situation model, for instance) is a complete model.

In an implementation of a knowledge base in an Knowledge System Development Environment, the information state of the component is represented by what is often called the (dynamic) *facts base*. The facts in the facts base are usually represented by objects and attributes of them. We will show how such representations can be formalized in the form of a partial model. An attribute with a Boolean value such as in

bird1.female = false

encodes the information that the atomic statement

female(bird1)

is false; this corresponds to the symbol **0** in a partial model, assigned to the atom **female(bird1)**. Similarly, a Boolean value true corresponds to a symbol **1** in a partial model; for instance **male(bird1)** has the truth value **1**. An attribute with a non-Boolean value codes a set of atomic statements. For example the expression

bird1.color = grey

stating that the attribute **colour** of **bird1** has the value **grey**, encodes the information that the propositional atomic statement

colour(bird1, grey)

is true. In the same time we can have the information that **colour(bird1, red)** is false. The attributes for which no values are filled in in the objects base may be interpreted as 'unknown' or as 'undefined'. These express the partiality of the information state; they correspond to the symbols **u** in a partial model. Assume that we do not have information about the attribute 'size'. Then both

size(bird1, tall)

size(bird1, small)

are unknown. So this for simple example we can build the partial model, corresponding to

<female(bird1), male(bird1), colour(bird1, grey), colour(bird1, red), size(bird1, tall), size(bird1, small)>

in the following manner:

< 0, 1, 1, 0, u, u >

Notice that there may be some dependencies between the different atoms. We will not discuss these at this place.

Definition 2.3

Associated with any (partial) model **M** we define the input model **In(M)** that copies the input truth-assignment of **M**, but assigns **u** to all other atoms, and similarly the output model **Out(M)**. In connection to a world description **W** we define the sets **In(W)** and **Out(W)** as the sets of input models respectively output models that are associated to the situations in **W**.

$\neg\varphi$	
1	0
0	1
u	u

$\varphi \wedge \psi$	1	0	u
1	1	0	u
0	0	0	0
u	u	0	u

$\varphi \vee \psi$	1	0	u
1	1	1	1
0	1	0	u
u	1	u	u

$\varphi \rightarrow \psi$	1	0	u
1	1	0	u
0	1	1	1
u	1	u	u

Figure 1 Strong Kleene truth tables

Because we use Strong Kleene semantics (cf. [13]; see Fig. 3), the truth or satisfaction relation $\models$ applies to partial models too, and we can express the truth of a proposition **p** in a (partial) model **M** by $\mathbf{M} \models \mathbf{p}$. This naturally leads to an alternative representation of partial models by a set of literals, i.e. the set of all literals that are true in **M**. Thus all atoms mentioned in this set are known (either true or false), whereas the literals not in the set are assumed to be unknown. Yet another way to represent such a partial model is by a proposition: the conjunction of the literals in the related set of literals. This conjunction will contain all literals that are known, and any atom that does not occur has truth-value unknown.

Note that we are using partial logic here. This has a relation to the fact that we do not assume the closed world assumption, so negations must be explicitly observed, or deduced. An essential part of partial logic is the refinement relation as defined below: This can be understood by defining a partial ordering on the set of truth values by $\mathbf{u} < \mathbf{0}$, and $\mathbf{u} < \mathbf{1}$.

Definition 2.4

Let **M** and **N** be two arbitrary partial situation models with respect to a signature Σ. Now **N** is a *refinement* of **M**, denoted by $\mathbf{M} \leq \mathbf{N}$, if for all atoms $\mathbf{a} \in \mathrm{At}(\Sigma)$ it holds $\mathrm{M}(a) \leq \mathrm{N}(a)$. Equivalently,

$$\mathbf{M} \leq \mathbf{N} \quad \Leftrightarrow \quad \forall \mathbf{c} \in \mathrm{Lit}(\Sigma) \;\; \mathbf{M} \models \mathbf{c} \Rightarrow \mathbf{N} \models \mathbf{c}$$

Derivation or inference in our view is a process that sequentially derives new facts from old ones, thus always refining the partial model representing the current information state. In essence, we view inference as a process of refinement, and in particular we would like to refine the (observable) input model to the complete model that describes the actual situation that is under consideration. Here the following property is essential.

Lemma 2.5

For any (partial) model $\mathbf{M}$, it holds $\mathbf{In(M) \leq M}$.

Another important property of input models is that they maintain the refinement ordering.

Lemma 2.6

For any two (partial) models $\mathbf{M, N}$ such that $\mathbf{M \leq N}$ it holds: $\mathbf{In(M) \leq In(N)}$

Given a set of models $\mathbf{V}$ it is sometimes useful to consider the set $\mathbf{P(V)}$ of partial models that can be refined to a model in $\mathbf{V}$. For a world description $\mathbf{W}$ the set $\mathbf{P(W)}$ can be viewed as the search space of the system. Reasoning is a process that starts with a partial situation model or information state, and subsequently tries to refine this model until an answer is found.

The set $\mathbf{P(In(W))}$ gives us the set of all possible (partial) inputs of a domain. As we stated before in the literature the set of possible inputs is often described by a set of constraints that an input model must obey. The following lemma is useful when using the operator $\mathbf{P}$.

Lemma 2.7

For any world description $\mathbf{W}$, we have $\mathbf{P(In(W)) = In(P(W))}$. □

We base our notions on a given inference relation $\vdash$ that is assumed monotonic and sound but not necessarily complete (for example chaining, or unit resolution, or Horn resolution, or full resolution). Based on this given (generic) inference relation any language that is used to formulate a knowledge base will have an associated (specific) *inference relation*; the notation $\mathbf{M \vdash_{KB} p}$ denotes that a proposition $\mathbf{p}$ can be inferred from the information represented in a model $\mathbf{M}$ by the inference relation associated with $\mathbf{KB}$. In fact we consider $\vdash$ as a ternary relation (instead of a binary relation as usual) due to our strict separation of situation independent domain knowledge from situation-specific information.

For a knowledge base in rule format (if-then-rules) an obvious inference relation is chaining. Our rule format will consist of implications that have only one literal as conclusion, and a conjunction of literals as their condition. Having a negation in a conclusion is sometimes not allowed in the literature (if one restricts oneself to Horn-clauses, for instance). Continuing the example above we shall consider the following $\mathbf{KB}$ for the signature

$\Sigma = < s_1, s_2, s_3, s_4, s_5 \ ; i_1 ; h_1, h_2 >.$

$$
\begin{aligned}
s_1 \wedge \neg s_2 &\rightarrow i_1 \\
\neg s_5 &\rightarrow \neg i_1 \\
i_1 \wedge s_3 &\rightarrow h_1 \\
i_1 \wedge s_4 &\rightarrow h_2
\end{aligned}
$$

We will list some criteria that are useful to enforce on an inference relation. Conservativity seems obvious, that what is already known in a model can be derived from it (and, in fact cannot be overwritten by other inferences). Monotonicity implies that a larger input should infer a larger output. Chaining is an example of an inference relation that obeys these properties.

Definition 2.8

Given a knowledge base **KB**, any proposition **p**, and arbitrary models **M**, **N** $\in$ **P(W)**, all with respect to a signature Σ, we distinguish the following properties for the inference relation related to the knowledge base:

- *conservativity* $\mathbf{M} \models \mathbf{p}$ implies $\mathbf{M} \vdash_{\mathbf{KB}} \mathbf{p}$.
- *monotonicity* $\mathbf{In(M)} \leq \mathbf{In(N)}$ implies $\mathbf{In(M)} \vdash_{\mathbf{KB}} \mathbf{p} \Rightarrow \mathbf{In(N)} \vdash_{\mathbf{KB}} \mathbf{p}$.

Even though monotonicity seems very intuitive, in practice it is not used in many systems. A frame system, for instance, may specify default attribute values. These defaults are assumptions that may be (non-monotonically) retracted later. Also systems that use the Horn clause subset of predicate logic are often non-monotonic. They often apply the closed world assumption and the absence of an atom is interpreted as a negation. This negation may need to be retracted when the atom later becomes known. In this paper monotonicity is important for verification of properties like correctness, and soundness.

2.3 Forcing

A question one might ask oneself in light of verification is how one should specify the desired behaviour of a reasoning component. One could imagine the expert to specify the input-output correspondence of the component by means of a table defining the functionality. This is similar to the approach in [9] where a table of atoms is given that specifies what cause leads to which effects.

Our idea is to just consider the domain description **W** itself, and try to determine the standards for derivation from it. Here we consider the fact that some relation $\models_{\mathbf{W}}$ specifies what a knowledge base should derive.

Definition 2.9

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

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

In a way this notion sets a standard for the knowledge base, what are the hypotheses it should derive given some partial model. Before we proceed we will mention a number of useful properties of forcing.

Lemma 2.10

a) For any model $\mathbf{M} \in \mathbf{P(W)}$ and proposition $\mathbf{p}$ it holds $\mathbf{M} \models \mathbf{p} \Rightarrow \mathbf{M} |\models_W \mathbf{p}$

b) For any two models $\mathbf{M_1}, \mathbf{M_2} \in \mathbf{P(W)}$ and proposition $\mathbf{p}$ such that $\mathbf{M_1} \leq \mathbf{M_2}$ it holds

$$\mathbf{M_1} |\models_W \mathbf{p} \Rightarrow \mathbf{M_2} |\models_W \mathbf{p}$$

Here a) is trivial; the proof of b) runs as follows. If $\mathbf{N} \geq \mathbf{M_2}$ then $\mathbf{N} \geq \mathbf{M_1}$ which together with $\mathbf{M_1} |\models_W \mathbf{h}$ implies $\mathbf{N} \models h$ for arbitrary $\mathbf{N} \in \mathbf{W}$. We have $\mathbf{M_2} |\models_W \mathbf{h}$.

In general the properties for forcing and inference are closely related, and what is natural for forcing can be enforced on inference. The main theme of this paper is that this is what verification should check. The following corollary parallels conservatism and monotonicity, if one remembers Lemma 2.6 by which $\mathbf{M} \leq \mathbf{N}$ implies $\mathbf{In(M)} \leq \mathbf{In(N)}$. Like monotonicity does for inference, the following corollary then says for forcing that a larger input model forces a larger output. In fact this may be seen as the reason for assuming monotonicity of inference in the first place. Also the conservatism property is satisfied for forcing.

Corollary 2.11

a) For any model $\mathbf{M} \in \mathbf{P(W)}$ and proposition $\mathbf{p}$ it holds

$$\mathbf{In(M)} \models \mathbf{p} \Rightarrow \mathbf{In(N)} |\models_W \mathbf{p}$$

b) For any two models $\mathbf{M} \in \mathbf{P(W)}$ and $\mathbf{N} \in \mathbf{W}$ such that $\mathbf{M} \leq \mathbf{N}$ it holds:

$$\mathbf{In(M)} |\models_W \mathbf{h} \Rightarrow \mathbf{In(N)} |\models_W \mathbf{h}$$

The proof uses that if $\mathbf{M} \leq \mathbf{N}$ then $\mathbf{In(M)} \leq \mathbf{In(N)}$ which follows from Lemma 2.6; then Lemma 2.10 can be used.

Lemma 2.12

For any model $\mathbf{N} \in \mathbf{W}$: $\mathbf{In(N)} |\models_W \mathbf{h} \Rightarrow \mathbf{N} \models \mathbf{h}$

Since $\mathbf{N \geq In(N)}$ the definition of forcing immediately proves the lemma.

2.4 Verifying a knowledge base

Besides defining foundations verification, it is important to be able to actually verify a knowledge base in practice. In this light we should make some remarks about complexity. First of all, there are exponentially many more models in $\mathbf{P(W)}$ than there are in $\mathbf{W}$. Therefore, checking inference $\vdash_{\mathbf{KB}}$ against forcing $\Vdash_{\mathbf{W}}$ will probably be more complicated than checking it against truth $\models$. Usually there are many models in $\mathbf{W}$ so to compute forcing is time consuming. Moreover, no domain description is considered in the literature. These might be the reasons why verification in the literature has only considered the truth relation.

Now there is a problem between forcing, on the one hand as the basis for verification, and on the other hand as the more complicated property to check. A solution is to consider particular illegal constellations of rules that can be proven to violate one of our notions. In the literature this (special case) approach to verification has had a lot of emphasis, mainly because special occurrences of rules are more easily checked than the more difficult notions like soundness, completeness, etc.

The following notion describes what the test is that the knowledge base makes for the hypotheses in terms of the observables. It defines a proposition that is induced by a knowledge base for each hypothesis.

Definition 2.13

A *knowledge base expression* of a literal $\mathbf{h}$ in $\mathbf{OutLit(\Sigma)}$ is a proposition $\mathbf{p}$ in terms of the observables in $\mathbf{InLit(\Sigma)}$ such that $\mathbf{h}$ can be deduced from $\mathbf{M}$ via $\mathbf{KB}$ exactly when $\mathbf{p}$ is true in any situation $\mathbf{M} \in \mathbf{W}$:

$$\mathbf{In(M) \models p \;\Leftrightarrow\; In(M) \vdash_{KB} h}$$

In a rule-based system, with chaining as inference relation, this proposition can be found in the form of a disjunction of conjunctions of literals. The disjunction in a sense lists all possible cases (conjunctions) that lead to an hypothesis $\mathbf{h}$. If nothing concludes hypothesis $\mathbf{h}$, then the proposition for that hypothesis $\mathbf{h}$ is always false (i.e., an empty disjunction).

Example 2.14

Consider a signature $\mathbf{< p, q ; r >}$, and a knowledge base with just two rules $\mathbf{KB = \{ p \wedge q \rightarrow \neg r \,,\, \neg p \rightarrow \neg r \}}$. Then a knowledge base expression for the output literal $\neg r$ is the disjunction $\mathbf{(p \wedge q) \vee (\neg p)}$. Of course, a subtlety arises if some inputs do not arise in a input description. For instance, if $\mathbf{W_{in} = \{ <0,1>, <1,0>, <0,0> \}}$ an alternative knowledge base expression is $\mathbf{(\neg p)}$ describing the starting point of the only "relevant" chain.

If $\mathbf{h}$ is an element from $\mathbf{OutLit(\Sigma)}$ then a knowledge base expression will be made not only for $\mathbf{h}$, but for its negation $\neg \mathbf{h}$ as well. This expression is not necessarily the negation of the proposition $\mathbf{p}$ for $\mathbf{h}$. A knowledge base expression is particularly useful when implementing verification of correctness and decisiveness of knowledge bases with respect to domain description, as will be shown in following sections.

3 Consistency and Correctness

An important issue in the verification of knowledge-based systems is to assure that the knowledge applied by the system does not derive any contradiction. There are two related notions lying behind this statement. First, we want the knowledge base to be such that it does not derive two conclusions that are inconsistent. Secondly, we want the conclusions to be consistent with the domain we are considering. These two notions are related as will be shown below.

3.1 Consistency

A first property that we define here is consistency: we would like inference to be such that one and the same model can never lead to some proposition as well as its negation. If a knowledge base has this property under an inference relation, we call it *consistent.*

Definition 3.1

Given a set of partial models $\mathbf{S}$ for a signature Σ, and an inference relation $\vdash_{\mathbf{KB}}$. We say that the inference is *consistent* if for any model $\mathbf{M} \in \mathbf{S}$ and proposition $\mathbf{p}$

never $\mathbf{M} \vdash_{\mathbf{KB}} \mathbf{p}$ and $\mathbf{M} \vdash_{\mathbf{KB}} \neg \mathbf{p}$.

The common definition of consistency is that two rules (or inferences) are inconsistent if they succeed in the same situation, but have conflicting results. In our view, it misses the point that the input models that lead to these inconsistencies may never occur. Therefore we formulated consistency as a property of the knowledge base *with respect to* a certain domain description.

3.2 Correctness

A system is correct w.r.t. a domain description $\mathbf{W}$ when it holds for any specific situation model $\mathbf{N} \in \mathbf{W}$ that if an output atom or its negation is inferred from the input information in that situation $\mathbf{N}$, via the $\mathbf{KB}$, then it must also be true in $\mathbf{N}$. This is a notion that is commonly used in the literature of verification.

Definition 3.2

Assume $\mathbf{W}$ is a domain description with respect to a signature Σ, and a knowledge base $\mathbf{KB}$. Then $\mathbf{KB}$ is *correct* with respect to $\mathbf{W}$ if for all output literals $\mathbf{h} \in \mathbf{OutLit}(\Sigma)$ the following holds:

$$\forall \mathbf{N} \in \mathbf{W} \quad \mathbf{In(N)} \vdash_{\mathbf{KB}} h \quad \Rightarrow \quad \mathbf{N} \models \mathbf{h}$$

Closely related to this is what we call soundness. This checks whether what is derived from the input in an actual situation should indeed follow from that input. In the previous section we described that forcing will be used to set the standards. The main difference with correctness is that we use the forcing relation $\Vdash_{\mathbf{W}}$ instead of the satisfaction relation $\models$.

Definition 3.3

Assume a domain description $\mathbf{W}$ with respect to a signature Σ, and $\mathbf{KB}$ a knowledge base. Then $\mathbf{KB}$ is *weakly sound* with respect to $\mathbf{W}$ if for all output literals $\mathbf{h} \in \mathbf{OutLit}(\Sigma)$ the following holds:

$$\forall \mathbf{N} \in \mathbf{W} \quad \mathbf{In(N)} \vdash_{\mathbf{KB}} \mathbf{h} \quad \Rightarrow \quad \mathbf{In(N)} \Vdash_{\mathbf{W}} \mathbf{h}.$$

Because an inference can take place also when we have only partial input information about a situation, we will define a second kind of soundness that checks the above property for all possible partial models instead of only the complete models.

Definition 3.4

Assume a domain description $\mathbf{W}$ with respect to a signature Σ, and a knowledge base $\mathbf{KB}$. Then $\mathbf{KB}$ is *strongly sound* with respect to $\mathbf{W}$ if for all output literals $\mathbf{h} \in \mathbf{OutLit}(\Sigma)$ the following holds:

$$\forall \mathbf{M} \in \mathbf{P(W)} \quad \mathbf{In(M)} \vdash_{\mathbf{KB}} \mathbf{h} \quad \Rightarrow \quad \mathbf{In(M)} \Vdash_{\mathbf{W}} \mathbf{h}.$$

The last of the above three definitions seems to be the strongest because it checks all possible partial models instead of only the complete models. Moreover, it checks forcing instead of the simpler holds relation. However, it can be proven that for a knowledge base with a monotonic inference all these notions are equivalent.

Theorem 3.5

Let $\mathbf{W}$ be a domain description and $\mathbf{KB}$ a knowledge base with respect to signature Σ. If the inference relation $\vdash_{\mathbf{KB}}$ is monotonic with respect to $\mathbf{W}$, then the following are equivalent:

(i) $\mathbf{KB}$ is strongly sound with respect to $\mathbf{W}$.

(ii) $\mathbf{KB}$ is weakly sound with respect to $\mathbf{W}$.

(iii) $\mathbf{KB}$ is correct with respect to $\mathbf{W}$.

The proof can be found in the Appendix. The notion of correctness (soundness) is related to consistency as defined in Section 3.1. Indeed, if a knowledge base satisfies the weak soundness we can prove consistency.

Theorem 3.6

If a knowledge base **KB** is weakly sound with respect to **W**, it is also consistent with respect to **In(W)** and output atoms.

The proof runs as follows. If **KB** is weakly sound we know for any model $\mathbf{N} \in \mathbf{W}$ and output atom $\mathbf{h} \in \mathbf{OutAt}(\Sigma)$ that if $\mathbf{In(N)} \vdash_{\mathbf{KB}} \mathbf{h}$ and $\mathbf{In(N)} \vdash_{\mathbf{KB}} \neg \mathbf{h}$ then these two facts together imply $\mathbf{In(N)} \models_{\mathbf{W}} \mathbf{h}$ and $\mathbf{In(N)} \models_{\mathbf{W}} \neg \mathbf{h}$. In turn by Lemma 2.12 this implies $\mathbf{N} \models \mathbf{h}$ and $\mathbf{N} \models \neg \mathbf{h}$. This is a contradiction, so **KB** is consistent with respect to **In(W).**

If a rule set contains rules that lead to contradicting results, the behaviour of the knowledge base is not clear. The final advice the expert system gives will depend on the strategies for selecting rules. It is possible that incorrect conclusions will be drawn.

3.3 Verifying Consistency and Correctness

When verifying correctness the notion of rule-based expression turns out useful. Consider the signature $\Sigma = < \mathbf{s_1, s_2, s_3, s_4, s_5 ; i_1 ; h_1, h_2} >$ and the domain description **W** given by**:**

$$\mathbf{N_1} = \mathbf{< 1, 0, 1, 0, 0 ; 1 ; 1, 0 >}$$
$$\mathbf{N_2} = \mathbf{< 1, 0, 0, 1, 0 ; 1 ; 0, 1 >}$$
$$\mathbf{N_3} = \mathbf{< 0, 1, 0, 0, 1 ; 0 ; 1, 0 >}$$

that was reasoned about by the **KB** with rules:

$$\mathbf{s_1} \wedge \neg \mathbf{s_2} \rightarrow \mathbf{i_1}$$
$$\neg \mathbf{s_5} \rightarrow \mathbf{i_1}$$
$$\mathbf{i_1} \wedge \mathbf{s_3} \rightarrow \mathbf{h_1}$$
$$\mathbf{i_1} \wedge \mathbf{s_4} \rightarrow \mathbf{h_2}$$

Correctness can be established by considering all possible inferences to hypotheses. For example, because $\mathbf{In(N_1)} \models \mathbf{s_1} \wedge \neg \mathbf{s_2}$ it allows the chain

$$\mathbf{s_1} \wedge \neg \mathbf{s_2} \rightarrow \mathbf{i_1} \;\; ; \;\; \mathbf{i_1} \wedge \mathbf{s_3} \rightarrow \mathbf{h_1}$$

Thus $\mathbf{h_1}$ can be inferred by **KB**, and therefore $\mathbf{In(N_1)} \vdash_{\mathbf{KB}} \mathbf{h_1}$. Correctness is obeyed, because $\mathbf{N_1} \models \mathbf{h_1}$. No other inference chain leads to an output from $\mathbf{In(N_1)}$. Because we do not assume the closed-world assumption we cannot derive (by failure) any negations of hypotheses. Next, $\mathbf{In(N_2)}$ allows the chain $\mathbf{s_1} \wedge \neg \mathbf{s_2} \rightarrow \mathbf{i_1}$; $\mathbf{i_1} \wedge \mathbf{s_4} \rightarrow \mathbf{h_2}$, so $\mathbf{In(N_2)} \vdash_{\mathbf{KB}} \mathbf{h_2}$. There is no

incorrectness because $\mathbf{N_2} \models \mathbf{h_2}$. Finally, $\mathbf{In(N_3)}$ allows nothing to be inferred via the **KB**, so correctness holds automatically here.

A program can check the knowledge base expressions (see Definition 2.13) of every hypothesis and its negation. In this way it is easy to check whether e.g. $\mathbf{In(N_1)}$ infers $\mathbf{h_1}$ by taking the proposition **p** associated with $\mathbf{h_1}$, and checking whether $\mathbf{In(N_1)}$ makes **p** true. If so, then check whether $\mathbf{N_1} \models \mathbf{h_1}$. Do this for every hypothesis and its negation, i.e. do it for every proposition that has been built. The following lemma allows one to easily check for correctness in this way.

Lemma 3.7

Let **p** be the knowledge base expression for a literal **h** from $\mathbf{OutLit}(\Sigma)$. If all models $\mathbf{N} \in \mathbf{W}$ such that $\mathbf{In(N)} \models \mathbf{p}$ also satisfy $\mathbf{N} \models \mathbf{h}$ then **KB** is correct with respect to **W.**

There is no need to check weak and strong soundness once correctness has been checked, because (assuming the inference relation is monotonic) they are equivalent by Theorem 3.5.

Lemma 3.8

If all rules **r** in **KB** are true in all models of **W**, then **KB** is correct with respect to **W.**

Another way to verify correctness is given in the above lemma. It is possible to check whether each rule of **KB** is true in all models of **W**. This follows as one can easily show that then any chain of rules that is true in a model, provides a conclusion that is true in that model too. By Theorem 3.6 we have then automatically checked consistency.

The literature considers several special cases of rules that make a knowledge base inconsistent. Some of these are self-contradicting rules and rule-chains, and more important, contradicting pairs of rules and rule-chains. The first two cases, those of self-contradicting antecedents in rules and rule-chains, are not consistency in the semantic sense. They will be considered as a case of redundancy later.

• self-contradicting rule:

This is a rule which contradicts itself and of which the negation of its conclusion occurs in its antecedent. For example, $\mathbf{p} \wedge \mathbf{q} \rightarrow \neg p$.

• self-contradicting chain of inference:

Same as above, but the contradiction does not occur until after some inferences. For example, $\mathbf{p} \wedge \mathbf{q} \rightarrow \mathbf{r}$; $\mathbf{r} \wedge \mathbf{s} \rightarrow \neg p$.

If they occur in reasoning sessions, these examples are shown to be special cases of our definition of inconsistency as follows. For $\neg p$ to be inferred, $\mathbf{p}$ has to be true (in both examples); so, $\mathbf{In(N)} \vdash_{\mathbf{KB}} \mathbf{p}$. Now assume (considering both examples) $\mathbf{q}$ and $\mathbf{s}$ (in the second example) are in $\mathbf{In(N)}$, then $\mathbf{In(N)} \vdash_{\mathbf{KB}} \neg \mathbf{p}$, and the inconsistency follows.

For self-contradicting inferences it is checked if the conditions that have to be true to derive the contradiction are ever true simultaneously in a model. When all the conditions of a rule or chain are input literals then the contradiction is easily verified in the models. However, if one of these conditions contains an internal literal, it is somewhat more difficult to see if the internal literal is true for a model. This is because internal literals are not specified in models (only input and output literals). To check the truth value of an internal literal in a model, it is checked if the observables in one of the models together with the rules from the KB infer the internal literal. When all conditions, that have to be true to infer a contradiction, are true in the model (input and internal literals), the contradiction is also implied by the model.

• contradicting rule-pair:

  This is the case when two rules succeed with the same antecedent, but with conflicting conclusions. For example, $\mathbf{p} \wedge \mathbf{q} \rightarrow \mathbf{s}$; $\mathbf{p} \wedge \mathbf{q} \rightarrow \neg \mathbf{s}$.

• contradicting chain of inference:

  Same as above, but the contradiction happens after some inferences. For example, $\mathbf{p} \rightarrow \mathbf{q}$; $\mathbf{q} \rightarrow \mathbf{r}$ and $\mathbf{p} \rightarrow \mathbf{s}$; $\mathbf{s} \rightarrow \neg r$.

Note that verifying contradicting rules can lead to an extra possibility of contradiction, one that is not considered when checking without a domain description. In a model a conflict can not only occur when two rules or chains of rules with conflicting conclusions have equivalent conditions, but additionally conflicts occur for rules with different conditions and conflicting conclusions. For example, consider the following rules: $\mathbf{p} \wedge \neg q \rightarrow \mathbf{s}$, and $\mathbf{p} \wedge \mathbf{r} \rightarrow \neg \mathbf{s}$. Without a domain description the conflict is not detected, but a conflict arises in a model if $\mathbf{p}$, $\neg q$, and $\mathbf{r}$ are true simultaneously. If no such model exists in the domain there is no inconsistency.

These examples are special cases of our definition of inconsistency too. Take for example the two rules from contradicting rule-pairs: $\mathbf{p} \wedge \mathbf{q} \rightarrow \mathbf{s}$; $\mathbf{p} \wedge \mathbf{q} \rightarrow \neg \mathbf{s}$. Apply the definition of consistency and one can see $\mathbf{In(N)} \vdash_{\mathbf{KB}} \mathbf{s}$ and $\mathbf{In(N)} \vdash_{\mathbf{KB}} \neg \mathbf{s}$, where $\mathbf{In(N)}$ is the input part of model $\mathbf{N}$, here based on $\mathbf{p}$ and $\mathbf{q}$.

4 Weak Completeness and Gaps in the Knowledge Base

The development of a knowledge-based system is an iterative process in which knowledge is tested, added, changed, and refined. This iterative process can leave gaps in the knowledge base, which both the knowledge engineer and the expert may have overlooked during the knowledge acquisition process.

In the previous section the direction of verification was from the knowledge base back to the world. For any derivation it was checked whether it violated any criteria set by the world. In this section the direction will be reversed, and the idea is that the standards as set by the world must be achieved by the inferences. In general this is called completeness; as will turn out in this and subsequent sections three different types of completeness can be defined.

The gaps or missing rules may or may not be easy to detect, but they usually indicate that an intended derivation is not possible. This incompleteness may be intentional or not; intentionally missing rules relate to those cases where the omitted rules do not have any bearing on the task at hand. Unintentionally missing rules are a result of omissions in the development process.

4.1 Weak Completeness

Usually incompleteness is indicated by a gap or missing rule. If a hypothesis should be derivable from an input model, possibly because it holds in a situation with that input, but a derivation of that hypothesis cannot be performed then there may be some missing rules. Suwa et al. [31] have informally defined missing rules as follows: "A situation exists in which a particular inference is required, but there is no rule that succeeds in that situation and produces the desired conclusion."

Such a definition is based on a knowledge base in a certain format. But what it does in essence is to capture a relation between “a desired conclusion” and “a particular inference”. In our formalism this can be defined as a relation between forcing $\models_{\mathbf{W}}$ and inference $\vdash_{\mathbf{KB}}$ as in the following definition.

Definition 4.1

Assume a domain description $\mathbf{W}$ with respect to a signature Σ, and a knowledge base $\mathbf{KB}$. Then $\mathbf{KB}$ is *weakly complete* with respect to $\mathbf{W}$ if for all output literals $\mathbf{h} \in \mathbf{OutLit}(\Sigma)$ the following holds:

$$\forall \mathbf{N} \in \mathbf{W} \quad \mathbf{In(N)} \models_{\mathbf{W}} \mathbf{h} \quad \Rightarrow \quad \mathbf{In(N)} \vdash_{\mathbf{KB}} \mathrm{h}$$

Note that weak completeness is the converse of weak soundness. These two notions, soundness and weak completeness show the essence of verification as described here. They show most clearly the comparison between forcing and inference. Where the notions of correctness and soundness w.r.t. a domain description were shown to be equivalent, the notions of (weak/strong) completeness and decisiveness (as will be defined in subsequent sections) are not equivalent. They are, however, if some additional properties of the knowledge base and domain description are satisfied (i.e., well-informedness and empirically foundedness, introduced in Section 5 and 6).

Consider the following example again.

W: $N_1 = < 1, 0, 1, 0, 0 ; 1 ; 1, 0 >$

$N_2 = < 1, 0, 0, 1, 0 ; 1 ; 0, 1 >$

$N_3 = < 0, 1, 0, 0, 1 ; 0 ; 1, 0 >$

$s_1 \wedge \neg s_2 \rightarrow i_1$

$\neg s_5 \rightarrow i_1$

$i_1 \wedge s_3 \rightarrow h_1$

$i_1 \wedge s_4 \rightarrow h_2$

The knowledge base in the example is not weakly complete because the input of N_2 forces $\neg h_1$, and the inputs of N_1 and N_3 both force $\neg h_2$. These hypotheses are not inferred from the appropriate inputs, for the simple reason that there is no rule that contains these literals as its conclusion. We should add rules like $s_4 \rightarrow \neg h_1$ and $\neg s_4 \rightarrow \neg h_2$ to make the knowledge base weakly complete with respect to **W**.

4.2 Gaps in the knowledge base

Checking for weak completeness can be complex mainly because forcing is a complicated notion. In line with the literature, we have identified five cases indicative of gaps (incomplete information): unnecessary literals, dead ends, unreachable literals, unreachable rules, and illegal literals. The main intuition behind these special cases is that the occurrence of an atom in the signature indicates that the atom is needed to determine (it helps to force) the truth of some hypothesis in some situation. An atom **a** is essential for the forcing of an output literal **h** if any model that forces **h** either assigns unknown to **a** or its sub model with unknown assigned to **a** does not force **h**. We will try to show that the cases below can be seen as special cases of failure of weak completeness.

*** Unnecessary literals**

A literal that is mentioned in the input signature is *unnecessary* if it is not mentioned in any rule, neither in the antecedent nor in the conclusion, and if it is not mentioned in the output.

In the latter case, the literal can have a value through interaction with other components and pass that value on. Then an input literal is necessary even if it does not occur as an antecedent of any rule, and an output literal is necessary even if it does not occur as a conclusion of any rule.

For example suppose we have the relation **problem** with the set of legal values (input-objects) **{cd_dirty, cd_upside_down, power_off}**. If **problem(cd_dirty)** and **problem(cd_upside_down)**

occur in the antecedent of a rule in the knowledge base, but **problem(power_off)** does not, and **problem(power_off)** does not occur in the output signature then either the knowledge base is a rule short or the atom can be deleted from the input signature.

This can be seen as a special case of failure of weak completeness. Suppose there is a model **N**, with the literal **s** in it, that forces an output **h** (so $\mathbf{In(N)} \models_{\mathbf{W}} \mathbf{h}$), and the output **h** cannot be inferred from **In(N)** and the knowledge base (i.e., $\mathbf{In(N)} \not\vdash_{\mathbf{KB}} \mathbf{h}$). This is because there is no rule that uses the literal **s** in question as part of its condition. So weak completeness does not hold, and there is a missing rule. Then again, if the literal in question is not used in any model to force **h** then it is unnecessary.

*** Illegal literals**

This is the case when a rule refers to a literal that does not occur in any signature. For example, **problem(power_off)** occurs in a rule's antecedent but does not occur in the input or internal signature.

The same holds for output literals. Some conclusion is drawn by a rule in the knowledge base, but that conclusion is not part of the output signature nor is it part of the internal signature. Conditions are not allowed to come from the output signature, unless they are declared in the input signature as well. In the same way, conclusions can only be input literals in they are declared in the output signature as well. Actually, the occurrence of illegal literals is not really part of weak completeness, but it is more a declaration error.

This part is actually a kind of syntax checker as well: if one misspells something, like **problem(cd_dirty)**, then this is determined to be an illegal literal, because **problem(cd_dirty)** is the right input literal. Illegal literals in models are values which are declared in the models-file, but which are not declared in the signatures of the component.

*** Dead-ends**

A dead end occurs in an inference chain if a fact, that is not an output, is concluded and does not form a part of the antecedents of any other rule. So **q** is a dead end, if in the rule $\mathbf{p} \rightarrow \mathbf{q}$, the literal **q** is no output and is no part of the antecedents of any other rule. A dead-end related to a model occurs when from the observables of a model via the **KB** the dead-end can be inferred.

*** Unreachable rules**

A rule is not reachable if one of its conditions is not concluded in any rule, and is not an input literal. This means that the rule will never be used in a chain. Unreachability of rules does not depend on whether or not a domain description is considered.

*** Unreachable literals**

This happens when there is a literal that is concluded in a rule, but the rule itself and all rules that conclude it are unreachable. This looks a lot like unreachable rules, but there is a difference. Consider the rules $\mathbf{s} \rightarrow \mathbf{h}$ and $\mathbf{i} \rightarrow \mathbf{h}$. The second rule is unreachable if there are no rules that conclude the internal. This does not mean that the output hypothesis is unreachable. The output is still reachable but through the first rule. One could in fact merge unreachable rules and unreachable literals into one, because every unreachable literal is also part of an unreachable rule, so it will be reported twice (the opposite does not hold, i.e. not every unreachable rule contains an unreachable literal).

This can also be seen as a special case of failure of weak completeness: say $\mathbf{r}$ is forced by $\mathbf{In(N)}$ (so $\mathbf{In(N)} \models_W \mathbf{r}$), but it is not possible to infer $\mathbf{r}$ (i.e. $\mathbf{In(N)} \nvdash_{KB} \mathbf{r}$) because the rule(s) that conclude r are unreachable. So, the knowledge base is not weakly complete. In the same trivial way it can be shown that dead-ends, unreachable literals, and unreachable rules are special cases of lack of weak completeness.

5 Empirically Foundedness and Decisiveness

Our basic intuition behind verification is that one should check whether the knowledge base in some sense fits the description of the world, i.e., that it always gives the right answer about the output atoms. In order to answer this question it is important to know whether the signature that is used to describe the world is detailed enough to make such a knowledge base at all possible. It simply may be the case that the domain description is underspecified, and due to this fact there cannot be found a knowledge base that always gives the right answer. This section addresses this.

5.1 Empirically foundedness

We would like the domain or world to be described in such a way that each hypothesis is testable in terms of (can be connected by reasoning to the) observables; this criterion is also extensively discussed in the literature on the Philosophy of Science; e.g., [10]. An obvious way to check this is to make certain that for any output literal $\mathbf{h}$ there is a proposition $\mathbf{p}$ in terms of observables that is equivalent to it in all situation models (inspired by [3]).

Definition 5.1

The hypotheses (or output literals) of a signature Σ are *explicitly definable* with respect to a world description $\mathbf{W}$ if for every output literal $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$, there is a proposition $\mathbf{p}$ in terms of the observables in $\mathbf{InLit(\Sigma)}$ such that any situation $\mathbf{N} \in \mathbf{W}$ it holds

$$\mathbf{N} \models \mathbf{h} \quad \Leftrightarrow \quad \mathbf{N} \models \mathbf{p}$$

This definition would seem to imply that every hypothesis has to be given a proposition, but how such a proposition can be found is not said. As we defined in Section 2.4 the knowledge base gives rise to a proposition of input literals for each hypothesis. As we will show in Section 5.3 what we want for decisiveness is that this knowledge base expression satisfies the criterion of explicit definability.

A more implicit way to check whether a world description is empirically founded is given in the following definition (inspired by [3]).

Definition 5.2

The hypotheses (or output literals) of a signature Σ are *implicitly definable* with respect to a world description $\mathbf{W}$ if for every pair of situations $\mathbf{N_1}, \mathbf{N_2} \in \mathbf{W}$ it holds that if $\mathbf{N_1}$ and $\mathbf{N_2}$ have the same observables (or input literals) then they must have the same hypotheses as well:

$$\mathbf{In(N_1) = In(N_2)} \Rightarrow \mathbf{Out(N_1) = Out(N_2)}$$

This property will turn out to be important, because it is an easy testable criterion for decisiveness, see Section 5.2; no knowledge base is decisive unless the hypotheses of a world description are implicitly definable.

Definition 5.3

A world description $\mathbf{W}$ with respect to a signature Σ is *empirically founded* if for every output literal $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$

$$\forall \mathbf{N} \in W \qquad \mathbf{N} \models \mathbf{h} \Rightarrow \mathbf{In(N)} \models_{\mathbf{W}} \mathbf{h}$$

An equivalent way of defining this is:

$$\forall \mathbf{N} \in W \qquad \mathbf{In(N)} \models_{\mathbf{W}} \mathbf{h} \vee \mathbf{In(N)} \models_{\mathbf{W}} \neg \mathbf{h}$$

The inverse implication was already shown to hold in Lemma 2.12 and then it is not too difficult to prove that empirical foundedness is equivalent to explicit definability as well as to implicit definability.

Theorem 5.4

Assume a domain description $\mathbf{W}$ with respect to a (finite) signature Σ. The following statements are equivalent:

(i) The hypotheses are explicitly definable w.r.t. $\mathbf{W}$

(ii) The hypotheses are implicitly definable w.r.t. $\mathbf{W}$

(iii) $\mathbf{W}$ is empirically founded.

The proof can be found in the Appendix. It is easy to see that the domain description in the example in Section 2 is empirically founded because the hypotheses are implicitly definable. Consider the models of the domain description:

$$\mathbf{N_1} = \mathbf{< 1, 0, 1, 0, 0 ; 1 ; 1, 0 >}$$
$$\mathbf{N_2} = \mathbf{< 1, 0, 0, 1, 0 ; 1 ; 0, 1 >}$$
$$\mathbf{N_3} = \mathbf{< 0, 1, 0, 0, 1 ; 0 ; 1, 0 >}$$

Clearly the hypotheses are implicitly definable because all input models are different. We then also know that the hypotheses are explicitly definable, and indeed for each hypothesis there exists a proposition that is equivalent in all models: for instance, for all $\mathbf{N} \in \mathbf{W}$

$$\mathbf{N} \models \mathbf{h_1} \quad \Leftrightarrow \quad \mathbf{N} \models (\mathbf{s_1} \wedge \neg \mathbf{s_2} \wedge \mathbf{s_3}) \vee \mathbf{s_5}$$
$$\mathbf{N} \models \mathbf{h_2} \quad \Leftrightarrow \quad \mathbf{N} \models \mathbf{s_1} \wedge \neg \mathbf{s_2} \wedge \mathbf{s_4}$$
$$\mathbf{N} \models \neg \mathbf{h_1} \quad \Leftrightarrow \quad \mathbf{N} \models \mathbf{s_1} \wedge \neg \mathbf{s_2} \wedge \neg \mathbf{s_3}$$
$$\mathbf{N} \models \neg \mathbf{h_2} \quad \Leftrightarrow \quad \mathbf{N} \models \neg \mathbf{s_4}$$

Note that these propositions are made from the models are just examples of propositions that obey the definitions. They are not the simplest that are possible, nor identical to the knowledge base expressions shown in Section 2.4. However, we want the rule base expressions to form explicit definitions of the hypotheses.

Domains may exist for which domain descriptions have been modelled that are not empirically founded. As will be discussed in Section 5.2 below, for such domains it is impossible to find a knowledge base that fulfils certain desirable properties (i.e., decisiveness, see below), since the existence of such a knowledge base implies empirically foundedness. In such cases it is very useful if in an analysis of a knowledge base it can be found out that in fact the domain description is the cause of the problems. These problems can usually be solved if additional observation possibilities can be added to the domain description that allow for discrimination between hypotheses that cannot be discriminated within the given domain description.

5.2 Decisiveness

Not only do we want a domain to be empirically founded (the definition can be paraphrased as the hypotheses are in principle testable with the observables), we moreover want the actual knowledge base **KB** to perform this test. This is expressed by the following definition where each literal that is true in a model of the domain can be inferred from / tested with the input.

Definition 5.5

Assume a domain description **W** with respect to a signature Σ, and a knowledge base **KB**. Then **KB** is *decisive* with respect to **W** if for all output literals $\mathbf{h} \in \mathbf{OutLit}(\Sigma)$ the following holds:

$$\forall \mathbf{N} \in W \qquad N \models \mathbf{h} \quad \Rightarrow \quad \mathbf{In(N)} \vdash_{\mathbf{KB}} h$$

If **KB** is correct with respect to **W**, then an equivalent definition is:

$$\forall \mathbf{N} \in W \quad \mathbf{In(N)} \vdash_{\mathbf{KB}} h \vee \mathbf{In(N)} \vdash_{\mathbf{KB}} \neg h$$

A variant of this notion was introduced (under the name completeness) in [33]. Note that decisiveness is the converse of correctness. Moreover, decisiveness can be viewed as a variant of completeness, because if a hypothesis **h** is true in **N**, and **h** cannot be inferred from the observables of **N** by **KB**, then **KB** is incomplete (i.e. a rule should probably be added), or of course **W** is too large (i.e. it contains a counter example that may be deleted).

Notice that Definition 5.5 only guarantees that a hypothesis can be decided upon if sufficient input is available; e.g., in any case a complete input model suffices. In practical cases often only a partial input model will be present. It then depends on the situation whether or not a hypothesis can be decided; Definition 5.5 gives no guarantee for an arbitrary given partial input model. However, what Definition 5.5 does guarantee is that there at least exists a suitable extension of the input information that allows to decide upon the hypothesis. A knowledge base that is not decisive does not give such a guarantee. A decisive knowledge base has this guarantee and thus this property supports the use of (e.g., heuristic) information gathering strategies to extend the input information. Decisiveness says that if the strategy ultimately can exhaust the acquisition of all possible input information (e.g., some form of A*-search), then there will be a point in time that sufficient information is available to make a decision on the hypothesis.

The definition of decisiveness parallels the definition of empirical foundedness (see Section 5.1) and thus imposes a (possible) property of forcing onto inference. Indeed, the criterion that distinguishes weak completeness from decisiveness is precisely the empirical foundedness of the domain as defined in Section 5.1.

Theorem 5.6

Assume a domain description **W**, and a sound knowledge base **KB** with respect to a signature Σ. Then the following notions are equivalent:

(i) **KB** is decisive with respect to **W**

(ii) **KB** is weakly complete with respect to **W** and **W** is empirically founded.

The proof can be found in the Appendix. In [34] the situation is addressed where the observables in a domain have a layered structure, such that decisiveness can be achieved

(only) if the observations of all layers are allowed, but if only observables of some of the layers are taken into account, then decisiveness does not hold. The layered structure of the observables induces a number of levels of relative incompleteness. In [34] methods are presented to focus observation such that a higher level of decisiveness can be achieved taking into account only the relevant observations of the next layer.

5.3 Verifying decisiveness

Decisiveness is not a very difficult notion to check, because one just considers all models and checks whether the knowledge base expressions are satisfied. This is relatively simple because only the complete models need to be checked, and one does not need to consider forcing. The following two observations can be made for decisiveness:

1. A knowledge base does not always have to be decisive (especially not in the case of domains that are not empirically founded).
2. When a component is decisive, then the closed world assumption could be applied to the part of the knowledge base involved in derivation of output atoms to obtain conclusions about the negative output literals.

Shortcomings in decisiveness are mostly unintentional. Consider the example domain description $\mathbf{W = \{N_1 , N_2 , N_3 \}}$ with the models:

$\mathbf{N_1 : < 1, 0, 1, 0, 0; 1, 0 >}$
$\mathbf{N_2 : < 1, 0, 0, 1, 0; 0, 1 >}$
$\mathbf{N_3 : < 0, 1, 0, 0, 1; 1, 0 >}$

Now consider the next **KB** with this set **W**:

$\mathbf{s_1 \wedge \neg s_2 \rightarrow i_1}$
$\mathbf{i_1 \wedge s_3 \rightarrow h_1}$
$\mathbf{i_1 \wedge s_4 \rightarrow h_2}$
$\mathbf{\neg s_4 \rightarrow \neg h_2}$
$\mathbf{s_5 \rightarrow h_1}$
$\mathbf{\neg s_3 \wedge \neg s_5 \rightarrow \neg h_1}$

In the example decisiveness can be established. First of all, the hypotheses in **W** are checked for implicit definability (see Definition 5.2). It is easy to see that the hypotheses are implicitly definable. Now, the system can be checked for decisiveness (according to the definition). An easy way to check if $\mathbf{In(N) \vdash_{KB} h}$ is given in the following lemma.

Lemma 5.7

Let $\mathbf{p}$ be the knowledge base expression for a literal $\mathbf{h}$ from $\mathbf{OutLit(\Sigma)}$. If all models $\mathbf{N} \in \mathbf{W}$ such that $\mathbf{N} \models \mathbf{h}$ also satisfy $\mathbf{In(N)} \models \mathbf{p}$ then $\mathbf{KB}$ is decisive with respect to $\mathbf{W}$.

Thus testing $\mathbf{In(N)} \vdash_{\mathbf{KB}} \mathbf{h}$ is reduced to testing $\mathrm{In}(\mathbf{N}) \models p$, whenever $\mathbf{N} \models \mathbf{h}$ (as was used for explicit definability; see Definition 5.1). For example, for the hypotheses $\mathbf{h_1}$ and $\mathbf{h_2}$ the following propositions can be built from **KB:**

$$\mathbf{p = (s_1 \wedge \neg s_2 \wedge s_3) \vee s_5}$$
$$\mathbf{p = s_1 \wedge \neg s_2 \wedge s_4}$$

Now the following applies:

$$\mathbf{N} \models \mathbf{h_1} \quad \leftrightarrow \quad \mathrm{In}(\mathbf{N}) \models (\mathbf{s_1} \wedge \neg \mathbf{s_2} \wedge \mathbf{s_3}) \vee \mathbf{s_5}$$
$$\mathbf{N} \models h_2 \quad \leftrightarrow \quad \mathrm{In}(\mathbf{N}) \models \mathbf{s_1} \wedge \neg \mathbf{s_2} \wedge \mathbf{s_4}$$

This means that if $\mathbf{h_1}$ is true in an $\mathbf{N}$ say $\mathbf{N_1}$ then the observables in $\mathbf{N_1}$ will have to make $\mathbf{p}$ (i.e., $\mathbf{(s_1 \wedge \neg s_2 \wedge s_3) \vee s_5}$) true and vice versa.

This is the case as one can see ($\mathbf{s_1}$, $\neg \mathbf{s_2}$ and $\mathbf{s_3}$ are true in $\mathbf{N_1}$). Now decisiveness for model $\mathbf{N_1}$ can be easily verified:

$$\mathbf{N_1} \models \mathbf{h_1} \quad \& \quad \mathbf{In(N_1)} \vdash_{\mathbf{KB}} \mathbf{h_1},$$
$$\mathbf{N_1} \models \neg \mathbf{h_2} \quad \& \quad \mathbf{In(N_1)} \vdash_{\mathbf{KB}} \neg \mathbf{h_2}$$

So, with respect to the model $\mathbf{N_1}$ the knowledge base $\mathbf{KB}$ is decisive. It is not hard to see that with respect to the other two models decisiveness also holds. It is important to note that with this way of testing, a proposition can be built only for literals that are concluded by the rules in the knowledge base. If a literal is true in a model, but there is no rule that concludes that literal, then the proposition associated with that literal is false, and consequently $\mathbf{KB}$ is not decisive for that model.

The concepts of correctness and decisiveness are relative with respect to the world $\mathbf{W}$ of real situations. This appears when $\mathbf{W}$ is expanded with new situations. It can very well be that in the new reality the system is not decisive anymore for every situation.

For example: if one adds situation model $\mathbf{N_4}$ **: < 0, 0, 0, 1, 1; 1, 1 >** to $\mathbf{W}$ then the system is still correct, ($\mathbf{In(N_4)} \vdash_{\mathbf{KB}} \mathbf{h_1} \Rightarrow \mathbf{N_4} \models \mathbf{h_1}$, and neither $\mathbf{h_2}$ nor its negation can be derived from the observables in $\mathbf{N_4}$ and the $\mathbf{KB}$). But the knowledge base is not decisive anymore:

$$\mathbf{N_4} \models h_1 \Rightarrow \mathbf{In(N_4)} \vdash_{\mathbf{KB}} \mathbf{h_1},$$

but

$$\mathbf{N_4} \models h_2 \text{ and } \mathbf{In(N_4)} \nvdash_{\mathbf{KB}} \mathbf{h_2},$$

i.e., $\mathbf{In(N_4)}$ does not make the knowledge base expression of $\mathbf{h_2}$ true. So the thing to do now is to shrink $\mathbf{W}$ to get a knowledge base that is decisive or to add a rule to make it decisive. For example one could add rule

$$\mathbf{s_4} \wedge \mathbf{s_5} \rightarrow \mathbf{h_2}$$

This gives an explicit definition for $\mathbf{h_2}$ in the form that

$$\mathbf{N} \models h_2 \Leftrightarrow N \models (\mathbf{s_1} \wedge \neg \mathbf{s_2} \wedge \mathbf{s_4}) \vee (\mathbf{s_4} \wedge \mathbf{s_5})$$

but it is up to the expert or knowledge engineer to determine which observables are to be used for the new rule; $\mathbf{s_4} \wedge \mathbf{s_5} \rightarrow \mathbf{h_2}$ is just a proposal.

A domain can be tested for the possibility of a decisive knowledge base by looking at implicit definability. If implicit definability does not hold then the system cannot be expanded in order to make it decisive.

For example, if $\mathbf{N_5}$ **: < 1, 0, 1, 0, 0; 1, 1 >** is added to the world of models then it is easy to see that implicit definability does not hold:

$$\mathbf{In(N_1)} = \mathbf{In(N_5)} \text{ and } \mathbf{Out(N_1)} \neq \mathbf{Out(N_5)}$$

The system now needs extra observables to distinguish between the two models, so no matter what a decisive knowledge base will not exist.

Note that checking decisiveness can be reduced to checking implicit definability and then checking weak completeness. This is because of Theorem 5.6, and the fact that both definitions use complete models.

6 Strong Completeness and Well-informedness

The previous two notions, weak completeness and decisiveness, are both converses of the notions of weakly sound respectively correct. The remaining case, strong completeness, is addressed in this section.

Intuitively, a knowledge base should obey the desirable property that it uses as little (input) information as possible. Having too many inputs, as well as having too many rules negatively influences the performance of a knowledge base. But there are several ways to violate this property. On the one hand the knowledge base contains superfluous rules (redundancy or subsumedness), but the inference can still be correct and decisive. On the

other hand, having an inadequate collection of rules or conditions may lead to undesired behaviour, where unnecessary input is asked for.

6.1 Well-informedness

In this section the concept well-informedness is introduced. In the literature one sometimes speaks of unnecessary if-conditions, and Nguyen [23], [24] treats these as problems of consistency. We believe, however, that the subject is broader, and for this reason a section will be devoted to the subject of well-informedness.

Before giving a formal semantical definition, we will describe the notion well-informedness intuitively. In general, we would like a knowledge base to obey the property that it uses as little (input) information as possible. Having too many inputs, as well as having too many rules negatively influences the performance of a knowledge base. An occurrence of this is often referred to in the literature as unnecessary if-conditions. But there are several ways to violate this property.

On the one hand, a knowledge base can contain superfluous rules (redundancy or subsumedness, see Section 6.3). This may not be a problem logically (i.e. the inference relation may be logically sound and complete), but it may be a problem in practice if input is asked for unnecessarily. One way to capture this is defined below : well-informedness.

A very simple example of a knowledge base, not satisfying well-informedness is:

$$\mathbf{p} \rightarrow \mathbf{q}$$
$$\neg\, \mathbf{p} \rightarrow \mathbf{q}$$

This small knowledge base will not able to derive $\mathbf{q}$ unless something is known about $\mathbf{p}$. If we allow partial (input) information states, where an atom may be unknown, this is a problem. To be well-informed, the knowledge base should just consist of the general fact $\mathbf{q}$ that allows a system to derive $\mathbf{q}$ even when $\mathbf{p}$ is unknown.

Well-informedness is akin to the following property of monotonic inference relations. This is a natural result of monotonicity, because it says that if some model allows some inference, then all refinements of this model allow it too.

Proposition 6.1

Suppose a set of complete models $\mathbf{V}$ is given and the inference relation for $\mathbf{KB}$ is monotonic.

If $\mathbf{M}$ is any partial model in $\mathbf{P(V)}$ and $\mathbf{h} \in \mathbf{OutLit}(\Sigma)$ can be derived from $\mathbf{M}$, then $\mathbf{h}$ can be derived from all refinements of $\mathbf{M}$ in $\mathbf{V}$. More formally,

$$\forall \mathbf{M} \in \mathbf{P(V)} \quad \mathbf{M} \vdash_{\mathbf{KB}} h \quad \Rightarrow \quad \forall \mathbf{N} \in \mathbf{V} \; [\, \mathbf{N} \geq \mathbf{M} \Rightarrow \mathbf{N} \vdash_{\mathbf{KB}} \mathbf{h} \,]$$

The proof runs as follows.
Given $\mathbf{M}$ such that $\mathbf{M} \vdash_{\mathbf{KB}} \mathbf{h}$, consider an arbitrary complete refinement $\mathbf{N} \in \mathbf{W_{in}}$ with $\mathbf{N} \geq \mathbf{M}$. By monotonicity we know that also $\mathbf{N} \vdash_{\mathbf{KB}} \mathbf{h}$. As we have taken an arbitrary $\mathbf{N}$, this holds for all $\mathbf{N} \in \mathbf{W_{in}}$ with $\mathbf{N} \geq \mathbf{M}$ and the theorem is proven.

Well-informedness is in fact the converse of this proposition, as defined below.

Definition 6.2

Suppose a set of complete input models $\mathbf{S}$ is given. A knowledge base $\mathbf{KB}$ is *well-informed* with respect to $\mathbf{S}$ if for any partial model $\mathbf{M}$ in $\mathbf{P(S)}$ any $\mathbf{h} \in \mathbf{OutLit}(\Sigma)$ can be derived from the model $\mathbf{M}$ if it can be derived from all refinements $\mathbf{N}$ in $\mathbf{S}$ of $\mathbf{M}$. In other words, if for all $\mathbf{M} \in \mathrm{P(S)}$ the following holds:

$$\{\ \forall \mathbf{N} \in \mathbf{S}\ [\ \mathbf{N} \geq \mathbf{M} \Rightarrow \mathbf{N} \vdash_{\mathbf{KB}} \mathbf{h}\ ]\ \} \Rightarrow \quad \mathbf{M} \vdash_{\mathbf{KB}} \mathbf{h}.$$

Unlike monotonicity that depends on the inference relation, well-informedness generally depends on the contents of the knowledge base. Therefore, a verifier can aid the knowledge engineer by checking this.

There is also a more principled reason for wanting a knowledge base to be well-informed with respect to an inference relation. It can be proven that the forcing relation itself obeys a condition similar to well-informedness, a is shown in the following theorem.

Theorem 6.3

Assume $\mathbf{W}$ is a domain description. For any partial model $\mathbf{M} \in \mathbf{P(In(W))}$ such that all its complete refinement inputs force $\mathbf{h}$, the model $\mathbf{M}$ itself forces $\mathbf{h}$, i.e., for any $\mathbf{M} \in \mathbf{P(In(W))}$ it holds

$$\forall \mathbf{N} \in \mathbf{W}\ [\mathbf{In(N)} \geq \mathbf{M} \Rightarrow \mathbf{In(N)} \models_{\mathbf{W}} \mathbf{h}\ ] \quad \Rightarrow \quad \mathbf{M} \models_{\mathbf{W}} \mathbf{h}$$

The proof runs as follows. Given $\mathbf{M} \in \mathbf{P(In(W))}$, consider an arbitrary complete refinement $\mathbf{N} \in \mathbf{W}$ such that $\mathrm{In}(\mathbf{N}) \geq \mathbf{M}$. By the premise it follows that $\mathbf{In(N)} \models_{\mathbf{W}} \mathbf{h}$ and Lemma 2.12 then implies $\mathbf{N} \models \mathbf{h}$. As we have taken an arbitrary $\mathbf{N}$ we can conclude for all $\mathbf{N} \in \mathbf{W}$ that $\mathbf{N} \geq \mathbf{M} \Rightarrow \mathbf{N} \models \mathbf{h}$. This is precisely the definition of $\mathbf{M} \models_{\mathbf{W}} \mathbf{h}$; that proves the theorem.

6.2 Strong Completeness

In the previous section, we saw that the notions of weak completeness and decisiveness are precisely distinguished by the notion of empirical foundedness. A nice property of well-informedness is that it precisely distinguishes strong completeness from weak completeness.

Definition 6.4

Assume a domain description **W** with respect to a signature Σ is given. The knowledge base **KB** is *strongly complete* with respect to **W** if for all output literals $\mathbf{h} \in \mathbf{OutLit}(\Sigma)$ the following holds:

$$\forall \mathbf{M} \in \mathbf{P(W)} \quad [\ \mathbf{In(M)} \models_{\mathbf{W}} \mathbf{h} \quad \Rightarrow \quad \mathbf{In(M)} \vdash_{\mathbf{KB}} \mathbf{h}\ ]$$

Unlike the converse implications (correct, weakly sound, and strongly sound), the completeness notions are not equivalent. Indeed, the differentiating property between strongly complete and weakly complete is precisely well-informedness (along with monotonicity) as the following theorem shows (for a proof, see the Appendix):

Theorem 6.5

Assume a domain description **W**, and a knowledge base **KB** with respect to a signature Σ. If **KB** is sound with respect to **W**, then the following notions are equivalent:

(i) **KB** is strongly complete with respect to **W**

(ii) **KB** is weakly complete with respect to **W** and well-informed with respect to **In(W)**.

6.3 Verifying Strong Completeness

In the literature a number of specific notions are used that relate to strong completeness. To relate to this literature, in this section we will discuss some of the configurations of rules that allow rules to be removed and are useful for a verifier. Notice especially the discussion on well-informedness, since this is a fundamental concept in our semantical analysis.

Circularity

A set of rules is circular if the chaining of those rules in the set forms a cycle. Circular rules can cause an infinite loop with backward chaining. To prevent getting lost in such a loop, loop checks are needed during inference that may be quite computationally expensive.

We consider two cases:

1. *self-referent rule*:
This is a rule in which the conclusion is a condition from its antecedent.
For example: $\mathbf{p} \wedge \mathbf{q} \rightarrow \mathbf{q}$, or even simpler $\mathbf{p} \rightarrow \mathbf{p}$.

2. *self-referent chain of inferences*:
Same as 1. but the self reference occurs after some inferences.
For example: $\mathbf{r} \wedge \mathbf{s} \rightarrow \mathbf{p}$ and $\mathbf{p} \wedge \mathbf{q} \rightarrow \mathbf{r}$. Here **p** does not occur in the conditions of

$\mathbf{r} \wedge \mathbf{s} \rightarrow \mathbf{p}$, but does occur in the conditions of the rule with conclusion $\mathbf{r}$. So a program can look for a rule with $\mathbf{r}$ or $\mathbf{s}$ as its conclusion and then checks to see if $\mathbf{p}$ is a member of the conditions of that rule. If yes, then a cycle is found. If not, try to go further back.

Redundancy

A rule is redundant if it fires in the same situation as another one and has the same results. This means that the antecedents of the two rules are equivalent, as are the conclusions. The antecedents of the two rules can only be equivalent if they have the same number of conditions and every condition in one rule has an equivalent condition in the other rule.

We consider two cases:

1. *redundancy in rule pairs*:
   Identical rules, $\mathbf{p} \rightarrow \mathbf{r}$ and $\mathbf{p} \rightarrow \mathbf{r}$ are a form of redundancy.
   It is obvious that removing one of the identical rules will not influence the inference process.

2. *redundancy in chained inference*:
   The chains $\mathbf{p} \rightarrow \mathbf{q}$; $\mathbf{q} \rightarrow \mathbf{r}$ and $\mathbf{p} \rightarrow \mathbf{r}$, where $\mathbf{q}$ cannot be derived through other rules, are an example of redundancy. Because **q** cannot be derived through other rules, either rule $\mathbf{q} \rightarrow \mathbf{r}$ or $\mathbf{p} \rightarrow \mathbf{r}$ can be deleted.

Redundant rules have no influence on the system's results, but do affect the system's performance. Also, inconsistencies are more likely to arise if one of the redundant rules is changed or deleted, whereas the other rule is not changed (this is not the knowledge engineer's intention; why else would he or she have changed the first rule?).

Subsumedness

A rule is subsumed by another if the two rules have the same conclusions, but one rule contains additional constraints on the situation in which it will succeed. This rule has more conditions to fulfil, so it is redundant.

Two cases:

1. *subsumedness in rule-pairs*:

$$\mathbf{p} \wedge \mathbf{q} \rightarrow \mathbf{r}$$
$$\mathbf{p} \rightarrow \mathbf{r}$$

It is obvious that if the first rule succeeds then the second, more general, rule does as well, resulting in redundancy. In this case one could delete the more specific rule $\mathbf{p \wedge q \rightarrow r}$.

2. *subsumedness in chained inference*:
The chain $\mathbf{p \wedge q \rightarrow s\ ;\ s \rightarrow r}$ is subsumed by the rule $\mathbf{p \rightarrow r}$. One could delete $\mathbf{s \rightarrow r}$ if there are no other rules with $\mathbf{s}$ as conclusion.

Self-contradiction

A rule contains a self contradiction if it contains a formula that can never be true. The same can be said for rule chains. Obviously some part of the rule or chain may then be removed.

1. *self-contradicting antecedents:*
This is a rule that has the negation of one of its conditions in its antecedents. For example, $\mathbf{p \wedge q \wedge \neg}$ p $\rightarrow \mathbf{r}$. This rule clearly has no chance of succeeding.

2. *self-contradicting antecedents in chaining:*
Same as above, but the contradiction only occurs until after some inferences. Consider the chain e.g. $\neg$ p $\rightarrow \mathbf{q;\ q \wedge p \rightarrow t.}$

For contradicting antecedents there is no difference whether a domain description is taken into account or not. For the rule $\mathbf{p \wedge q \wedge \neg}$ p $\rightarrow \mathbf{r}$, for example, to be used the conditions $\mathbf{p}$ and $\mathbf{\neg p}$ should be true in a model at the same time. So, if this rule is to be used in a reasoning session, we need to derive a contradiction first. Therefore contradicting antecedents are always reported, whether checking was done with or without a domain description. One could as well argue that these errors fall under redundancy, but redundancy errors are of a non harmful kind, and these errors are definitely harmful. Rules like these have no chance of succeeding, so it could very well be that the hypothesis concluded by such a rule becomes unreachable (which in turn could imply incompleteness).

Well-informedness

The following algorithm can be applied for checking well-informedness. Let any set of complete input models $\mathbf{W_{in}}$ be given. Take a partial model $\mathbf{M \in P(W_{in})}$. From this $\mathbf{M}$ take all complete refinements within $\mathbf{W_{in}}$, and for each one see what can be derived from it via $\mathbf{KB}$. Take from these derivations the greatest common information state, i.e. the conclusions that all refinements agree on. This greatest common information state must equal that what can be derived from the partial model $\mathbf{M}$ via $\mathbf{KB}$. If it is equal, then well-informedness holds. To make it clearer, consider the following example. Assume we are given a signature

$\Sigma = < \mathbf{s_1, s_2, s_3, s_4; h_1, h_2, h_3} >$

and a partial input model $\mathbf{M} = < \mathbf{1, 1, u, u} >$. So, only something is known about the first two observables. Consider the following knowledge base:

$$
\begin{aligned}
s_1 \wedge s_2 &\rightarrow h_1 \\
s_3 &\rightarrow h_2 \\
\neg s_3 &\rightarrow \neg h_2 \\
s_4 &\rightarrow h_3 \\
\neg s_4 &\rightarrow \neg h_3
\end{aligned}
$$

Now, the complete refinements of $\mathbf{M} = < \mathbf{1, 1, u, u} >$ are:

< 1, 1, 1, 1 >
< 1, 1, 1, 0 >
< 1, 1, 0, 1 >
< 1, 1, 0, 0 >

From these the following can be derived:

< 1, 1, 1, 1; 1, 1, 1 >
< 1, 1, 1, 0; 1, 1, 0 >
< 1, 1, 0, 1; 1, 0, 1 >
< 1, 1, 0, 0; 1, 0, 0 >

The greatest common information state has truth value **1** for $\mathbf{h_1}$. All four refinements agree on this hypothesis. Only the first two complete models agree on $\mathbf{h_2}$, the last two do not agree with the first two models. So the greatest common information state of $\mathbf{h_2}$ is **u** (unknown). In the same way is the greatest common information state of $\mathbf{h_3}$ unknown as well. So, the greatest common information state of all complete output models is $< \mathbf{1, u, u} >$. Now, let's look at what can be derived from the partial model **M**. It turns out that from **M** exactly the same, i.e. **< 1, u, u >**, can be derived. All partial models have to be checked in this way before we can conclude that well-informedness applies.

As one can imagine, checking well-informedness in this way can take a lot of time if there are many observables, and if the knowledge base **KB** is very large. So, in the literature verifying well-informedness is often limited to checking the rules in the knowledge base for contradicting antecedents in rules. For well-informedness the domain description can be restricted to input models only; for simplicity it is assumed to be the set of all possible truth

assignments to the input atoms. When applying the previous to the **KB** only, checking for well-informedness results in checking for contradicting antecedents. Rules are checked for well-informedness by checking if they contain unnecessary conditions. Two rules contain unnecessary conditions if the rules have the same conclusion, a condition in one rule's antecedent is in conflict with a condition in the other rule's antecedent and all the other conditions they have are equivalent. We distinguish four cases:

1. *Lack of well-informedness with redundancy in rule pairs*:

$\mathbf{p} \wedge \mathbf{q} \rightarrow \mathbf{r}$

$\mathbf{p} \wedge \neg \mathbf{q} \rightarrow \mathbf{r}$

It is obvious that the condition with predicate **q** is unnecessary in both rules. When **p** is true then **r** is true, because it does not matter whether **q** is true or not. The two rules can be merged into one rule, namely $\mathbf{p} \rightarrow \mathbf{r}$. If one did not change the rules then **r** could not be inferred if **q** was unknown.

2. *Lack of well-informedness with subsumedness in rule pairs*:

$\mathbf{p} \wedge \mathbf{q} \rightarrow \mathbf{r}$

$\neg \mathbf{q} \rightarrow \mathbf{r}$

The rules have equivalent conclusions and a condition in conflict, but the first rule contains more conditions. It is subsumed by the second rule (considering only the conditions not in conflict). In this case, the second condition in the first rule is unnecessary, but both rules are still needed. They can be reduced to the following two rules :

$\mathbf{p} \rightarrow \mathbf{r}$

$\neg \mathbf{q} \rightarrow \mathbf{r}$

3. *Lack of well-informedness with redundancy in chains*:

Two chains produce a condition in conflict and the other conditions in the chains are redundant. For example:

$\mathbf{p} \wedge \neg \mathbf{q} \rightarrow \mathbf{s}$

$\mathbf{p} \wedge \mathbf{q} \rightarrow \mathbf{r}$; $\mathbf{r} \rightarrow \mathbf{s}$

In the rules above, condition **q** is in conflict. One would expect that, according to case 1, the rule and the chain reduce to:

$\mathbf{p} \rightarrow \mathbf{s}$

$\mathbf{p} \rightarrow \mathbf{r}$; $\mathbf{r} \rightarrow \mathbf{s}$

But it could be that, to conclude **r**, the literal **q** is needed as an extra restriction, so it is not at all obvious that one can make this adjustment. A solution is:

$\mathbf{p} \rightarrow \mathbf{s}$

$\mathbf{p} \wedge \mathbf{q} \rightarrow \mathbf{r}$; $\mathbf{r} \rightarrow \mathbf{s}$

That is, only change the first rule, which is not a chain and leave the chain intact.

Consider the following chains as well, where the first chain consists of rules:

$\neg\, \mathbf{p} \rightarrow \mathbf{s}$; $\mathbf{s} \wedge \mathbf{q} \rightarrow \mathbf{r}$

and the second chain consists of:

$\mathbf{p} \wedge \mathbf{q} \rightarrow \mathbf{t}$; $\mathbf{t} \rightarrow \mathbf{r}$

As one can see condition **p** is in conflict. The problem is, what to do now. A possible solution would be to change the first chain into:

$\neg\, \mathbf{p} \rightarrow \mathbf{s}$; $\mathbf{q} \rightarrow \mathbf{r}$

and leave the second chain intact.

4. *Lack of well-informedness with subsumedness in chaining*:

Two chains produce a condition in conflict and the set of other conditions if the first chain is a subset of the other chains conditions. Consider the following two chains:

$\neg\, \mathbf{p} \rightarrow \mathbf{s}$; $\mathbf{s} \rightarrow \mathbf{r}$

$\mathbf{p} \wedge \mathbf{q} \rightarrow \mathbf{t}$; $\mathbf{t} \rightarrow \mathbf{r}$

Condition $\mathbf{p}$ is in conflict, but what to do now. One would expect (as in case 2) the solution:

$$\neg\ \mathbf{p} \rightarrow \mathbf{s}\ ;\ \mathbf{s} \rightarrow \mathbf{r}$$
$$\mathbf{q} \rightarrow \mathbf{t}\ ;\ \mathbf{t} \rightarrow \mathbf{r}$$

But again $\mathbf{t}$ may need $\mathbf{p}$ in order to be properly inferred. So in this case nothing can be deleted. One thing one can do is add the rule $\mathbf{q} \rightarrow \mathbf{r}$. This would solve well-informedness by enlarging the knowledge base. This shows that making a knowledge base well-informed not necessarily simplifies it.
Another case is when the second chain is:

$$\mathbf{q} \rightarrow \mathbf{t}\ ;\ \mathbf{t} \wedge \mathbf{p} \rightarrow \mathbf{r}$$

Then it is possible to delete $\mathbf{p}$ and make the chain:

$$\mathbf{q} \rightarrow \mathbf{t}\ ;\ \mathbf{t} \rightarrow \mathbf{r}$$

7 Knowledge Verification in Compositional Knowledge-Based Systems

The above considerations are directed to knowledge-based systems consisting of only one knowledge base, or to a component of a more complex system. The case of a knowledge-based system with a compositional structure is the more relevant case in practical applications. In such a system the inputs for one component are determined by the other components (it is possible to consider the outside world as a component also); see [2] for an overview of the principles behind compositional design of knowledge-based agent systems.

The semantical framework introduced here can be applied to verify properties of any knowledge base, and in particular for knowledge bases in primitive components of a compositional knowledge-based system. Within this section we will *not* present an approach to verification of compositional knowledge-based systems, since this can be found elsewhere; e.g., [4], [11]. However, what we *will* do in this section is show how the semantical verification approach of the current paper connects to compositional verification, i.e., how it can be used in conjunction with compositional verification of knowledge-based systems. Compositional verification defines how properties of a composed component relate to properties of the sub-components. At the lowest level in the composition hierarchy primitive components are specified by a knowledge base.

In this subsection it is briefly shown how the semantical framework introduced in this paper can be applied in conjunction with compositional verification of an example compositional process model for diagnostic reasoning (see [4]) . The processes at different

abstraction levels of this generic diagnostic model are given in Fig. 2. The primitive component Hypothesis Determination generates hypotheses that are validated by the component Hypothesis Validation. The latter component is not primitive: it is composed of the primitive components Observation Determination, Observation Execution, and Hypothesis Evaluation.

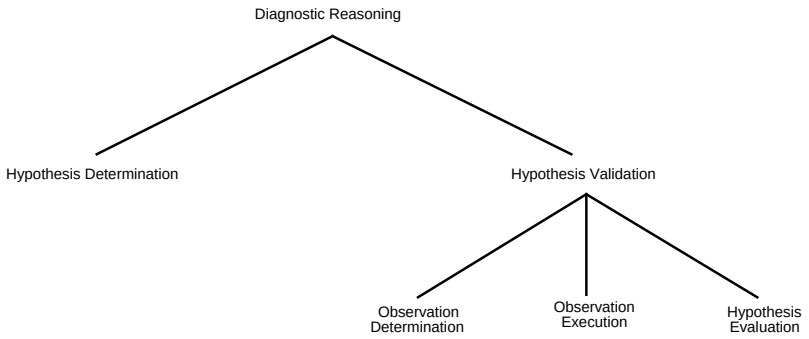

Fig. 2 Processes at different abstraction levels in the diagnostic process model

In [4] it is shown how (dynamic) properties of the process model as a whole can be reduced to properties of Hypothesis Determination, and how properties of Hypothesis Validation can be reduced to properties of Observation Determination, Observation Execution, and Hypothesis Evaluation. Since the primitive components were left open in this generic model, here the story ends in [4]. To apply the generic model, instantiations of these primitive components are needed, for example selected from a library of components and using specific knowledge bases. Using the approach of [4], the model as a whole is only guaranteed to work properly if the required properties of the primitive components presented in [4] are satisfied by these primitive components. In this section we show how for some of the primitive components these properties can be formulated as constraints on a related domain description.

To start, the component Hypothesis Determination should satisfy ‘focus efficiency’ and ‘focus effectiveness’. Focus efficiency means that no hypotheses are chosen in focus that already have been assessed. In the temporal language used in [4] this is expressed in the following form. For all traces, at all time points if at the input the information is available that some hypothesis h already was assessed, then it will not be at the output that it is in focus:

$\forall \mathcal{M} \in$ **Traces(HD)** $\forall$ **t** $\forall$ **h**

[state$_{HD}(\mathcal{M}$**, t, input(HD)) |= assessed(h)** $\Rightarrow$ **state**$_{HD}(\mathcal{M}$**, t, output(HD)) |≠ focus(h)]**

The domain description **W(HD)** is defined by some constraints (related to the required properties) on the set of complete models **CMod(**Σ**(HD))**, where Σ**(HD)** is the combination of input and output signature of the component Hypothesis Determination. The constraint **C1** related to the above property is expressed as follows:

$\bigwedge_{\mathbf{h}}$ $\neg$ **(assessed(h)** $\wedge$ **focus(h))**

Here $\bigwedge_{\mathbf{h}}$ means taking the conjunction over all hypotheses **h**; similarly $\bigvee_{\mathbf{h}}$ denotes taking the disjunction. The second property to be satisfied by Hypothesis Determination is focus effectiveness; this means that as long as not all hypotheses have been assessed, and no hypothesis has been confirmed, there will be generated focus hypotheses. In the temporal language of [4] this is expressed as follows. For all traces and time points, if there exists at least one hypothesis for which no information is at the input that it was assessed, and for no hypothesis there is information on the input that it was confirmed, then there exists at least one hypothesis such that on the output there is information that it is in focus:

$\forall \mathcal{M} \in$ **Traces(HD)** $\forall$ **t**

[$\exists$**h state**$_{HD}(\mathcal{M}$**, t, input(HD)) |≠ assessed(h)** $\wedge$ $\forall$**h state**$_{S}(\mathcal{M}$**, t, input(HD)) |≠ confirmed(h)]**

$\Rightarrow$ **[** $\exists$**h' state**$_{HD}(\mathcal{M}$**, t, output(HD)) |= focus(h')]**

This property can be reformulated to the following defining constraint **C2** for **W(HD)**:

$\bigwedge_{\mathbf{h}}$ **assessed(h)** $\vee$ $\bigvee_{\mathbf{h}}$ **confirmed(h)** $\vee$ $\bigvee_{\mathbf{h}}$ **focus(h)**

In conclusion, a component can safely be chosen to play the role of Hypothesis Determination in the diagnostic model if it is sound and strongly complete with respect to the domain description **W(HD)** defined by the two constraints above (i.e., the set of complete models satisfying the constraints), related to the temporal properties, i.e.,

W(HD) = { M $\in$ **CMod(**Σ**(HD))** | **M |= C1** $\wedge$ **C2 }**

In a similar manner the properties 'observation effectiveness' and 'observation efficiency' of the component Observation Determination reduce to constraints (on the set of complete models) defining domain description **W(OD)**. Observation efficiency means that no observations are generated that already were performed:

$\forall \mathcal{M} \in$ **Traces(OD)** $\forall$ **t** $\forall$ **o**

[state$_{OD}(\mathcal{M}$**, t, input(OD)) |= observed(o)** $\Rightarrow$ **state**$_{OD}(\mathcal{M}$**, t, output(OD)) |≠ to_be_observed(o)]**

This is reformulated as the following defining constraint **C3** for **W(OD)**:

$\bigwedge_{\mathbf{o}}$ $\neg$ **(observed(o)** $\wedge$ **to_be_observed(o))**

Observation effectiveness means that if there exists at least one hypothesis in focus, and not all observations have been performed, then at least one observation is generated.

$$\forall \mathfrak{M} \in \mathbf{Traces(OD)}\ \forall t\ \forall h$$

$$[\exists o\ \ state_{OD}(\mathfrak{M}, t, input(OD)) \models \neq observed(o)$$

$$\wedge\ state_{OD}(\mathfrak{M}, t, input(OD)) \models focus(h) \Rightarrow$$

$$\exists o'\ state_{OD}(\mathfrak{M}, t, output(OD)) \models to\_be\_observed(o')\ ]$$

This is reformulated as the following defining constraint **C4** for **W(OD)**:

$$\bigwedge_h\ \neg\ focus(h)\ \ \vee\ \bigwedge_o\ observed(o) \vee\ \bigvee_o\ to\_be\_observed(o)$$

Again, a proper choice for an instantiation of component Observation Determination is made if it satisfies soundness and strong completeness with respect to domain description **W(OD)** defined by the two constraints:

$$W(OD) = \{\ M \in \mathbf{CMod}(\Sigma(OD)) \mid M \models C3 \wedge C4\ \}$$

One of the required properties of Hypothesis Evaluation is assessment decisiveness, which means that if for all possible observations, observation results have been input, then for every hypothesis an assessment can be derived:

$$\forall \mathfrak{M} \in \mathbf{Traces(HE)}\ \forall t$$

$$[\forall o\ [\ state_{HE}(\mathfrak{M}, t, input(HE)) \models o\ \vee\ state_{HE}(\mathfrak{M}, t, input(HE)) \models \neg\ o\ ] \Rightarrow$$

$$\forall h\ [\ state_{HE}(\mathfrak{M}, t, output(HE)) \models h\ \vee\ state_{HE}(\mathfrak{M}, t, output(HE)) \models \neg\ h\ ]$$

This can be reformulated into the property ‘empirically foundedness’ of the domain description **w** describing the world situations to which the system is applied.

This shows by an example (based on [4]) how in addition to a compositional verification process, required properties of candidate primitive components for a generic model can be formulated as properties of their domain description.

8 Discussion

In the literature various approaches to verification of knowledge-based systems can be found. Most of these approaches have a rather practical nature, and seem sometimes ad hoc in the definitions of the properties that are tested. Often only partial tests are carried out. It turns out that a more principled investigation from a formal point of view reveals some restrictions of the available approaches. First we restrict ourselves to verification of systems that consist of one knowledge base (or to just one component of a more complex system). For this case the collection of formal notions as introduced in [35] and extended and developed further in the

current paper provides a useful framework for a more principled analysis. Central in this framework is the semantical notion of forcing, that is useful as a standard for the inferences.

The property on which verification has been developed best in the literature is consistency. A precise common definition is available of the notion, and tests are described for it. However, even on this notion points of discussion are possible. The first point is whether all possible sets of input facts for the system (or component) are taken into consideration. If not, then it may be the case that inconsistencies are detected that will never cause a problem in practice, because the underlying combination of inputs will never occur. Resolving this type of inconsistencies may lead to a knowledge base that is more complex than needed. If these harmless inconsistencies should be distinguished from the harmful ones, a set of input situations is needed. This can easily lead to a problem in practice, because it may not be known precisely which input situations do occur (and there may be many of them). In such a case often working with a set of sample situations is possible. But then testing on them does not guarantee consistency with respect to the other input situations. Another option is to define the set of possible input situations by a set of constraints. It depends on whether in practice these constraints are easy or difficult to acquire. So, in testing consistency there exists not one absolutely ideal approach. The kind of certainty obtained is relative with respect to the defined set of situations. This is not a drawback of specifically our method of analysis but more a characteristic of the nature of the problem of verification of interactive knowledge-based systems. Every serious verification method for interactive knowledge-based systems encounters this issue. Our semantic framework offers means to reveal and explain the issue in (formal) detail.

A property that comes next is correctness, or soundness, for which different formal semantical definitions can be given (we distinguish correctness, weak soundness, and strong soundness). In the current paper we prove that, assuming monotonicity of the inference relation, they are semantically equivalent. So, it does not matter on which one a test is carried out: one can choose the one that is computationally the least expensive. Of these three, correctness is the easiest to check, because it only considers the truth of an hypothesis in a complete situation model in the domain description. For example, in our own program DES-CHECK knowledge base expressions are exploited to do this. The notions of (weak and strong) soundness which use the more complex forcing from partial models, are then automatically satisfied.

In this case the comparison with the situations occurring in the world (concerning both input and output facts this time) can not be avoided as it can in the case of consistency. Here essentially a test set of world situations is needed, and correctness cannot be guaranteed beyond this set.

The most complex notion turns out to be completeness. It seems very hard to find in the literature on verification any formal definition of this notion. Testing on completeness is often

done in an ad hoc manner, based on an informal definition. In the current paper we have presented formal definitions for three semantically non-equivalent variants of completeness. Moreover, we have analysed that the difference between them depends on two underlying concepts: empirically foundedness of a domain, and well-informedness of a knowledge base. These underlying concepts can be tested separately. The outcome of such tests give more detailed information on the causes of a lack of completeness. Of course also in this case the issue of the set of sample situations plays an important role.

Checking decisiveness is not too difficult as it considers the truth relation in complete models of the domain description. In the program DES-CHECK this is done by the knowledge base expressions. Because we have already shown soundness, we have then automatically have shown weak soundness and empirical foundedness. Another straightforward possibility is to first check the three equivalent notions, implicit definability, explicit definability, and empirical foundedness. The first of these is easiest as it can be determined by pair-wise comparison of the models in the domain description. One can check implicit definability to first determine empirical foundedness.

The remaining notion of strong completeness is more difficult to check. The more traditional approach taken in the literature of (pair-wise) rule comparison is an incomplete but reasonably simple possibility.

It turns out that our semantical framework provides adequate logical descriptions for the functionality of an interactive knowledge-based reasoning system. In particular the relation of the conclusions that may be drawn by a system and the situation in reality that is concerned can be made transparent by our framework.

The framework assumes at crucial places that inferencing is monotonic. The area of verification of nonmonotonic knowledge bases (e.g., [0]) is an area that requires a treatment by itself, since a number of new issues arise if the monotonicity assumption is not made. A challenge for further research is to investigate the possibility for a semantic framework addressing the key issues of verification of nonmonotonic knowledge bases, comparable to the semantical framework for the monotonic case presented above.

In comparison, some approaches (like COVADIS [30]) 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) in our approach. These models 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, but it is impossible to guarantee that a knowledge base is completely debugged.

Recently a number of approaches to verification of knowledge-based systems have been developed that take into account some form of structuring of the knowledge base or the task structure of the system; e.g., [4], [7], [8], [11], [20], [27]. In general, these approaches make

use of knowledge bases as building blocks and concentrate on the way in which they are processed in combination to achieve the system's task. An interesting area of further research is to investigate how the semantical approach to knowledge base verification can be exploited in these higher order methods. In Section 7 above, for the compositional approach exploited in [4] and [11], such a relationship has been pointed out.

Acknowledgements

Catholijn Jonker, Rineke Verbrugge and Mehdi Dastani have contributed improvements in the text.

References

[0] G. Antoniou, Verification and Correctness Issues for Nonmonotonic Knowledge Bases. *International Journal of Intelligent Systems* 12,10 (1997): 725-738

[1] M. Ayel, Protocols for consistency checking in expert systems knowledge bases: SACCO, These d'etat, Chambery, Sept. 1987.

[2] F.M.T. Brazier, C.M. Jonker, and J. Treur, Principles of Compositional Multi-agent System Development. In: J. Cuena (ed.), *Proceedings of the 15th IFIP World Computer Congress, WCC'98, Conference on Information Technology and Knowledge Systems, IT&KNOWS'98*, 1998, pp. 347-360. To be published by IOS Press, 2001.

[3] C.C. Chang, H.J. Keisler, *Model theory*, North Holland, 1973.

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

[5] B.J. Cragun, H.J. Steudel, A decision-table-based processor for checking completeness and consistency in rule based expert systems, *Int. J. Man-Machine Studies*, 26(5), 1987, p 633-648.

[6] S. Craw, Judging knowledge base quality. In: *Validation, Verification and test of knowledge-based systems*, by M. Ayel and J-P. Laurent (Eds.), Wiley & Sons, 1991, pp. 207-219

[7] D. Fensel and A. Schonegge. Inverse verification of problem-solving methods. *International Journal of Human-Computer Studies*, 49:4, 1998.

[8] P. Groot and A. ten Teije and F. van Harmelen, Formally verifying dynamic properties of KBS. In: D. Fensel and R. Studer (eds.), *Proceedings of the 11th European Workshop on Knowledge Acquisition, Modeling, and Management, EKAW'99*, Lecture Notes in Artificial Intelligence, vol. 1621, Springer-Verlag, 1999, 157-172.

[9] 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 (Editors), Wiley & Sons, 1991, Chapter 8.

[10] C.G. Hempel, *Philosophy of Science*, Prentice-Hall, Englewoods Cliffs, 1966

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

1536, Springer Verlag, 1998, pp. 350-380. Extended version in: *International Journal of Cooperative Information Systems.* In press, 2001.

[12] P.H.G. van Langen, J. Treur, *Representing world situations and information states by many-sorted partial models*, Report PE8904, Department of Mathematics and Computer Science, University of Amsterdam, 1989

[13] T. Langholm, Partiality, Truth and Persistence, CSLI Lecture Notes No. 15, Stanford University, Stanford, 1988.

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

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

[16] A. Ligeza, Completeness verification of rule-based control systems. *SAMS, Systems Analysis -- Modelling -- Simulation* (Int. Journal), Vol. 24, 1996, 211-220.

[17] A.D. Lunardhi, and Passino, K.M., Verification of Qualitative properties of rule-based expert systems. *Applied AI Journal*, vol. 9, 1995, pp. 587-621

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

[19] P. Meseguer, Incremental Verification of Rule-Based Expert Systems. In: B. Neumann (ed.), *Proc. of the 10th European Conference on Artificial Intelligence, ECAI'92*, Wiley and Sons, 1992, pp. 840-844.

[20] P. Meseguer, and Verdaguer, A., Verification of Multi-level rule-based expert systems: theory and practice. *Int. J. of Expert Systems*, vol. 6, 1993, pp. 163-192.

[21] 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., Oct. 1986, p 30-35.

[22] D.L. Nazareth, Issues in the verification of knowledge in rule-based systems, *Int. J. Man-Machine Studies* (1989), 30, p 255-271.

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

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

[25] W.A. Perkins, T.J. Laffey, D. Pecora, T.A. Nguyen, Knowledge base verification, *AI Magazine*, 8(2), p 69-75 (1987).

[26] A.D., Preece, A new approach to detecting missing knowledge in expert system rule bases. *Int. J. of Man-Machine Studies*, vol. 38, 1993, pp. 161-181.

[27] A.D. Preece, Grossner, C., Chander, P.G., and Radhakrishnan, Structure-based validation of rule-based systems. *Data and Knowledge Engineering*, vol. 26, 1998, pp. 161-189.

[28] A.D. Preece, R. Shinghal, Verifying Knowledge Bases by Anomaly Detection: An Experience Report, *Proc. of ECAI, 1992*, p. 835-839.

[29] A. Preece, S Talbot and L Vignollet. Evaluation of Verification Tools for Knowledge-Based Systems. *International Journal of Human-Computer Studies*, 47, 629-658, 1997.

[30] M.-C. Rousset, On the consistency of knowledge bases: the COVADIS system. *Computational Intelligence*, vol. 4, 1988, pp. 166-170. Also in: *Proc. of ECAI-88*, Munich, Germany, 1988, Pitman Publishing, p 79-84.

[31] M. Suwa, A. Garlisle Scott,E.H. Shortliffe,Completeness and consistency in a rule based system, In: B.G.Buchanan, E.H.Shortliffe, *Rule-based expert systems*, Addison Wesley 1985, pp. 159-170.

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

[33] J. Treur, Completeness and definability in diagnostic expert systems, *Proc. Eur. Conf. on AI, ECAI-88*, Munich, Germany, 1988, p 619-624.

[34] J. Treur, Heuristic reasoning and relative incompleteness. *International Journal of Approximate Reasoning*, vol. 8, 1993, pp. 51-87.

[35] J. Treur, Declarative functionality descriptions of interactive reasoning modules. In: H. Boley, M.M. Richter (eds.), *Processing Declarative Knowledge, Proc. of the International Workshop PDK-91*, Lecture Notes in AI, vol. 567, Springer Verlag, 1991, pp. 221-236.

[36] J. Treur, and M. Willems, A logical foundation for verification. In: A.G. Cohn (ed.), *Proc. of the 11th European Conference on Artificial Intelligence, ECAI'94*. John Wiley & Sons, Chichester, 1994, pp. 745-749.

[37] W.-T. Tsai, Vishnuvajjala, R., and Zhang, D., Verification and Validation of Knowledge-Based Systems, *IEEE Transactions on Knowledge and Data Engineering*, Vol. 11, No. 1, January/February 1999, pp. 202-212.

[38] R.W. Weyhrauch, Prolegomena to a theory of mechanized formal reasoning, *Artificial Intelligence* 13 (1980), pp. 133-170

Appendix Proofs

Theorem 3.5

Assume a domain description **W** a knowledge base **KB** with respect to a signature Σ. If the inference relation $\vdash_{\mathbf{KB}}$ is monotonic with respect to **W**, then the following statements are equivalent:

(i) **KB** is strongly sound with respect to **W**.

(ii) **KB** is weakly sound with respect to **W**.

(iii) **KB** is correct with respect to **W**.

Proof:

(i) => (ii) If **KB** is strongly sound, then for all $\mathbf{M} \in \mathbf{P(W)}$ the soundness property holds. In particular, because $W \subseteq P(W)$, the same property holds for all $\mathbf{N} \in \mathbf{W}$. Hence **KB** is weakly sound with respect to **W.**

(ii) => (iii) Assume for an arbitrary $\mathbf{N} \in \mathbf{W}$ that $\mathbf{In(N)} \vdash_{\mathbf{KB}} h$. Then by weak soundness we know that $\mathbf{In(N)} \models_{\mathbf{W}} \mathbf{h}$. And by Lemma 2.12 this implies $\mathbf{N} \models \mathbf{h}$ and hence we have correctness.

(iii) => (i) Given a partial model $\mathbf{M} \in \mathbf{P(W)}$ such that $\mathbf{In(M)} \vdash_{\mathbf{KB}} h$, we consider an arbitrary complete refinement $\mathbf{N} \in \mathbf{W}$ such that $\mathbf{In(M)} \leq \mathbf{N}$. Lemma 2.6 together with $\mathbf{In(In(M))} = \mathbf{In(M)}$ implies $\mathbf{In(M)} \leq \mathbf{In(N)}$, so from $\mathbf{In(M)} \vdash_{\mathbf{KB}} h$ by monotonicity we can conclude $\mathbf{In(N)} \vdash_{\mathbf{KB}} h$, that in turn by correctness implies $\mathbf{N} \models \mathbf{h}$. As we have taken an arbitrary **N** we can conclude that for all models $\mathbf{N} \in \mathbf{W}$, it holds $\mathbf{N} \geq \mathbf{In(M)} \Rightarrow \mathbf{N} \models \mathbf{h}$ which is the definition for $\mathbf{In(M)} \models_{\mathbf{W}} \mathbf{h}$. □

Theorem 5.4

Assume a domain description **W** with respect to a (finite) signature Σ. The following statements are equivalent:

(i) The hypotheses are explicitly definable w.r.t. **W**

(ii) The hypotheses are implicitly definable w.r.t. **W**

(iii) **W** is empirically founded.

Proof:

(i) => (ii) Suppose $\mathbf{N_1}$ and $\mathbf{N_2}$ with $\mathbf{In(N_1) = In(N_2)}$ are given. Take an arbitrary output literal $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ and the associated proposition of input literals **p** that exists by explicit definability. Then for any pair of situations $\mathbf{N_1, N_2} \in \mathbf{W}$ the property $\mathbf{In(N_1)} =$

$\mathbf{In(N_2)}$ implies $\mathbf{N_1 \models p}$ iff $\mathbf{N_2 \models p}$. By explicit definability this implies $\mathbf{N_1 \models h}$ iff $\mathbf{N_2 \models h}$. As we have taken an arbitrary output literal $\mathbf{h}$ we can conclude $\mathbf{Out(N_1) = Out(N_2)}$ that gives us implicit definability.

(ii) => (iii) Take an arbitrary output literal $\mathbf{h \in OutLit(\Sigma)}$ and a model $\mathbf{N_1 \in W}$ such that not $\mathbf{In(N_1)} \models_{\mathbf{W}} h$. Then there must exist an $\mathbf{N_2 \in W}$ such that $\mathbf{In(N_1) = In(N_2)}$ but not $\mathbf{N_2 \models h}$. The model $\mathbf{N_2}$ is complete so $\mathbf{N_2 \models \neg h}$. By implicit definability we then have $\mathbf{Out(N_1) = Out(N_2)}$, so also $\mathbf{N_1 \models \neg h}$.

This proves the contraposition of empirical foundedness.

(iii) => (i) Take an arbitrary output literal $\mathbf{h \in OutLit(\Sigma)}$ and consider the set of situation models of which the input forces $\mathbf{h}$, i.e.

$$\mathbf{V = \{ N \in W \mid In(N) \models_W h \}}.$$

The input of each model in $\mathbf{V}$ can be described by a proposition in the format of a conjunction of literals, and the disjunction of all these propositions is an explicit definition of $\mathbf{h}$ in terms of input literals. □

Theorem 5.6

Assume a domain description $\mathbf{W}$, and a sound knowledge base $\mathbf{KB}$ with respect to a signature Σ. Then the following notions are equivalent:

(i) $\mathbf{KB}$ is decisive with respect to $\mathbf{W}$

(ii) $\mathbf{KB}$ is weakly complete with respect to $\mathbf{W}$ and $\mathbf{W}$ is empirically founded.

Proof:

(i) $\Rightarrow$ (ii) Assume $\mathbf{KB}$ is decisive, and an output literal $\mathbf{h \in OutLit(\Sigma)}$ is given. We first prove weak-completeness, by considering an arbitrary situation model $\mathbf{N \in W}$ such that $\mathbf{In(N) \models_W h}$. By Lemma 2.12 we know that $\mathbf{N \models h}$ and in turn by decisiveness $\mathbf{In(N) \vdash_{KB}} h$. Hence, $\mathbf{KB}$ is weakly complete with respect to $\mathbf{W}$.

It remains to be shown that $\mathbf{W}$ is empirically founded. Consider an arbitrary model $\mathbf{N} \in W$ such that $\mathbf{N \models h}$. Decisiveness implies $\mathbf{In(N) \vdash_{KB}} h$.

Next by soundness $\mathbf{In(N)} \models_{\mathbf{W}} h$, hence empirical foundedness.

(ii) $\Rightarrow$ (i) Given an output literal $\mathbf{h \in OutLit(\Sigma)}$, we consider an arbitrary situation model $\mathbf{N} \in W$ such that $\mathbf{N \models h}$. Because $\mathbf{W}$ is empirically founded we know that $\mathbf{In(N) \models_W h}$, and next because $\mathbf{KB}$ is weakly complete $\mathbf{In(N) \vdash_{KB}} h$. As we have taken an arbitrary model $\mathbf{N}$ the property of decisiveness is satisfied. □

Theorem 6.5

Assume a domain description **W**, and a knowledge base **KB** with respect to a signature Σ. If **KB** is sound with respect to **W**, then the following notions are equivalent:

(i) **KB** is strongly complete with respect to **W**

(ii) **KB** is weakly complete with respect to **W** and well-informed with respect to **In(W)**.

Proof:

(i) $\Rightarrow$ (ii) If **KB** is strongly complete then for all partial models $\mathbf{M} \in \mathbf{P(In(W))}$ the completeness property holds. Because $\mathbf{W} \subseteq \mathbf{P(W)}$ we know that this property holds in particular for all complete models $\mathbf{N} \in \mathbf{In(W)}$. So **KB** is weakly complete. It remains to be shown that **KB** is well-informed. For this we assume given a partial model $\mathbf{M} \in \mathbf{P(In(W))}$ and output literal $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ such that for all $\mathbf{N} \in \mathbf{In(W)}$ it holds $\mathbf{N} \geq \mathbf{M} \Rightarrow \mathbf{N} \vdash_{\mathbf{KB}} \mathbf{h}$. By soundness we then know that for all such **N** also $\mathbf{N} \models_{\mathbf{W}} \mathbf{h}$ and by Theorem 6.3 we can conclude $\mathbf{M} \models_{\mathbf{W}} \mathbf{h}$. Finally strong completeness implies $\mathbf{M} \vdash_{\mathbf{KB}} \mathbf{h}$. Thus it has been shown that KB is well-informed.

(ii) $\Rightarrow$ (i) Given a model $\mathbf{M} \in \mathbf{P(In(W))}$ and output literal $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ such that $\mathbf{M} \models_{\mathbf{W}} \mathbf{h}$, we consider an arbitrary complete refinement $\mathbf{N} \in \mathbf{In(W)}$ with $\mathbf{N} \geq \mathbf{M}$. Here $\mathbf{N} = \mathbf{In(N')}$ for some $\mathbf{N'} \in \mathbf{W}$. By Corollary 2.11 we know that $\mathbf{N} \models_{\mathbf{W}} \mathbf{h}$ too. Next, because **KB** is weakly complete we know that $\mathbf{N} \vdash_{\mathbf{KB}} \mathbf{h}$. As we have taken an arbitrary **N** we can conclude for all $\mathbf{N} \in \mathbf{In(W)}$ that $\mathbf{N} \geq \mathbf{M} \Rightarrow \mathbf{N} \vdash_{\mathbf{KB}} \mathbf{h}$, that by well-informedness implies $\mathbf{M} \vdash_{\mathbf{KB}} \mathbf{h}$. Concluding $\mathbf{M} \models_{\mathbf{W}} \mathbf{h}$ implies $\mathbf{M} \vdash_{\mathbf{KB}} \mathbf{h}$ and **KB** is strongly complete with respect to **W**.□