Semantic Formalisation of Interactive Reasoning Functionality

Jan Treur
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
In this paper a semantical framework is developed that provides a logical description of the functionality of an interactive reasoning process. The concept of functionality description defines the functionality of a reasoning process abstracting from specific inference relations or knowledge bases. Moreover, a domain description is formalised. A number of properties of a functionality description are identified, and related to properties of the domain. It is established under which conditions a functionality, can be implemented by an inference relation and a knowledge base.

Introduction

In compositional agent- and knowledge-based systems the task or reasoning pattern is built up as a dynamic interaction between the components representing the subtasks; see [3], or [7]. Each of the primitive components is an interactive reasoning system based on an (often domain-specific) knowledge base. To design and specify compositional agent- and knowledge-based systems for complex tasks, the compositional design method DESIRE (DEsign and Specification of Interacting REasoning components; for instance see [2]) has been developed

To obtain a clear and well-defined analysis of the behaviour of such a dynamic compositional system, one has to start by defining the (interactive) role of a given primitive interactive component in such a system: a clear definition is needed of an interactive reasoning component's functionality. In particular, to support reusability and maintainability a functionality description independent of the component's specific internal knowledge representation, inference relations or implementation is required. This enables information hiding within a reasoning system: the component's internal structure can be changed as long as its functionality remains the same.

In a compositional system questions concerning the behaviour of the whole system or satisfaction of the system's functionality, can be decomposed into questions related to the system’s compositional structure and questions for the case of a single primitive interactive reasoning component (cf. [12]).

This paper concentrates on the case of a primitive reasoning component and its interactions. For the reasons indicated, in this paper we analyse in more detail semantical and functional aspects of interactive reasoning components. It will be shown how to obtain in a semantical manner a logical description of reasoning that makes use of information "from outside"; in this case by "outside" we mean outside the reasoning component, but possibly inside the system. Such a logical description provides an explicit distinction between situation-independent knowledge in the knowledge base (described as a logical theory) and situation-specific information which can be used as an input, imported from the world situation outside (that is described as a given situation model). This perspective is similar to the perspective in [18] (and [10]), where the models representing specific information are called simulation structures.

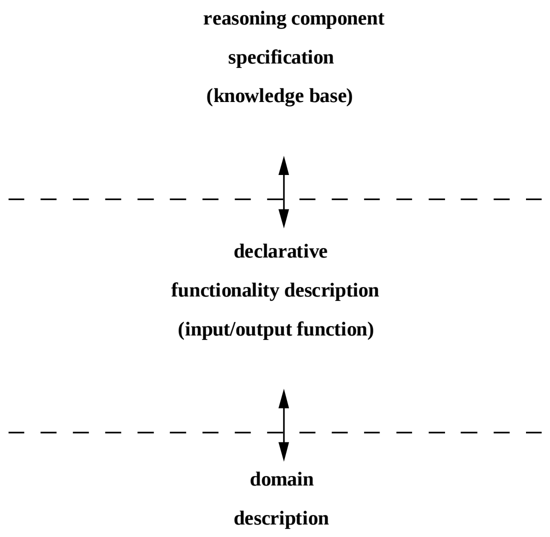

Fig. 1 Domain description, functionality description and reasoning component specification

On the one hand, from the computer science viewpoint, the framework enables us to define what a declarative functionality of a reasoning component is (the input/output possibilities provided by the component). We abstract from the dynamic aspects of the component: in this paper we are only concerned with what facts *can* be derived in a given situation (information state), and not with *at what time* and in which order specific facts *are* derived. We prove that for any declarative functionality description a knowledge base specification in rule-format can be found such that the semantical consequence relation applied to this knowledge base and the additional input facts satisfies the required functionality. Reasoning components are able to draw partial conclusions if only a partial input information state is given. Therefore our formal description of the declarative functionality of a component also treats partiality of information both at the input side and the output side of the component.

On the other hand, from the logical viewpoint, this framework enables us to define in a direct and logical way when a reasoning component's functionality is sound and complete with respect to the domain that is concerned. The central part of this paper is built up as shown in Fig. 1.

In Section 2 we define what a domain description is, and especially, in which manner world situations occurring in a domain and partial descriptions of them (information states) can be modelled (the lowest layer in Fig. 1). We will give some examples and prove some properties. In Section 3 we give a formal definition of a declarative functionality description of an interactive reasoning component and treat some examples (the middle layer in Fig. 1). Furthermore, we give some results on how well a declarative functionality description fits to (or describes) a given domain description (the lowest arrow in Fig. 1). In Section 4 it is defined what a reasoning component specification is (the highest layer in Fig. 1) and how well it fits to (or describes) a declarative functionality description (the highest arrow in Fig. 1) and to a domain description (both arrows). In Section 5 it is shown for an application to a diagnostic process model how declarative functionality descriptions can be used to specify (required) properties of a primitive reasoning component in a concise and transparent manner. In the Appendix proofs can be found. Before treating this all, Section 1 provides an introduction to what an interactive reasoning component is.

1 Interactive Reasoning

In this section we explain the notions of interactive reasoning and of an interactive reasoning component. Logical descriptions will be given that will be used and extended in later chapters. In these logical descriptions the notion of (partial) model as known from logic plays an important role. We use partial models to represent information states of a reasoning component at a certain moment. Our logical descriptions will be based on partial propositional logic. If (many-sorted) predicate logic with a finite number of object names and relation names is used, we can always use a propositional translation of the formulas to fall in the scope of propositional logic.

1.1 Interactive reasoning by combining knowledge from theory and model

Usually a reasoning component is described logically by a static theory **KB** (the knowledge base) which is given beforehand. From this theory conclusions can be derived using some inference mechanism. In case of an interactive reasoning component the information that serves as input facts during the reasoning is viewed as knowledge which is present from the start. In fact one deals with some extended theory $\mathbf{KB}^+$ which additionally contains all information which could be put in from the outside. Using such information is essentially the using of knowledge elements from the theory $\mathbf{KB}^+$ in the inference process, just like the knowledge from **KB** is used. This logical description enables one to simply apply the classical theory on derivability as studied in logic. This seems plain; however, there are some aspects which stay implicit or informal this way.

In fact there is not one theory **KB** but a whole *variety of theories* $\mathbf{KB}^+$ which depend on situations that can occur in reality, part of which is represented by the input facts that are given to the component. This may be viewed as an implicit parametrization of the knowledge base. How can an explicit logical description of this parametrization be obtained ? Moreover, interactive reasoning components often have the property that during a session the input facts are added incrementally: for any moment there is only partial input information available. The component draws partial conclusions from this information; these conclusions may occasionally affect which other input facts are added. For example, in a complex reasoning task like diagnosis these dynamic effects are essential (see [17]). Therefore the knowledge base $\mathbf{KB}^+$ may be dynamic during the reasoning.

It turns out that some parts of the knowledge (the general or situation-independent knowledge) which can be used in the inference process can be described as a *theory* **KB** but other parts (situation-specific knowledge) are better described as a *model* which represents the actual situation in reality, part of which is represented by the input facts.

In reasoning these two kinds of knowledge are *combined* by an interactive process as depicted in Fig. 2. Here the lowest layer depicts the world situation we are interested in; the shaded area of the world situation is the part of the world situation that is not observable. Since we cannot observe it, we are, in particular, interested in drawing conclusions about this part.

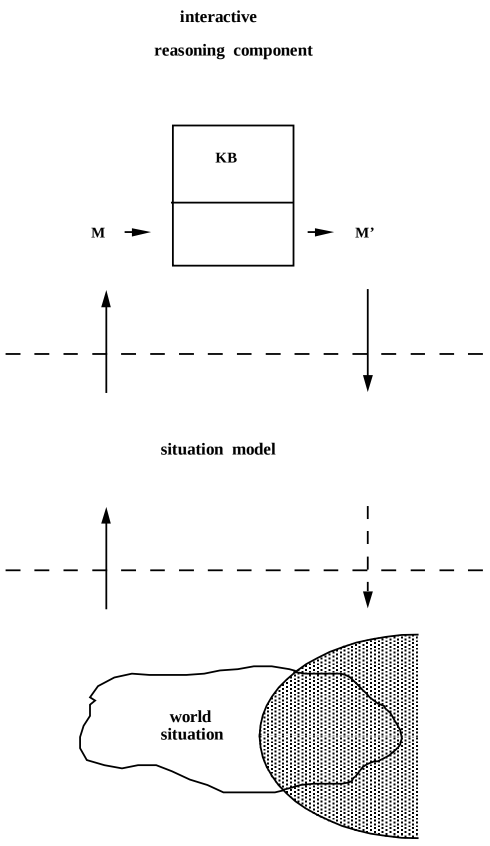

Fig. 2 Interactive reasoning

This can be done by reasoning, as performed by the reasoning component in the highest layer in Fig. 2. The world situation is modelled in some manner: we are only interested in some types of information, and the reasoning component requires a strict format of inputs. Therefore we use a situation model (the middle layer in Fig. 2). The reasoning makes use of observable facts from the world situation (the arrows on the left side, pointing upwards), as modelled by the situation model. In reasoning it applies general domain knowledge as stored in **KB**. This reasoning process provides additional information that can be used to extend the situation model (the highest right arrow, pointing downwards); this can be used to take a decision to carry out some action in the world situation (the lowest right arrow).

If the model representing the actual situation is a logical model of the theory which is used, and the inference mechanism of the component is sound, then logical conclusions derived in this way are always true in the model. In this paper we deal with these issues. The treatment we give provides a formal framework for interactive reasoning components as defined in the following section.

1.2 Logical description of an interactive reasoning component

We consider a description in terms of propositional with a non-empty set of *atoms* **A** containing two subsets **S** (possible input atoms) and **H** (possible output atoms). The set of atoms **A** collects all atomic propositions that are used in the reasoning component. The input atoms (sometimes called: observables) are the atoms for which a truth value may serve as an input to the component from the outside: from other reasoning components or from the outside world. The truth values of output atoms may be wanted by the user; since the component can derive them by logical inferences from the input atoms, the component may provide them to the user or to another component that needs them. It is possible that these two subsets contain all atoms, but it is also possible that there exist atoms outside these subsets; these *intermediate atoms* can play a role as intermediate results or subgoals in inferences. In the general case they form some subset **I** of **A**. Summarizing:

$$\mathbf{A} = \mathbf{S} \cup \mathbf{I} \cup \mathbf{H}$$

$$\mathbf{S} = \{s_1, s_2, ...\} \quad \mathbf{I} = \{i_1, i_2, ...\} \quad \mathbf{H} = \{h_1, h_2, ...\}$$

Notice that these sets may have nonempty intersections, although in the examples given below we deal with a disjoint union. From these atoms the set of propositional formulas can be built up by using the logical connectives $\wedge, \vee, \rightarrow, \neg$. Applications in knowledge bases are often based on a subset of these propositions, namely propositions in *rule-format*. These are propositions in implication form where the if-part consists of a conjunction of literals and the then-part consists of one literal (a literal is the basic unit of information we use: it is an atom or the negation of an atom). As a special case of formulas in rule-format, we allow single literals. These are to be interpreted as *general facts* (they may be viewed as rules with an empty condition-part). Examples of propositions in rule-format are the ones in Fig. 3 below.

rule 1	$s_1 \wedge s_2$	$\rightarrow$	i_1
rule 2	$i_1 \wedge s_3 \rightarrow$	h_1	
rule 3	$i_1 \wedge \neg s_3$	$\rightarrow$	$\neg h_1$
rule 4	$i_1 \wedge \neg s_3$	$\rightarrow$	h_2
rule 5	s_3	$\rightarrow$	$\neg h_2$
rule 6	s_4	$\rightarrow$	h_1

Fig. 3 Example of a knowledge base

To obtain a formal description of the interaction of the reasoning component with the outside, we will have to give a formal description of all situations the component may get in touch with (for instance all possible patients). For instance one may think of patient models containing all factual knowledge on some patient, such as symptoms, diseases etcetera. The reasoning is supposed to be always about one of these situations; in one session the user always gives the component information about a fixed one of them. This excludes a user who gives incorrect or arbitrary answers: we expect a user to transform factual knowledge from the situation that is considered to the component in a correct manner.

We define a situation model as a *truth assignment* to the atoms. Some examples of situation models related to the knowledge base of Fig. 3 may be obtained by taking the following truth-assignments, corresponding to the tuple $< \mathbf{s_1, s_2, s_3, s_4; i_1 ; h_1, h_2} >$.

$$\mathbf{M_1 :\ < 1, 1, 1, 0; 1; 1, 0 >}$$
$$\mathbf{M_2 :\ < 1, 1, 0, 0; 1; 0, 1 >}$$
$$\mathbf{M_3 :\ < 0, 0, 0, 1; 0; 1, 0 >}$$

Fig. 4 Some situations for KB from Fig. 3

Notice that these situation models are models of the theory **KB** (i.e. in each situation **M** every rule of **KB** is true in **M**). In this way one can represent knowledge which is situation-specific (facts from reality) in the form of truth assignments. Below we will give a number of definitions to explain this point more precisely.

During a reasoning process at each moment in the system only a part of the information as given by a situation model is available. To represent this partiality we make use of partial (situation) models: assignments of truth values from **{0, 1, u}** to each of the atoms. Here the **u** denotes *undefined* or *unknown*. Notice that the models as defined above (e.g. in Fig. 4) are included here as the truth assignments where to no atom an **u** is assigned; we distinguish them as the *complete models* among the partial models. Therefore the set of complete models is a subset of the set of partial models. Sometimes partial models that are not complete will be called *incomplete models*. So the word "partial" means "maybe complete, maybe incomplete".

A partial model may be constructed by replacing in any model as above some symbols **0** or **1** by the symbol **u**; examples of partial models are:

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

It is easy to see that, for the complete models given in Fig. 4, there may exist several hundreds of partial models, representing all information states of the component that eventually may occur; of course we will not enumerate them.

The dynamics of the reasoning of a component may be described by representing the trace of subsequent information states by a sequence of partial models. Each inference step constructs a new partial model as a refinement of the current partial model by adding to it the derived information. To illustrate this, in Fig. 5 we give an example that is related to the knowledge base in Fig. 3; here the tuples correspond to
$< s_1, s_2, s_3, s_4; i_1 ; h_1, h_2 >$.

$$N_1 : \quad < 1, 1, 0, u; u; u, u >$$

inference step 1: rule 1

$$N_2 : \quad < 1, 1, 0, u; 1; u, u >$$

inference step 2: rule 3

$$N_3 : \quad < 1, 1, 0, u; 1; 0, u >$$

inference step 3: rule 4

$$N_3 : \quad < 1, 1, 0, u; 1; 0, 1 >$$

Fig. 5 Example of a reasoning trace

Here the initial facts $s_1, s_2, \neg s_3$ are observations, done in the world situation that is reasoned about; this may be modelled by:

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

Furthermore, we assume that the initial information we know is about s_1, s_2, s_3. As pointed out earlier, for the atomic statements on which no truth value **0** or **1** is known the symbol **u** is noted. The reasoning is data-driven and based on chaining; it stops if no new conclusions can be derived anymore. The chain of partial models constructed by this reasoning process is:

$$N_1 \leq N_2 \leq N_3$$

Here $\leq$ is the refinement relation between partial models (i.e. in N_2 more information is known than in N_1, et cetera). Notice that $N_i \leq M$ for each **i**.

We finish this section by summing up some of the properties we assume reasoning components have:
- the component can be used by putting in some input facts (data-driven, forward) as well as by asking a query to it (goal-directed, backward) or a combination of these.

- the component can draw (partial) conclusions from partial input information: not for all possible input facts truth values are needed
- the input facts are not changed by the processing of the component; i.e., once a fact has been input as a true fact it remains true during the inference proces (conservatism)
- if a larger set of input facts is given to the component, also the set of derivable facts is larger (monotonicity)

1.3 Information states in an implemented system

In an implementation of a reasoning component in an Expert System Shell, the information state of the component is represented by what is often called the (dynamic) *facts base*. The relation of such a representation with a partial model may be viewed as depicted in Fig. 6. Here the horizontal arrow depicts that the information state in the system represents (a part of) the world situation that is reasoned about. The vertical arrows depict that both the information state of the system and the world situation can be formalized by the use of a partial model.

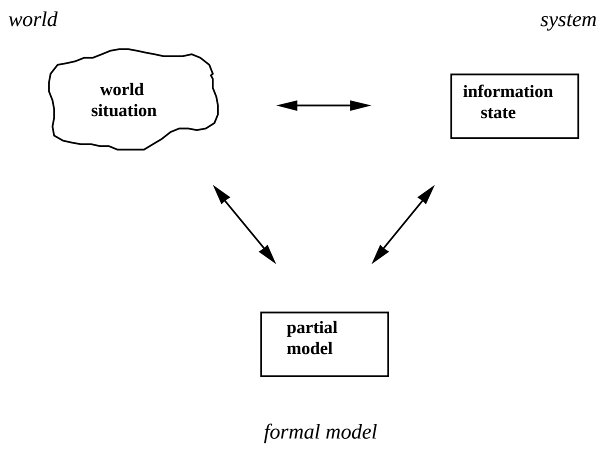

Fig. 6 World, system and formal model

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

encodes the information that the atomic statement

color(bird1, grey)

is true. In the same time we can have the information that **color(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 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), color(bird1, grey), color(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.

2 Domain Descriptions

In section 1 we have sketched how a reasoning component relates to the domain that is concerned, and the specific world situation it is reasoning about. To be able to compare the conclusions drawn by a reasoning component to the facts in this world situation in the domain, a formal framework is needed for describing the domain and the interaction of the reasoning component with it. Therefore below we will start by giving more precise definitions that constitute a formal framework to describe a domain. In section 3 and 4 the relation between a domain and a reasoning component is defined more formally.

2.1 Describing world situations by complete models

We start by giving the definition of the language elements to describe a domain. In this paper we will work only with finite propositional logical languages.

Definition 2.1

A (propositional) *signature* Σ is a 3-tuple $<$ **InSig(Σ); IntSig(Σ); OutSig(Σ)** $>$ where **InSig(Σ)**, **IntSig(Σ)**, **OutSig(Σ)** are ordered sets of atom names, respectively called the *input signature*, the *internal signature* and the *output signature*. The input and output signatures may contain common atom names, but the internal signature is disjoint from them. In this paper all signatures are assumed to be *finite*.

The corresponding sets of literals are denoted by **InLit(Σ)**, **IntLit(Σ)**, **OutLit(Σ)**, while their union is denoted by **Lit(Σ)**. If **M** is a (partial) model based on this signature, then we say **M** is *of signature* Σ .

The signature Σ' is called a *subsignature* of Σ if **InSig(Σ') ⊂ InSig(Σ), IntSig(Σ') ⊂ IntSig(Σ)** and **OutSig(Σ') ⊂ OutSig(Σ).**

Each of the input, internal and output signatures are viewed as special cases of subsignatures. The following is an example of a signature:

$$< s_1, s_2, s_3, s_4; i_1 ; h_1, h_2 >.$$

In this paper we often consider only the relations between inputs and outputs; in that case we leave out the internal signature.

Definition 2.2

A *domain description* **W** for Σ is a non-empty set of complete truth assignments to the atoms.

In section 1, Fig. 4, an example of a domain description is shown. We assume **W** is an (arbitrary) given set of situations, representing these situations that actually occur in reality. This enables us to disregard from non-existing world situations with some as yet unknown combination of observables and hypotheses. As discussed in section 1, the reasoning is supposed to be always about one of these situations: we expect a user to transform factual knowledge from the situation in a correct manner.

Remark: unless the example suggests, in practice the set **W** will not be enumerated; it simply will be established that such a set exists, and maybe some typical and/or critical examples of situation models will be described and used as a test set.

2.2 Describing information states by partial models

During a reasoning process, only a part of the information about a world situation is available to the reasoning system. At any moment the system's information state can be described by a partial model, as has been discussed in the sections 1.2 and 1.3. In this section we will give the formal framework.

A (partial) *model* $\mathbf{M}$ of signature Σ is an assignment of truth values from $\{\mathbf{0, 1, u}\}$ to the atoms of Σ. By $\mathbf{M(a)}$ we will denote the truth value assigned to atom $\mathbf{a}$. We call $\mathbf{M}$ a *complete model* if for all atoms $\mathbf{a}$ the truth value $\mathbf{M(a)}$ is not $\mathbf{u}$. Let $\mathbf{V}$ be a non-empty set of partial models of signature Σ. By $\mathbf{P(V)}$ we denote the set of all partial models that may be refined to a model in $\mathbf{V}$, and by $\mathbf{C(V)}$ we denote the set of complete models in $\mathbf{V}$. Notice that $\mathbf{C(V)} \subset \mathbf{V}$ and $\mathbf{V} \subset \mathbf{P(V)}$ but in general $\mathbf{P(V)} \subset \mathbf{V}$ and $\mathbf{V} \subset \mathbf{C(V)}$ are not the case. A relevant example is $\mathbf{V} = \mathbf{P(W)}$, where $\mathbf{W}$ is a domain description. In this case $\mathbf{C(P(W))} = \mathbf{W}$.

If $\mathbf{M} \in \mathbf{P}$ we denote by $\mathbf{Lit(M)}$ the set of literals which are true in $\mathbf{M}$, and by $\mathbf{InLit(M)}$ the set of input literals which are true in $\mathbf{M}$. Similarly $\mathbf{OutLit(M)}$ denotes the set of output literals which are true in $\mathbf{M}$. These sets of propositions correspond to the restrictions of the model $\mathbf{M}$ to the restricted sets of atoms $\mathbf{S}$ and $\mathbf{H}$. These restricted models are sometimes called *reducts* (also see [5]); the reduct of $\mathbf{M}$ to a subsignature Σ' is denoted by $\mathbf{M|\Sigma'}$.

Suppose Σ' is a subsignature of Σ and $\mathbf{M'}$ is a model of signature Σ'. The *trivial expansion* $\mathbf{M}$ of $\mathbf{M'}$ to the signature Σ is the model of signature Σ created from $\mathbf{M'}$ by assigning a $\mathbf{u}$ to any atom outside Σ'.

Let $\mathbf{M}$ be a model of signature Σ .The model $\mathbf{In(M)}$ is defined as the trivial expansion of $\mathbf{M|\Sigma_{in}}$ to the signature Σ. This model represents the *input part* of $\mathbf{M}$; similarly $\mathbf{Out(M)}$ represents the *output part* of $\mathbf{M}$. For example, the input parts of the models of Fig. 4 are given in Fig. 7.

reduct to Σ_{in}	*input part*
$\mathbf{M_1|\Sigma_{in}} : < 1, 1, 1, 0 >$	$\mathbf{In(M_1)} : \ < 1, 1, 1, 0; u; u, u >$
$\mathbf{M_2|\Sigma_{in}} : < 1, 1, 0, 0 >$	$\mathbf{In(M_2)} : \ < 1, 1, 0, 0; u; u, u >$
$\mathbf{M_3|\Sigma_{in}} : < 0, 0, 0, 1 >$	$\mathbf{In(M_3)} : \ < 0, 0, 0, 1; u; u, u >$

Fig. 7 Input parts of the models in Fig. 4

The user (or other component) who is interacting with the reasoning component can only provide information about the input part of the situation that is concerned. Notice that

$$\mathbf{InLit(M)} = \mathbf{Lit(In(M))}$$

$$\mathbf{OutLit(M)} = \mathbf{Lit(Out(M))}$$

If $\mathbf{W}$ is a domain description then the *related set of complete input models* (resp. of *complete output models*), denoted by $\mathbf{In(W)}$ (resp. $\mathbf{Out(W)}$), is the set of all possible complete input (resp. output) models from models of $\mathbf{W}$, that is:

$$\mathbf{In(W)} = \{\mathbf{In(M)} \mid \mathbf{M} \in \mathbf{W}\}$$

$$\mathbf{Out(W)} = \{\mathbf{Out(M)} \mid \mathbf{M} \in \mathbf{W}\}$$

A model $\mathbf{M}$ of signature Σ with $\mathbf{M(a) = u}$ for all atoms $\mathbf{a}$ not in $\mathbf{InSig}(\Sigma)$ is called an *input model*. Similarly, $\mathbf{M}$ is called an *output model* of signature Σ if $\mathbf{M(a) = u}$ for all atoms $\mathbf{a}$ not in $\mathbf{OutSig}(\Sigma)$. We will use the symbols $\mathbf{P_{in}}$, $\mathbf{P_{out}}$ for non-empty sets of input models, respectively output models. If all these models are complete with respect to $\mathbf{InSig}(\Sigma)$ resp. $\mathbf{OutSig}(\Sigma)$ they are denoted by respectively $\mathbf{W_{in}}$, $\mathbf{W_{out}}$. Notice that the models $\mathbf{In(M)}$ resp. $\mathbf{Out(M)}$ are input resp. output models in this sense. If no confusion is expected, for convenience we sometimes will leave out the $\mathbf{u}$'s; for instance we may write $< \mathbf{1, 1, 1, 0} >$ for the input model $< \mathbf{1, 1, 1, 0; u ; u, u} >$.

By $\mathbf{M} \vDash \mathbf{p}$ (to be read as: $\mathbf{M}$ *satisfies* $\mathbf{p}$) we denote that the proposition $\mathbf{p}$ is true in the model $\mathbf{M}$ according to the strong Kleene semantics. The combination tables for truth values according to this approach to partial semantics are given by:

$\neg\varphi$	
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

For more information on partial logic, see [1], [14]. By Λ we denote the *empty* partial model that contains no information at all: only $\mathbf{u}$'s are assigned. The refinement relation $\leq$ between partial models is defined by
$\mathbf{M} \leq \mathbf{N}$ if for every atom $\mathbf{a}$ it holds $\mathbf{M(a)} \leq \mathbf{N(a)}$ (i.e., point by point), where the partial ordering of truth values is defined by **unknown < true**, **unknown < false.**

Lemma 2.3

Suppose Σ' is a subsignature of Σ, and $\mathbf{M}$, $\mathbf{N}$ are models of signature Σ .

a) $\mathbf{M} \leq \mathbf{N}$ if and only if for all literals $\mathbf{c}$ it holds

$$\mathbf{M} \vDash \mathbf{c} \Rightarrow \mathbf{N} \vDash \mathbf{c}$$

b) If $\mathbf{M} \leq \mathbf{N}$ then $\mathbf{M}|\Sigma' \leq \mathbf{N}|\Sigma'$.

c) If $\mathbf{M'}$, $\mathbf{N'}$ are of signature Σ' and $\mathbf{M}$, $\mathbf{N}$ are their trivial expansions to Σ then $\mathbf{M} \leq \mathbf{N}$ if and only if $\mathbf{M'} \leq \mathbf{N'}$.

Let $\mathbf{V}$ be a non-empty set of partial models. If $\mathbf{M} \in \mathbf{P(V)}$ then by $\mathbf{M} \;|\!\!\models_V \mathbf{p}$ (to be read as: $\mathbf{M}$ *forces* $\mathbf{p}$ with respect to $\mathbf{V}$) we denote that for every refinement $\mathbf{N}$ of $\mathbf{M}$ in $\mathbf{V}$ it holds $\mathbf{N} \models \mathbf{p}$:

$$\mathbf{M} \;|\!\!\models_V \mathbf{p} \;\;\Leftrightarrow\;\; \forall \mathbf{N} \in \mathbf{V} \;\; [\, \mathbf{M} \leq \mathbf{N} \;\Rightarrow\; \mathbf{N} \models \mathbf{p} \,]$$

The *semantic consequence relation* restricted to models in $\mathbf{V}$, denoted by $\mathbf{F} \models_V \mathbf{p}$ is defined by: for all models $\mathbf{M'}$ in $\mathbf{V}$ with $\mathbf{M'} \models \mathbf{F}$ it holds $\mathbf{M'} \models \mathbf{p}$.

There is a connection between the forcing relation and the semantic consequence relation restricted to models in $\mathbf{V}$, namely:

$$\mathbf{M} \;|\!\!\models_V \mathbf{p} \;\;\Leftrightarrow\;\; \mathbf{Lit(M)} \models_V \mathbf{p}$$

We say the members of $\mathbf{V}$ *agree* on the atom $\mathbf{a}$ if for all $\mathbf{M} \in \mathbf{V}$ the atom $\mathbf{a}$ has the same truth value in $\mathbf{M}$. In the other case we say the members of $\mathbf{V}$ *disagree* on $\mathbf{a}$. For a nonempty set $\mathbf{V}$ of partial models the *greatest common information state* of $\mathbf{V}$ is the partial model $\mathbf{N}$, denoted by $\mathbf{N} = \mathbf{gci(V)}$, such that for all $\mathbf{M} \in \mathbf{V}$ it holds $\mathbf{N} \leq \mathbf{M}$ and for any $\mathbf{N'}$ satisfying this condition it holds $\mathbf{N'} \leq \mathbf{N}$. One may view this as the maximal information on which all members of $\mathbf{V}$ agree. This model $\mathbf{gci(V)}$ can be constructed as follows: for any atom $\mathbf{a}$ on which all members of $\mathbf{V}$ agree, take this truth value, and if the members of $\mathbf{V}$ disagree take the truth value $\mathbf{u}$. As an example, take the set $\mathbf{V}$ consisting of the partial models:

$$\langle\, 1, 1, 0, 1 \,\rangle$$

$$\langle\, 1, u, 0, u \,\rangle$$

$$\langle\, 0, 0, 0, u \,\rangle$$

In this case we have

$$\mathbf{gci(V)} \;=\; \langle\, u, u, 0, u \,\rangle$$

Notice that, just like the refinement relation, the operation $\mathbf{gci}$ is taken point by point. The following Lemma will be useful later on.

Lemma 2.4

a) Suppose $\mathbf{V_1}$ and $\mathbf{V_2}$ are nonempty and for every $\mathbf{N}$ in $\mathbf{V_2}$ there exists a $\mathbf{M}$ in $\mathbf{V_1}$ such that $\mathbf{M} \leq \mathbf{N}$. Then $\mathbf{gci(V_1)} \leq \mathbf{gci(V_2)}$. In particular this holds if $\mathbf{V_2} \subset \mathbf{V_1}$.

b) If $\mathbf{M}$ is given such that for every $\mathbf{N}$ in $\mathbf{V}$ it holds $\mathbf{M} \leq \mathbf{N}$, then $\mathbf{M} \leq \mathbf{gci(V)}$.

For any partial model $\mathbf{M} \in \mathbf{P(V)}$ this $\mathbf{gci}$-construction can be applied to the set of refinements in $\mathbf{V}$ of $\mathbf{M}$, i.e. to $\mathbf{V_0} = \{\, \mathbf{N} \in \mathbf{V} \mid \mathbf{M} \leq \mathbf{N} \,\}$. The resulting greatest common information state of $\mathbf{V_0}$ is called the *semantic closure* of $\mathbf{M}$ with respect to $\mathbf{V}$, and is denoted by $\mathbf{sc_V(M)}$. This refinement of $\mathbf{M}$ satisfies precisely the literals that are forced by $\mathbf{M}$ with respect to $\mathbf{V}$. This and some other properties that we will use later on are included in Lemma 2.5 below. As an example, the semantic closure of

$$\mathbf{M_0 = < 1, u, u, u; u; u, u >}$$

with respect to the domain description $\mathbf{W}$ in Fig. 4 is the greatest common information state of $\mathbf{\{M_1, M_2\}}$; therefore

$$\mathbf{sc_W(M_0) = < 1, 1, u, 0; 1; u, u >} .$$

Notice that a semantic closure belongs to $\mathbf{P(V)}$ but in general does not belong to $\mathbf{V}$ and is not complete, even if all models in $\mathbf{V}$ are complete.

Lemma 2.5

Suppose $\mathbf{M, N \in P(V)}$.

a) For any literal $\mathbf{d}$ it holds

$$\mathbf{sc_V(M) \vDash d \Leftrightarrow M \Vdash_V d}$$

b) For all partial models $\mathbf{M, N \in P(V)}$ it holds:

(i) If $\mathbf{M \leq N \in V}$ then $\mathbf{sc_V(M) \leq N}$

(ii) $\mathbf{M \leq sc_V(M)}$

(iii) $\mathbf{M \leq N \Rightarrow sc_V(M) \leq sc_V(N)}$

c) $\mathbf{M \leq sc_V(N) \Rightarrow sc_V(M) \leq sc_V(N)}$

and in particular $\mathbf{sc_V(sc_V(M)) = sc_V(M)}$

d) If every complete refinement of $\mathbf{M}$ is a member of $\mathbf{V}$ then $\mathbf{sc_V(M) = M}$. In particular this holds for every complete model $\mathbf{M}$ in $\mathbf{V}$.

The formalisation of information states by partial models above is based on propositional logic; for a many-sorted first order logic formalisation of information states by partial models, see [13].

2.3 Empirically founded domains

In the philosophy of science, for instance as in [11], criteria are given which a certain domain (an empirical science) must satisfy in order to have an empirical basis. Informally stated, such a criterion requires that every statement on the domain is essentially testable. This means that there exists a number of tests such that the statement is true if and only if a certain logical (boolean) combination of outcomes of these tests is satisfied. In our logical framework such a combination may be described formally as a proposition in terms of the input atoms; this will be shown in section 4. In terms of a domain description empirically foundedness means that the truth value of each output atom is uniquely determined by the truth values of the inputs. After the following Lemma we will give a formal definition for this.

Lemma 2.6

Let a domain description be given by $\mathbf{W}$. Then the following conditions are equivalent:

(i) For every pair of situations $\mathbf{M, N} \in \mathbf{W}$ which satisfy the same input literals it holds that they also satisfy the same output literals, i.e.:
   if $\mathbf{M, N} \in \mathbf{W}$ then

$$\mathbf{In(M) = In(N)} \Rightarrow \mathbf{Out(M) = Out(N)}$$

(ii) For all $\mathbf{M} \in \mathbf{W_{in}}$ the model $\mathbf{sc_W(M)}$ is complete.

Definition 2.7

Let a domain description be given by $\mathbf{W}$. The domain $\mathbf{W}$ is called *empirically founded* if one of the (equivalent) conditions of Lemma 2.6 is satisfied.

In case $\mathbf{H = A}$ these definitions provide that the models $\mathbf{M} \in \mathbf{W}$ are characterised uniquely by their input parts (notice that in this case the sets $\mathbf{S}$ and $\mathbf{H}$ will have a nonempty intersection). Notice also that the example in Fig. 4 satisfies these definitions; also in this example the models are characterised by their input parts.

In many disciplines one tries to build up a domain in such a manner that it is empirically founded. For instance in medical domains for every disease one tries to give a testable criterion which determines in what cases the disease occurs. In some cases these criteria are not correct or they are incomplete. However, there also exist many cases in which essentially correct and complete criteria are available, although in practical situations some of these criteria are not testable in an easy manner. For instance, for testing a certain criterion the patient has to be referred to a specialist, or the test is expensive, or risky, or takes a long time, et cetera. This may imply that for some time one has to draw (heuristic) conclusions from incomplete knowledge. In this paper we will not discuss this heuristic approach to incomplete knowledge; in [17] a treatment of this can be found.

2.4 Example

The following example shows how sometimes the set of observables (input atoms) can be extended to obtain an empirically founded domain. The domain $\mathbf{W}$ consists of the following truth assignments, corresponding to signature $< \mathbf{s_1, s_2, s_3; h_1, h_2} >$:

$\mathbf{M_1}$: $\langle 1, 1, 0; 1, 0 \rangle$

$\mathbf{M_2}$: $\langle 0, 1, 1; 1, 1 \rangle$

$\mathbf{M_3}$: $\langle 0, 1, 1; 1, 0 \rangle$

$\mathbf{M_4}$: $\langle 1, 0, 1; 0, 0 \rangle$

$\mathbf{W} = \{M_1, M_2, M_3, M_4\}$.

Fig. 8 Domain, not empirically founded

The domain described by $\mathbf{W}$ is not empirically founded: both $\mathbf{M_2}$ and $\mathbf{M_3}$ satisfy the same input literals but differ in the output literals they satisfy: condition (i) of Lemma 2.6 fails.

Now we extend the set $\mathbf{A}$ by a fourth input atom $\mathbf{s_4}$. This $\mathbf{s_4}$ holds in the second and fourth situation but not in the first and third one. The set $\mathbf{W'}$ is given by the following truth assignments, corresponding to $\langle s_1, s_2, s_3, s_4; h_1, h_2 \rangle$.

$\mathbf{M'_1}$: $\langle 1, 1, 0, 0; 1, 0 \rangle$

$\mathbf{M'_2}$: $\langle 0, 1, 1, 1; 1, 1 \rangle$

$\mathbf{M'_3}$: $\langle 0, 1, 1, 0; 1, 0 \rangle$

$\mathbf{M'_4}$: $\langle 1, 0, 1, 1; 0, 0 \rangle$

$\mathbf{W'} = \{M'_1, M'_2, M'_3, M'_4\}$.

Fig. 9 Domain, empirically founded

The domain described by $\mathbf{W'}$ is empirically founded, as may be easily verified by the use of Lemma 2.6(i).

3 Declarative Functionality Descriptions

After having defined in section 2 more precisely what a domain description is, we now turn to the properties of a reasoning component related to a given domain. By the declarative functionality of a reasoning component we mean what the component is able to derive, given certain specific input data. Notice that in our terms funcionality is not covered by simply describing what type of inputs in general *may* be needed, and separate from this what type of output in general *may* be produced. In our case by functionality we mean to describe for *any* set of specific input data, what specific output data in particular *will* or *should* occur, given these input data.

In this section, for a given domain description we treat what (declarative) functionality may be required from a reasoning component, in order to cover the domain description. Therefore we define what a declarative functionality description of a reasoning component is in section 3.1. Furthermore, the notions of soundness and completeness of a declarative functionality description with respect to a given domain description are defined in section 3.2. In section 4, in addition it will be treated in what format a component's knowledge base can be specified and when this specification meets the requirement as posed by a declarative functionality description.

3.1 Definitions, constructions and examples

In case of an empirically founded domain it looks rather trivial how a corresponding reasoning component's functionality should be defined. Given a complete input model, the output of the component simply is prescribed by the unique model from **W** that refines the input model. In the case of the empirically founded **W'** of Fig. 9 we could simply define the component's declarative functionality by the mapping α**: In(W')** $\rightarrow$ **W'** given by

$$\begin{aligned}
&\mathbf{< 1, 1, 0, 0 >} \rightarrow \mathbf{< 1, 1, 0, 0; 1, 0 >}\\
&\mathbf{< 0, 1, 1, 1 >} \rightarrow \mathbf{< 0, 1, 1, 1; 1, 1 >}\\
&\mathbf{< 0, 1, 1, 0 >} \rightarrow \mathbf{< 0, 1, 1, 0; 1, 0 >}\\
&\mathbf{< 1, 0, 1, 1 >} \rightarrow \mathbf{< 1, 0, 1, 1; 0, 0 >}
\end{aligned}$$

Notice that for convenience we include the input parts in the resulting models.

This seems a rather straightforward approach. However, there are two complications that require a more detailed analysis. Firstly, the domain may not be empirically founded at all; in that case there is no *unique* refinement in **W**. So some output literals will have to remain indeterminate. Secondly, a reasoning component is expected to give some (partial) answers in the case of an incomplete input model as well. These partial answers cannot be read directly from the complete models in **W**. Both complications have to do with incomplete information (in input and/or in output). We will extend the above approach by using partial models both for input models and output models to specify these incompletenesses.

In this section we consider the following example of a domain description. The signature is given by $< \mathbf{s_1, s_2; h_1, h_2} >$. The domain $\mathbf{W}$ is given by the following situation models:

$\mathbf{M_1}$: $< 0, 0; 0, 0 >$

$\mathbf{M_2}$: $< 0, 1; 0, 1 >$

$\mathbf{M_3}$: $< 1, 0; 0, 0 >$

$\mathbf{M_4}$: $< 1, 0; 1, 0 >$

$\mathbf{M_5}$: $< 1, 1; 1, 1 >$

Fig. 10 Example domain

Notice that this domain is not empirically founded: both $\mathbf{M_3}$ and $\mathbf{M_4}$ have the same input part, but they have different output parts.

In this case we can define a corresponding declarative functionality description by the mapping α: $\mathbf{In(W)} \rightarrow \mathbf{P}$, where $\mathbf{P} = \mathbf{P(W)}$, given by:

$< 0, 0 > \rightarrow < 0, 0; 0, 0 >$

$< 0, 1 > \rightarrow < 0, 1; 0, 1 >$

$< 1, 0 > \rightarrow < 1, 0; u, 0 >$

$< 1, 1 > \rightarrow < 1, 1; 1, 1 >$

Fig. 11 Functionality description for complete input models

Here a $\mathbf{u}$ is assigned to the output atoms on which there is no common opinion in $\mathbf{W}$, given the complete input model. So, Fig. 11 is constructed by taking the greatest common information state of the refinements of (the trivial expansion of) $< 1, 0 >$ in $\mathbf{W}$, in other words by taking $\mathbf{sc_W}(< 1, 0 >)$, where $< 1, 0 >$ is identified with its trivial expansion. This indicates a simple quite natural technique to solve the first one of the complications mentioned above, by simply allowing partial models in the range of α.

The second complication as mentioned above deals with the case of incomplete input information. For example, suppose the partial input model $< u, 1 >$ is given. What should the reasoning component conclude about $\mathbf{h_1}$ and $\mathbf{h_2}$ in this case ? A trivial answer could be: nothing, i.e. assign $\mathbf{u}$ to both $\mathbf{h_1}$ and $\mathbf{h_2}$. However, if $\mathbf{W}$ is inspected it turns out that there are only two refinements of (the trivial expansion of) $< u, 1 >$ in $\mathbf{W}$, namely:

$\mathbf{M_2}$: $\mathbf{< 0, 1; 0, 1 >}$

$\mathbf{M_5}$: $\mathbf{< 1, 1; 1, 1 >}$

These two situation models disagree on $\mathbf{h_1}$, but they agree that $\mathbf{h_2}$ is true. Therefore the reasoning component may be expected to give as an answer: $\mathbf{< u, 1; u, 1 >}$. This is the greatest common information state of all refinements in $\mathbf{W}$ of the given partial input model $\mathbf{< u, 1 >}$, i.e. $\mathbf{sc_W(< u, 1 >) = gci(\{M_2, M_5\})}$. So there are methods to obtain a non-trivial extension of the functionality description $\alpha\mathbf{: In(W) \to P}$ to a mapping $\alpha\mathbf{: P(In(W)) \to P}$. In our example this leads to the mapping α as defined by Fig. 12.

$\mathbf{< 0, 0 > \to < 0, 0; 0, 0 >}$

$\mathbf{< 0, 1 > \to < 0, 1; 0, 1 >}$

$\mathbf{< 0, u > \to < 0, u; 0, u >}$

$\mathbf{< 1, 0 > \to < 1, 0; u, 0 >}$

$\mathbf{< 1, 1 > \to < 1, 1; 1, 1 >}$

$\mathbf{< 1, u > \to < 1, u; u, u >}$

$\mathbf{< u, 0 > \to < u, 0; u, 0 >}$

$\mathbf{< u, 1 > \to < u, 1; u, 1 >}$

$\mathbf{< u, u > \to < u, u; u, u >}$

Fig. 12 Functionality description α for partial input models

Here any right hand side is obtained by taking the greatest common information state of all refinements in $\mathbf{W}$ of the corresponding left hand side (i.e. by taking its semantic closure). We summarize the construction of the mapping α from the domain description as carried out above.

Construction of a functionality description from a domain description

1. List all possible (partial) input models and take any of them;
   as an example we choose $\mathbf{< u, 0 >}$
2. For any input model $\mathbf{M}$ collect the situations from $\mathbf{W}$ that refine $\mathbf{M}$ in a set $\mathbf{V(M)}$;
   according to Fig. 10 this set consists of $\mathbf{M_1}$, $\mathbf{M_3}$, and $\mathbf{M_4}$.
3. Take the greatest common information state of this set: $\mathbf{gci(V(M))}$;
   this is $\mathbf{< u, 0; u, 0 >}$
4. Define $\alpha\mathbf{(M) = gci(V(M))}$.

5. Repeat this for all other input models.

It turns out that this construction results in a mapping $\alpha\mathbf{:}\ \mathbf{P_{in}} \rightarrow \mathbf{P}$ which satisfies a number of nice properties as defined by the following:

Definition 3.1
Suppose a signature Σ is given, a non-empty set of complete input models $\mathbf{W_{in}}$ for Σ and a mapping $\alpha\mathbf{:}\ \mathbf{P_{in}} \rightarrow \mathbf{P}$, where $\mathbf{P_{in}} = \mathbf{P(W_{in})}$ and $\mathbf{P}$ is a set of partial models for Σ .
a) The mapping α is called *conservative* if for all $\mathbf{M} \in \mathbf{P_{in}}$ it holds:

$$\mathbf{M} \leq \alpha\mathbf{(M)}$$

b) The mapping α is called *monotonic* if for all $\mathbf{M, N} \in \mathbf{P_{in}}$ it holds:

$$\mathbf{M} \leq \mathbf{N} \Rightarrow \alpha\mathbf{(M)} \leq \alpha\mathbf{(N)}$$

c) The mapping α is called *self-bounded* if for all $\mathbf{M, N} \in \mathbf{P_{in}}$ it holds:

$$\mathbf{M} \leq \alpha\mathbf{(N)} \Rightarrow \alpha\mathbf{(M)} \leq \alpha\mathbf{(N)}$$

d) The mapping α is called *well-informed* if for all $\mathbf{M} \in \mathbf{P_{in}}$ it holds:

$$\mathbf{Out(}\alpha\mathbf{(M))} = \mathbf{gci(\{Out(}\alpha\mathbf{(N))} \mid \mathbf{N} \in \mathbf{W_{in}}\ \&\ \mathbf{M} \leq \mathbf{N}\})$$

e) The mapping $\beta\mathbf{:}\ \mathbf{P_{in}} \rightarrow \mathbf{P}$ such that for all $\mathbf{N} \in \mathbf{W_{in}}$ it holds $\alpha\mathbf{(N)} = \beta\mathbf{(N)}$ is called *better informed* than α if $\mathbf{Out(}\alpha\mathbf{(M))} \leq \mathbf{Out(}\beta\mathbf{(M))}$ for all $\mathbf{M} \in \mathbf{P_{in}}$.
f) The mapping α does not affect inputs if for all $\mathbf{M} \in \mathbf{P_{in}}$ it holds:

$$\alpha\mathbf{(M)|}\Sigma_{\mathbf{in}} = \mathbf{M|}\Sigma_{\mathbf{in}}$$

g) The mapping α is called *regular* if it does not affect inputs that are no outputs, i.e. if for all $\mathbf{M} \in \mathbf{P_{in}}$ it holds:

$$\alpha\mathbf{(M)|}\Sigma_{\mathbf{in\backslash out}} = \mathbf{M|}\Sigma_{\mathbf{in\backslash out}}$$

where $\Sigma_{\mathbf{in\backslash out}}$ is the part of the input signature that is not included in the output signature.

As a result of the analysis above, we use these properties to define the notion of a declarative functionality description. However, first we establish some logical relations between these properties, for instance:

Lemma 3.2
Suppose a signature Σ is given, a non-empty set of complete input models $\mathbf{W_{in}}$ for Σ and a mapping $\alpha\mathbf{:}\ \mathbf{P_{in}} \rightarrow \mathbf{P}$, where $\mathbf{P_{in}} = \mathbf{P(W_{in})}$ and $\mathbf{P}$ is a set of partial models for Σ .
a) α is conservative if and only if $\mathbf{M} \leq \mathbf{In(}\alpha\mathbf{(M))}$ for all $\mathbf{M} \in \mathbf{P_{in}}$
b) If α is conservative and self-bounded then α is monotonic.

c) If α is conservative then the following are equivalent:

(i) α is self-bounded

(ii) α is monotonic and for all $\mathbf{M} \in \mathbf{P_{in}}$ it holds

$$\alpha(\mathbf{In}(\alpha(\mathbf{M}))) = \alpha(\mathbf{M})$$

If, moreover, $\Sigma_{\mathbf{in}} = \Sigma_{\mathbf{out}}$ then this is equivalent to

(iii) α is monotonic and $\alpha(\alpha(\mathbf{M})) = \alpha(\mathbf{M})$ for all $\mathbf{M} \in \mathbf{P_{in}}$ (idempotency).

d) If α does not affect the inputs and α is conservative then the following are equivalent:

(i) α is monotonic

(ii) α is self-bounded.

e) If $\alpha, \beta\mathbf{:}\ \mathbf{P_{in}} \rightarrow \mathbf{P}$ are mappings such that for all $\mathbf{N} \in \mathbf{W_{in}}$ it holds $\alpha(\mathbf{N}) = \beta(\mathbf{N})$, α is monotonic and β is well-informed, then β is better informed than α.

f) If $\alpha, \beta\ \mathbf{:}\ \mathbf{P_{in}} \rightarrow \mathbf{P}$ are mappings such that for all $\mathbf{N} \in \mathbf{W_{in}}$ it holds $\alpha(\mathbf{N}) = \beta(\mathbf{N})$ and both mappings are monotonic and well-informed then for all $\mathbf{M} \in \mathbf{P_{in}}$ it holds

$$\mathbf{Out}(\alpha(\mathbf{M})) = \mathbf{Out}(\beta(\mathbf{M})) .$$

A declarative functionality description should satisfy some of the properties introduced above to exclude pathological examples that cannot be realized by reasoning components as described in section 1.4. On the other hand the notion should not be too restrictive. The following definition will provide such a notion, as will become clear in the rest of this paper.

Definition 3.3

Suppose a signature Σ is given.

a) A *declarative functionality description* for Σ consists of a non-empty set of complete input models $\mathbf{W_{in}}$ for Σ and a mapping $\alpha\mathbf{:}\ \mathbf{P_{in}} \rightarrow \mathbf{P}$, where $\mathbf{P_{in}} = \mathbf{P(W_{in})}$ and $\mathbf{P}$ is a set of partial models for Σ , such that α is conservative and self-bounded.

b) If α is a declarative functionality description, then a *well-informed refinement* of α is a well-informed β such that for all $\mathbf{N} \in \mathbf{W_{in}}$ it holds $\mathbf{Out}(\alpha(\mathbf{N})) = \mathbf{Out}(\beta(\mathbf{N}))$.

If no confusion is expected, for convenience we often omit the word "declarative". An example of a functionality description is the α as constructed in Fig. 12. This can be stated as the following more general Theorem.

Theorem 3.4

Let a non-empty set of complete input models $\mathbf{W_{in}}$ for signature Σ be given and a set $\mathbf{V}$ of partial models of signature Σ such that $\mathbf{W_{in}} \subset \mathbf{P(V)}$. Define the mapping $\alpha\mathbf{:} \mathbf{W_{in}} \to \mathbf{P}$, where $\mathbf{P = P(V)}$, by $\alpha\mathbf{(M) = sc_V(M)}$ for all $\mathbf{M} \in \mathbf{P_{in}}$.

Then α is a declarative functionality description.
In particular this holds if $\mathbf{V}$ is a domain description $\mathbf{W}$, and $\mathbf{W_{in} = In(W)}$.

In fact, the functionality description $\alpha = \mathbf{sc_W}$ additionally satisfies the property of well-informedness. We do not prove this here, since it will follow from more general results later on (Theorem 3.7 and Proposition 3.8). Definition 3.3 allows more functionality descriptions than that one (see section 3.2). But there are restrictions as well. For example it is not possible to express a functionality of a component that makes $\mathbf{h_1}$ true if $\mathbf{s_1}$ is unknown (1) and makes $\mathbf{h_1}$ unknown else (2):

$$\mathbf{< 0 >} \to \mathbf{< 0; u >}$$
$$\mathbf{< 1 >} \to \mathbf{< 1; u >}$$
$$\mathbf{< u >} \to \mathbf{< u; 1 >}$$

Fig. 13 Not a functionality description

This mapping does not satisfy the monotonicity condition: monotonicity would require that (1) implies that $\mathbf{h_1}$ is also true in case $\mathbf{s_1}$ is true or false, which contradicts (2). In fact the conditions of Definition 3.3 imply that it is possible to satisfy the functionality description by an ordinary monotonic deduction system. This will be treated in more detail in section 4.

3.2 Soundness and completeness

Definition 3.3 above does not say anything about how well such a functionality description fits to a given domain description. The example as constructed in section 3.1 does cover the concerning domain description, but a slight change may provide a different functionality description that does not quite fit to the domain description. For example, this is the case if the third line is changed to

$$\mathbf{< 0, u >} \to \mathbf{< 0, u; u, u >}$$

In this section we define additional requirements of soundness and completeness that should be satisfied by a functionality description in order to cover a given domain description.

Proposition 3.5

Suppose a domain description is given by signature Σ and non-empty set of situation models $\mathbf{W}$. Let $\alpha: \mathbf{P_{in}} \rightarrow \mathbf{P}$ be a functionality description, where $\mathbf{P_{in} = P(In(W))}$ and $\mathbf{P} = \mathbf{P(W)}$. The following conditions are equivalent:

(i) For all $\mathbf{M} \in \mathbf{P_{in}}$ each $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ that is true in $\alpha\mathbf{(M)}$ is also true in all complete refinements of $\mathbf{M}$ in $\mathbf{W}$, i.e.
if $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ and $\mathbf{M} \in \mathbf{P_{in}}$ then

$$\alpha\mathbf{(M)} \models \mathbf{h} \;\Rightarrow\; \mathbf{M} \Vdash_{\mathbf{W}} \mathbf{h}$$

(ii) For all $\mathbf{M} \in \mathbf{W_{in}}$ each $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ that is true in $\alpha\mathbf{(M)}$ is also true in all complete refinements of $\mathbf{M}$ in $\mathbf{W}$, i.e.
if $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ and $\mathbf{M} \in \mathbf{W_{in}}$ then

$$\alpha\mathbf{(M)} \models \mathbf{h} \;\Rightarrow\; \mathbf{M} \Vdash_{\mathbf{W}} \mathbf{h}$$

(iii) For all $\mathbf{M} \in \mathbf{W}$ each $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ that is true in $\alpha\mathbf{(In(M))}$ is also true in $\mathbf{M}$, i.e.
if $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ and $\mathbf{M} \in \mathbf{W}$ then

$$\alpha\mathbf{(In(M))} \models \mathbf{h} \;\Rightarrow\; \mathbf{M} \models \mathbf{h}$$

In Definition 3.6 we will use the three equivalent conditions of Proposition 3.5 to define soundness of a functionality description with respect to a given domain description. Similar notions for completeness are only equivalent under the stronger assumption that α is well-informed. Therefore in Definition 3.6 we distinguish between two versions of completeness: a strong notion and a weak one.

Definition 3.6

Suppose a domain description is given by signature Σ and non-empty set of situation models $\mathbf{W}$. Let $\alpha: \mathbf{P_{in}} \rightarrow \mathbf{P}$ be a functionality description, where $\mathbf{P_{in}} = \mathbf{P(W_{in})}$ and $\mathbf{P} = \mathbf{P(W)}$.

a) We call α *sound* with respect to $\mathbf{W}$ if the one of the (equivalent) conditions of Proposition 3.5 is satisfied.

b) We call α *(strongly) complete* with respect to $\mathbf{W}$ if for all $\mathbf{M} \in \mathbf{P_{in}}$ for each $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ that is true in all complete refinements of $\mathbf{M}$ in $\mathbf{W}$, this $\mathbf{h}$ is also true in $\alpha\mathbf{(M)}$, i.e.
if $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ and $\mathbf{M} \in \mathbf{P_{in}}$ then

$$\mathbf{M} \Vdash_{\mathbf{W}} \mathbf{h} \;\Rightarrow\; \alpha\mathbf{(M)} \models \mathbf{h}$$

c) We call α *weakly complete (w-complete)* with respect to $\mathbf{W}$ if for all $\mathbf{M} \in \mathbf{W_{in}}$ for each $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ that is forced by $\mathbf{M}$, this $\mathbf{h}$ is also true in $\alpha\mathbf{(M)}$, i.e.
if $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ and $\mathbf{M} \in \mathbf{W_{in}}$ then

$$\mathbf{M} \Vdash_{\mathbf{W}} \mathbf{h} \;\Rightarrow\; \alpha\mathbf{(M)} \models \mathbf{h}$$

d) If both the conditions a) and b) are satisfied we say that α *covers* $\mathbf{W}$. If both the conditions a) and c) are satisfied we say that α *weakly covers (w-covers)* $\mathbf{W}$.

It is easy to verify that the final functionality description α as constructed in Fig. 12 in section 3.1 covers the given domain description.

$$\begin{array}{lcl}
\mathbf{<0,0>} & \to & \mathbf{<0,0;0,0>} \\
\mathbf{<0,1>} & \to & \mathbf{<0,1;0,1>} \\
\mathbf{<0,u>} & \to & \mathbf{<0,u;u,u>} \\
\mathbf{<1,0>} & \to & \mathbf{<1,0;u,0>} \\
\mathbf{<1,1>} & \to & \mathbf{<1,1;1,1>} \\
\mathbf{<1,u>} & \to & \mathbf{<1,u;u,u>} \\
\mathbf{<u,0>} & \to & \mathbf{<u,0;u,u>} \\
\mathbf{<u,1>} & \to & \mathbf{<u,1;u,u>} \\
\mathbf{<u,u>} & \to & \mathbf{<u,u;u,u>}
\end{array}$$

Fig. 14 Not well-informed functionality description extending Fig. 11

An example where completeness is not satisfied, whereas w-completeness is satisfied is if in the lines concerning incomplete input models we replace in the example in section 3.1 the output truth values in the right hand side by $\mathbf{u}$ (see Fig. 14). This example is in some sense the contrary of a well-informed refinement of α . One could call it badly informed.

The following theorem shows that, given a domain description, there exists a unique regular functionality description that covers it.

Theorem 3.7

Suppose a domain description is given by signature Σ and non-empty set of situation models $\mathbf{W}$. Let $\alpha\mathbf{:}\ \mathbf{P_{in}} \to \mathbf{P}$ be a functionality description, where $\mathbf{P_{in} = P(W_{in})}$ with $\mathbf{W_{in} = In(W)}$ and $\mathbf{P = P(W)}$. Then the following hold:

a) The following conditions are equivalent:

(i) The functionality description α is sound with respect to $\mathbf{W}$

(ii) $\mathbf{Out(\alpha(M)) \leq Out(sc_W(M))}$ for all $\mathbf{M \in P_{in}}$.

(iii) $\alpha\mathbf{(M) \leq sc_W(M)}$ for all $\mathbf{M \in P_{in}}$.

b) The following conditions are equivalent:

(i) The functionality description α is w-complete with respect to $\mathbf{W}$

(ii) $\mathbf{Out}(\alpha(\mathbf{M})) \geq \mathbf{Out}(\mathbf{sc_W}(\mathbf{M}))$ for all $\mathbf{M} \in \mathbf{W_{in}}$.

(iii) $\alpha(\mathbf{M}) \geq \mathbf{sc_W}(\mathbf{M})$ for all $\mathbf{M} \in \mathbf{W_{in}}$.

c) The functionality description α is complete with respect to $\mathbf{W}$ if and only if

$$\mathbf{Out}(\alpha(\mathbf{M})) \geq \mathbf{Out}(\mathbf{sc_W}(\mathbf{M})) \text{ for all } \mathbf{M} \in \mathbf{P_{in}}.$$

d) The following conditions are equivalent:

(i) The functionality description α w-covers $\mathbf{W}$

(ii) $\mathbf{Out}(\alpha(\mathbf{M})) = \mathbf{Out}(\mathbf{sc_W}(\mathbf{M}))$ for all $\mathbf{M} \in \mathbf{W_{in}}$.

(iii) $\alpha(\mathbf{M}) = \mathbf{sc_W}(\mathbf{M})$ for all $\mathbf{M} \in \mathbf{W_{in}}$.

e) The functionality description α covers $\mathbf{W}$ if and only if

$$\mathbf{Out}(\alpha(\mathbf{M})) = \mathbf{Out}(\mathbf{sc_W}(\mathbf{M})) \text{ for all } \mathbf{M} \in \mathbf{P_{in}}.$$

There exists a functionality description that covers $\mathbf{W}$, namely $\mathbf{sc_W}$.

f) If α w-covers $\mathbf{W}$, then the domain description given by $\mathbf{W}$ is empirically founded if and only if for every $\mathbf{M} \in \mathbf{In(W)}$ the model $\alpha(\mathbf{M})$ is complete.

It turns out that the additional condition of well-informedness is strong enough to make w-completeness equivalent to completeness:

Proposition 3.8

Suppose a signature Σ is given with a non-empty set of complete input models $\mathbf{W_{in}}$ and $\mathbf{P}$ is a set of partial models for Σ. Assume the mapping $\alpha\mathbf{:}\ \mathbf{P_{in}} \to \mathbf{P}$, where $\mathbf{P_{in}} = \mathbf{P(W_{in})}$, is a declarative functionality description for Σ. Moreover, let a domain description $\mathbf{W}$ for signature Σ with $\mathbf{In(W)} = \mathbf{W_{in}}$ be given.

Then the following conditions are equivalent:

(i) $\mathbf{W}$ is covered by α

(ii) $\mathbf{W}$ is w-covered by α and α is well-informed.

From Theorem 3.7 and Proposition 3.8 it follows that in the situation of Theorem 3.4 the functionality description given by $\alpha = \mathbf{sc_W}$ is well-informed.

A most simple example of a functionality description not satisfying well-informedness for signature $< \mathbf{s_1}; \mathbf{h_1} >$ is given by the following:

$$< \mathbf{0} > \ \to \ < \mathbf{0; 1} >$$
$$< \mathbf{1} > \ \to \ < \mathbf{1; 1} >$$

$$\langle \mathbf{u} \rangle \rightarrow \langle \mathbf{u}; \mathbf{u} \rangle$$

Fig. 15 Simple example of a not well-informed functionality description

One may raise the question whether or not reasoning components that satisfy this type of functionality description are desirable. In section 4 we will return to this issue.

The question may arise whether for any given functionality description α a domain description can be found that is covered by α; this is the reverse situation of Theorem 3.7 above. According to Theorem 3.7e) this question can be formulated equivalently as: given α, does there exist a $\mathbf{W}$ such that $\mathbf{Out}(\alpha(\mathbf{M})) = \mathbf{Out}(\mathbf{sc}_{\mathbf{W}}(\mathbf{M}))$ for all $\mathbf{M} \in \mathbf{P_{in}}$. It turns out that any α can be expressed in this way if and only if α is well-informed, as is shown by Theorem 3.9.

Theorem 3.9
Suppose a signature Σ is given, $\mathbf{W_{in}}$ is a non-empty set of complete input models and $\mathbf{P}$ is a set of partial models for Σ. Assume $\alpha\mathbf{: P_{in}} \rightarrow \mathbf{P}$ is a declarative functionality description for Σ, where $\mathbf{P_{in}} = \mathbf{P(W_{in})}$.
Then a domain description $\mathbf{W^*}$ for Σ with $\mathbf{In(W^*)} = \mathbf{W_{in}}$ can be obtained that is w-covered by α. One can construct $\mathbf{W^*}$ from α by taking

$$\mathbf{W^* = \{ N \mid N \text{ is a complete model of signature } \Sigma \ \& \ \exists M \in W_{in} \ \ \alpha(M) \leq N\}}.$$

Moreover, $\mathbf{W^*}$ is covered by α if and only if α is well-informed.
For every domain description $\mathbf{W}$ with $\mathbf{In(W)} = \mathbf{W_{in}}$ for which α is sound, $\mathbf{W}$ is contained in $\mathbf{W^*}$.

Corollary 3.10
Suppose a signature Σ is given, $\mathbf{W_{in}}$ is a non-empty set of input models for Σ and $\mathbf{P}$ is a set of partial models for Σ. Assume $\alpha\mathbf{: P_{in}} \rightarrow \mathbf{P}$, where $\mathbf{P_{in}} = \mathbf{P(W_{in})}$, is a declarative functionality description for Σ.
Then there is a well-informed declarative functionality description $\beta\mathbf{: P_{in}} \rightarrow \mathbf{P}$ that is a well-informed refinement of α. This β is better informed than α; in particular, it holds

$$\mathbf{Out}(\alpha(\mathbf{M})) \leq \mathbf{Out}(\beta(\mathbf{M})) = \mathbf{Out}(\mathbf{sc}_{\mathbf{W^*}}(\mathbf{M}))$$

for all $\mathbf{M} \in \mathbf{P_{in}}$, with $\mathbf{W^*}$ as in Theorem 3.9.

Not surprisingly, the well-informed declarative functionality description β related to a given α as obtained in Corollary 3.10, sometimes is called *the well-informed refinement* of α.

4 Specifications of Interactive Reasoning Components

The examples of functionality descriptions given in section 3 are defined by enumerating complete tables for the mappings. In the context of systems that acquire their knowledge by learning from examples (cases), this may have some practical relevance: the tables may be used as a representation of the cases that were encountered in the past, and the functionality embodied by this history. However, in practical situations concerning knowledge-based systems that do not learn from examples, tables are not an efficient manner of specification. Therefore a more condensed form of specification is needed. This will be treated in this section.

As already sketched in section 2, the declarative aspects of a reasoning component are determined by the specification of a knowledge base that enables the component, using a suitable fixed inference relation, to derive new (output) facts from given input facts. In section 4.1 we make a choice on the format in which the knowledge base is specified. We will leave the inference relation unspecified. Instead here we will define a suitable notion of a semantic consequence relation. In principle a choice of a strict format for the knowledge implies a restriction on the expressiveness. However, we will prove in section 4.3 that for any relevant well-informed functionality description a knowledge base specification in the chosen format is possible such that by the semantic consequence relation the required (declarative) functionality is obtained. This means that any derivability relation that is sound and complete with respect to this semantic consequence relation is able to derive from a given input information state by use of the knowledge base the right conclusions. In another report it will be discussed that chaining provides such a suitable derivability relation.

4.1 Some definitions

By $\mathbf{KB}$ (the *knowledge base*) we denote the knowledge which may be used by the reasoning component to derive output literals from the available information on inputs. Recall that $\mathbf{P_{in}}$ is the non-empty set of all possible partial input models. If $\mathbf{M} \in \mathbf{P_{in}}$, and $\mathbf{c}$ is a conjunction of literals, then by $\mathbf{M} \models_{\mathbf{KB}} \mathbf{c}$ we will denote that $\mathbf{c}$ *semantically follows* from the information of $\mathbf{M}$ by use of $\mathbf{KB}$, i.e. is a semantic consequence of the theory $\mathbf{Lit(M)} \cup \mathbf{KB}$. The notions of rule-format, semantic consequence and reasoning component specification can be defined formally as follows:

Definition 4.1

Let Σ be a signature and $\mathbf{KB}$ a set of propositions for Σ.

a) A proposition in *rule-format*, or simply a *rule* is a proposition of one of the following two forms:

(i) $\mathbf{d}$ where $\mathbf{d}$ is a literal; these rules are sometimes called *general facts*

(ii) $\mathbf{c} \to \mathbf{d}$ where $\mathbf{c}$ is a conjunction of literals and $\mathbf{d}$ is a literal

b) We call $\mathbf{KB}$ *consistent* with respect to the input model $\mathbf{M}$ if there exists a model $\mathbf{N}$ with $\mathbf{M} \leq \mathbf{N}$ and $\mathbf{N} \models \mathbf{KB}$.

c) If $\mathbf{KB}$ is consistent with respect to the input model $\mathbf{M}$ then the *semantic consequence* relation $\mathbf{M} \models_{\mathbf{KB}} \mathbf{c}$ is defined as: for all models $\mathbf{N}$ for Σ with $\mathbf{M} \leq \mathbf{N}$ and $\mathbf{N} \models \mathbf{KB}$ it holds $\mathbf{N} \models \mathbf{c}$.

d) A *(declarative) reasoning component specification* consists of a finite signature Σ , a finite non-empty set of rules for this signature $\mathbf{KB}$ (knowledge base), and a finite set of input models $\mathbf{P_{in}} = \mathbf{P(W_{in})}$ where $\mathbf{W_{in}}$ is a non-empty set of complete input models for the related input signature.

Notice that the fact that we restrict the rule format to one conclusion only is not an essential restriction: every implication $\mathbf{c} \to \mathbf{d}$ where both $\mathbf{c}$ and $\mathbf{d} = \mathbf{d_1} \wedge \ldots \wedge \mathbf{d_k}$ are conjunctions of literals can simply be rewritten to a set of rules $\mathbf{c} \to \mathbf{d_1}, \ldots, \mathbf{c} \to \mathbf{d_k}$ in the sense of Definition 4.1a).

The notion of consistency can be tested for a given subset of $\mathbf{P_{in}}$; however, this does not guarantee that the component reasons sound with respect to a functionality description or domain description. This (stronger) notion of soundness will be explained further in section 4.2.

For a reasoning component that is consistent with respect to $\mathbf{M} \in \mathbf{P_{in}}$ we define the *consequence model* $\mathbf{cons_{KB}(M)}$ of $\mathbf{M}$ as the partial model where all literals that semantically follow are true, and the others are unknown, i.e. for all atoms $\mathbf{a}$ it holds

$$\mathbf{cons_{KB}(M)(a)} = \begin{cases} \mathbf{0} & \text{if } \mathbf{M} \models_{\mathrm{KB}} \neg\, \mathbf{a} \\ \mathbf{1} & \text{if } \mathbf{M} \models_{\mathrm{KB}} \mathbf{a} \\ \mathbf{u} & \text{else} \end{cases}$$

It is easy to verify that

$$\mathbf{M} \models_{\mathrm{KB}} \mathbf{a} \iff \mathbf{M} \mid\!\models_{\mathbf{Mod(KB)}} \mathbf{a}$$

and

$$\mathbf{cons_{KB}(M)} = \mathbf{sc_{Mod(KB)}\,(M)}$$

where $\mathbf{Mod(KB)}$ is the set of all models of $\mathbf{KB}$.

Lemma 4.2
Suppose a signature Σ is given and $\mathbf{W_{in}}$ is a non-empty set of input models. Take for $\mathbf{P}$ the set of all partial models of signature Σ and $\mathbf{P_{in}} = \mathbf{P(W_{in})}$. Let a consistent reasoning component specification for Σ and $\mathbf{P_{in}}$ be given by $\mathbf{KB}$.
Then the mapping $\mathbf{cons_{KB} : P_{in} \to P}$ given by $\mathbf{M} \to \mathbf{cons_{KB}(M)}$ is a well-informed (declarative) functionality description. Moreover, $\mathbf{cons_{KB} = sc_{Mod(KB)}}$.

Definition 4.3
Suppose a signature Σ is given and $\mathbf{W_{in}}$ is a non-empty set of input models. Take for $\mathbf{P}$ the set of all partial models of signature Σ and $\mathbf{P_{in}} = \mathbf{P(W_{in})}$. Let a consistent reasoning component specification for Σ and $\mathbf{P_{in}}$ be given by $\mathbf{KB}$.
a) We call $\mathbf{cons_{KB}}$ *the well-informed functionality description related to* (or *specified by*) the given reasoning component specification, or if no confusion is expected we simply call it the *well-informed functionality description related to* (or *specified by*) $\mathbf{KB}$.
b) We call two reasoning component specifications with the same set of input models $\mathbf{P_{in}}$ *equivalent* if they specify the same functionality description.
c) We say a functionality description α is *covered by* the well-informed functionality description related to the reasoning component specification given by $\mathbf{KB}$ if for all $\mathbf{M} \in \mathbf{P_{in}}$ it holds $\mathbf{Out(\alpha\ (M))} = \mathbf{Out(cons_{KB}\ (M))}$.

4.2 Soundness and completeness

In this section we give precise definitions of soundness and completeness of the well-informed functionality description related to a reasoning component specification with respect to a given domain description. In view of Definition 4.3 this can be done very easily:

Definition 4.4
Suppose a signature Σ is given and $\mathbf{W_{in}}$ is a non-empty set of input models and $\mathbf{W}$ a domain description for Σ and $\mathbf{W_{in}}$. Let a consistent reasoning component specification for Σ and $\mathbf{P_{in}}$ be given by $\mathbf{KB}$.
The quality of a reasoning component specification with respect to a given domain description can be expressed by respectively *soundness*, *completeness*, *w-completeness, covering* with respect to the given domain description of the well-informed functionality description related to the reasoning component specification.

It is easy to verify that the well-informed functionality description related to a reasoning component specification given by $\mathbf{KB}$ is sound with respect to $\mathbf{W}$ if and only if every model $\mathbf{M} \in \mathbf{W}$ is a model of $\mathbf{KB}$. Collecting this, together with the connections formulated in Definition 3.6, Theorem 3.7 and Proposition 3.8, we obtain the following statements for these notions:

Proposition 4.5

Suppose Σ is a signature, and a domain description is given by $\mathbf{W}$ and a non-empty set of input models $\mathbf{W_{in}}$. Let a consistent reasoning component specification for Σ and $\mathbf{P_{in}} = \mathbf{P(W_{in})}$ be given by $\mathbf{KB}$, and α the well-informed functionality description related to the reasoning component specification given by $\mathbf{KB}$.

a) The following conditions are equivalent:

(i) α is sound with respect to the domain description given by $\mathbf{W}$.

(ii) The output literals, which semantically follow using the knowledge base $\mathbf{KB}$ from input literals which are true in a given input model $\mathbf{M} \in \mathbf{P_{in}}$, are true in all complete refinements of $\mathbf{M}$ in $\mathbf{W}$, i.e.
if $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ and $\mathbf{M} \in \mathbf{P_{in}}$ then

$$\mathbf{M} \models_{\mathrm{KB}} \mathbf{h} \;\Rightarrow\; \mathbf{M} \Vdash_{\mathbf{W}} \mathbf{h}$$

(iii) The output literals, which semantically follow using the knowledge base $\mathbf{KB}$ from input literals which are true in a given complete input model $\mathbf{M} \in \mathbf{W_{in}}$, are true in all complete refinements of $\mathbf{M}$ in $\mathbf{W}$, i.e.
if $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ and $\mathbf{M} \in \mathbf{W_{in}}$ then

$$\mathbf{M} \models_{\mathrm{KB}} \mathbf{h} \;\Rightarrow\; \mathbf{M} \Vdash_{\mathbf{W}} \mathbf{h}$$

(iv) For all $\mathbf{M} \in \mathbf{W}$ each $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ that semantically follow using the knowledge base $\mathbf{KB}$ from input literals which are true in $\mathbf{M}$, is true in $\mathbf{M}$, i.e.
if $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ and $\mathbf{M} \in \mathbf{W}$ then

$$\mathbf{In(M)} \models_{\mathrm{KB}} \mathbf{h} \;\Rightarrow\; \mathbf{M} \models \mathbf{h}$$

(v) For all $\mathbf{M} \in \mathbf{P_{in}}$ it holds

$$\mathbf{cons_{KB}(M)} \leq \mathbf{sc_{W}(M)}$$

(vi) Every $\mathbf{M} \in \mathbf{W}$ is a model of $\mathbf{KB}$.

b) The following conditions are equivalent:

(i) α is complete with respect to $\mathbf{W}$.

(ii) α is w-complete with respect to $\mathbf{W}$.

(iii) For any input model $\mathbf{M} \in \mathbf{P_{in}}$ and for any output literal $\mathbf{h}$ which is true in all complete refinements of $\mathbf{M}$ in $\mathbf{W}$, this $\mathbf{h}$ semantically follows from $\mathbf{M}$ using $\mathbf{KB}$, i.e. if $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ and $\mathbf{M} \in \mathbf{P_{in}}$ then

$$\mathbf{M} \Vdash_{\mathbf{W}} \mathbf{h} \quad \Rightarrow \quad \mathbf{M} \models_{\mathrm{KB}} \mathbf{h}$$

(iv) For all $\mathbf{M} \in \mathbf{P_{in}}$ it holds

$$\mathbf{Out(cons_{KB}(M))} \;\geq\; \mathbf{Out(sc_W(M))}.$$

(v) For any complete input model $\mathbf{M} \in \mathbf{W_{in}}$ and for any output literal $\mathbf{h}$ which is true in all complete refinements of $\mathbf{M}$ in $\mathbf{W}$, this $\mathbf{h}$ semantically follows from $\mathbf{M}$ using $\mathbf{KB}$, i.e. if $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ and $\mathbf{M} \in \mathbf{W_{in}}$ then

$$\mathbf{M} \Vdash_{\mathbf{W}} \mathbf{h} \quad \Rightarrow \quad \mathbf{M} \models_{\mathrm{KB}} \mathbf{h}$$

(vi) For all $\mathbf{M} \in \mathbf{W_{in}}$ it holds

$$\mathbf{cons_{KB}(M)} \;\geq\; \mathbf{sc_W(M)}.$$

c) The following holds: α covers $\mathbf{W}$ if and only if

$$\mathbf{Out(cons_{KB}(M))} \;=\; \mathbf{Out(sc_W(M))} \text{ for all } \mathbf{M} \in \mathbf{P_{in}}.$$

In practice, in a reasoning component specification often the set of input models $\mathbf{P_{in}}$ is not mentioned. We will interpret this omission as if this set of input models is meant to be the set of all partial models for the input signature (all truth assignments that are theoretically possible). However, in practical domains, often not all theoretically possible input models are used, for instance since there are semantical dependencies between the input atoms. In these cases there may be theoretically possible input models that do not make sense in reality, and especially, there is no (domain) knowledge on what output should be expected for these input models. This means that in such a domain it is *essentially* impossible to prove soundness and completeness, as long as no restriction is put on the set of input models.

On the other hand, in practice it is often unfeasable to enumerate the set of all relevant input models, so in any case *practical* problems can be expected in proving soundness and completeness. What can be done is to collect (during knowledge acquisition) a set of typical, and critical examples of input models, and use this as a representative test set. Another possible approach is, as a part of the knowledge acquisition process, to make explicit all semantic dependencies between input atoms, and use these as constraints to specify $\mathbf{P_{in}}$. This approach has not been tried out yet.

4.3 Existence of reasoning component specifications

In this section we show that for any well-informed functionality description α a reasoning component specification can be found such that its related well-informed functionality description gives the same reults as α, and that this can be done in a minimal sense. To illustrate these issues we return to the example in section 3.1 of a domain description given by $\mathbf{W}$ and a well-informed functionality description α, given by Fig. 12. If we involve only complete input models we may create from α a knowledge base. This can be done by simply taking for each complete input model the conjunction of all input literals that are true, and use this as the condition of a rule, while the conclusion is given by the image under α. This results in the knowledge base of Fig. 16.

This knowledge base $\mathbf{KB}$ is consistent and satisfies $\mathbf{Out(\alpha(M))} \geq \mathbf{Out(cons_{KB}(M))}$ for all $\mathbf{M} \in \mathbf{P_{in}}$. But this is not an adequate specification, since in the first place using it one cannot conclude anything in a clear and direct manner from incomplete input models: some of the rules are more complicated than is needed. Secondly, it contains too many rules: it contains more rules than are needed, as we will see later on.

$\neg s_1 \wedge \neg s_2$	$\rightarrow$	$\neg h_1$
$\neg s_1 \wedge s_2$	$\rightarrow$	$\neg h_1$
$s_1 \wedge s_2$	$\rightarrow$	h_1
$\neg s_1 \wedge \neg s_2$	$\rightarrow$	$\neg h_2$
$\neg s_1 \wedge s_2$	$\rightarrow$	h_2
$s_1 \wedge \neg s_2$	$\rightarrow$	$\neg h_2$
$s_1 \wedge s_2$	$\rightarrow$	h_2

Fig. 16 Knowledge base weakly covering α

The first problem can be solved by adding to this knowledge base rules that are based on incomplete input models as well. Consider, for example, the third line of the functionality description α in Fig. 12:

$$< \mathbf{0, u} > \;\rightarrow\; < \mathbf{0, u; 0, u} >$$

If $\mathbf{M}$ is any model in $\mathbf{P_{in}}$ refining the left hand side (i.e. $< \mathbf{0, u} > \leq \mathbf{M}$), then by monotonicity

$$< \mathbf{0, u; 0, u} > \;=\; \alpha(< \mathbf{0, u} >) \;\leq\; \alpha(\mathbf{M})$$

Therefore for any $\mathbf{M} \in \mathbf{P_{in}}$ it holds:

$$\textit{if} \;\; < \mathbf{0, u} > \leq \mathbf{M} \;\; \textit{then} \;\; < \mathbf{0, u; 0, u} > \;\leq\; \alpha(\mathbf{M})$$

Since $< \mathbf{0, u} > \leq \mathbf{M}$ is equivalent to $\mathbf{M} \models \neg \mathbf{s_1}$ and $< \mathbf{0, u; 0, u} > \;\leq\; \alpha(\mathbf{M})$ is equivalent to $\mathbf{M} \models \neg \mathbf{s_1}$ and additionally $\alpha(\mathbf{M}) \models \neg \mathbf{h_1}$, we can restate the above *if-then* rule by the rule $\neg \mathbf{s_1} \rightarrow \neg \mathbf{h_1}$. Doing this for all relevant lines of Fig. 12, this results in the extension of $\mathbf{KB}$ given in Fig. 17.

By this knowledge base we obtain a reasoning component specification that is able to conclude from partial input information in a direct manner. But the second problem has become worse: the number of rules has even increased to 10: this is still a very inefficient specification.

$\neg s_1$	$\rightarrow$	$\neg \mathbf{h_1}$
$\neg s_2$	$\rightarrow$	$\neg \mathbf{h_2}$
s_2	$\rightarrow$	$\mathbf{h_2}$

Fig. 17 Extension of the knowledge base of Fig. 16 covering α

However, sometimes a number of the earlier rules are a special case of one new rule with less conditions. For instance, it is easy to see that the 10th rule makes the rules 5 and 7 superfluous, since both the complete input models related to the rules 5 and 7 are refinements of the incomplete input model related to rule 10. This enables us to prune the knowledge base until we obtain a minimal form for it. Comparatively, in practice knowledge bases are acquired from experts who have streamlined and minimized the storage of their knowledge in the past.

Here, in our example it can be shown how such a minimization can be done. For example, instead of considering all partial input models $\mathbf{M} \in \mathbf{P_{in}}$ for which $\alpha(\mathbf{M}) \models \mathbf{h_1}$, we only take the $\mathbf{M} \in \mathbf{P_{in}}$ among them that are minimal in $\mathbf{P_{in}}$, i.e. such that there does not exist an $\mathbf{M'} \in \mathbf{P_{in}}$ with $\alpha(\mathbf{M}) \models \mathbf{h_1}$ such that $\mathbf{M'} \leq \mathbf{M}$ and $\mathbf{M'} \neq \mathbf{M}$. Inspecting $\mathbf{P_{in}}$, for each of the four output literals $\mathbf{h_1}, \neg \mathbf{h_1}, \mathbf{h_2}, \neg \mathbf{h_2}$ we find one non-trivial minimal element. Using these we obtain the following more concise knowledge base (which is a subset of the knowledge base above):

$s_1 \wedge s_2$	$\rightarrow$	$\mathbf{h_1}$
$\neg s_1$	$\rightarrow$	$\neg \mathbf{h_1}$
$\neg s_2$	$\rightarrow$	$\neg \mathbf{h_2}$
s_2	$\rightarrow$	$\mathbf{h_2}$

Fig. 18 Minimal knowledge base covering α

By this knowledge base a specification is obtained such that its related well-informed functionality description has the same output models as α (it covers α) and which is minimal in the sense that will be defined below more precisely.

In the context of the example above we gave an informal description of a construction showing that for a given well-informed functionality description α there exists a reasoning component specification such that its related well-informed functionality description is covering α. Summarized, the procedure can be described using truth tables as shown below. Here a right hand side of the functionality description is interpreted as a truth-table (by example we take the one of Fig. 12):

s_1	s_2	h_1	h_2	$T(h_2)$	$mT(h_2)$	
0	0	0	0			(1)
0	1	0	1	+	×	(2)
0	u	0	u			(3)
1	0	u	0			(4)
1	1	1	1	+	×	(5)
1	u	u	u			(6)
u	0	u	0			(7)
u	1	u	1	+	V	(8)
u	u	u	u			(9)

Fig. 19 Truth table representing the functionality description of Fig 12

The lines correspond to partial output models as described by the well-informed functionality description, and the columns correspond to the truth values for the atom mentioned above it. The procedure runs as follows:

Minimal knowledge base construction

1. Select one of the output literals;
   as an example we choose $\mathbf{h_2}$
2. Collect all lines in the table where the $\mathbf{h_2}$-column shows that $\mathbf{h_2}$ is true (the set $\mathbf{T(h_2)}$);
   this results in the lines 2, 5, 8
3. Delete from this list the ones that are refinements of other lines (the set $\mathbf{mT(h_2)}$);

here the lines 2 and 5 are deleted since the input parts of them are refinements of the input part of line 8

4. For each of the remaining lines, construct a rule with as condition part the conjunction of input literals that are true and with $\mathbf{h_2}$ as conclusion;
   in our example this results in the rule $\mathbf{s_2} \rightarrow \mathbf{h_2}$
5. Repeat this procedure for all other output literals;
   so do the same for $\mathbf{h_1}, \neg\, \mathbf{h_1}, \neg\, \mathbf{h_2}$.

We will define this construction more formally and prove that it provides indeed a specification covering α . Moreover, we will prove that it results in a minimal specification. The example and procedure described above may help to understand the ideas behind the formal approach below. Notice that this is an extension to partial logic of the wellknown theorem from propositional logic, stating that any boolean function can be expressed as a combination of negation, conjunction and disjunction.

Definition 4.5

Let a consistent reasoning component specification be given by signature Σ, a knowledge base $\mathbf{KB}$, and non-empty set of partial input models $\mathbf{P_{in}}$. Then it is called *minimal* if for every rule in $\mathbf{KB}$, and every generalization of it by omitting one of the conditions, replacing the rule by its generalization makes a knowledge base that is not equivalent to $\mathbf{KB}$.

Lemma 4.6

Assume a signature Σ is given, $\mathbf{W_{in}}$ is a non-empty set of complete input models and $\mathbf{P}$ a set of models for Σ . Assume $\alpha : \mathbf{P_{in}} \rightarrow \mathbf{P}$ where $\mathbf{P_{in}} = \mathbf{P(W_{in})}$ is a functionality description. The non-empty set of rules $\mathbf{KB_\alpha}$ is constructed as follows. For each output literal $\mathbf{h}$, take

$$\mathbf{T(h)} = \{ \mathbf{M} \in \mathbf{P_{in}} \mid \alpha(\mathbf{M}) \models \mathbf{h} \}$$

and $\mathbf{mT(h)}$ the set of minimal (with respect to the refinement relation) elements in $\mathbf{T(h)}$.
Define $\mathbf{KB_\alpha}$ by the following set of rules

$$\mathbf{KB_\alpha} = \{ \mathbf{h} \mid \Lambda \in \mathbf{mT(h)} \} \cup \{ \mathbf{Con(Lit(M))} \rightarrow \mathbf{h} \mid \mathbf{M} \in \mathbf{mT(h)}, \mathbf{M} \neq \Lambda \}$$

where $\mathbf{Con(..)}$ means taking the conjunction of a set of literals.
Then a consistent reasoning component specification is obtained with signature Σ, knowledge base $\mathbf{KB_\alpha}$, and set of partial input models $\mathbf{P_{in}}$ such that for all complete models $\mathbf{M} \in \mathbf{W_{in}}$ it holds $\alpha(\mathbf{M}) \models \mathbf{KB_\alpha}$.

Theorem 4.7

Assume a signature Σ is given, $\mathbf{P_{in}}$ is a non-empty set of partial input models for Σ and let $\alpha : \mathbf{P_{in}} \rightarrow \mathbf{P(W)}$ be a well-informed functionality description.
Then the well-informed functionality description related to the consistent reasoning component specification with signature Σ, knowledge base $\mathbf{KB_\alpha}$, and a set of partial input models $\mathbf{P_{in}}$ covers α. This reasoning component specification is minimal.

Applying the construction of Lemma 4.6 above to the not well-informed functionality description of Fig. 15 gives the knowledge base of Fig. 20. Notice that here two different minimal elements occur in the set $\mathbf{T(h_1)}$.

$$\mathbf{s_1} \rightarrow \mathbf{h_1}$$
$$\neg\, \mathbf{s_1} \rightarrow \mathbf{h_1}$$

Fig. 20 Knowledge base based on a not well-informed functionality description.

A reasoning component specified by this knowledge base $\mathbf{KB_1}$ has $\mathbf{h_1}$ as a semantic consequence (independent of whether or not input information is given). However, the original functionality description gives as an output the truth value **unknown** to $\mathbf{h_1}$ if the truth value **unknown** of $\mathbf{s_1}$ is given as an input. This illustrates the plain fact that any functionality description that is not well-informed cannot be covered by the well-informed functionality description related to any knowledge base. The reasoning component is able to derive $\mathbf{h_1}$ from $\mathbf{KB_1}$ by use of any complete inference relation such as resolution or natural deduction. But using chaining $\mathbf{KB_1}$ is not able to derive $\mathbf{h_1}$ if nothing is known about $\mathbf{s_1}$.

This contrasts with the knowledge base $\mathbf{KB_2}$ just consisting of the general fact $\mathbf{h_1}$; with $\mathbf{KB_2}$ a component using any complete inference relation derives the same conclusions as with $\mathbf{KB_1}$. With chaining as an inference relation this time it is also able to derive always $\mathbf{h_1}$. It turns out that using chaining we are able to distinguish the two different functionality descriptions as discussed by the two knowledge bases $\mathbf{KB_1}$ and $\mathbf{KB_2}$ where other (so-called complete) inference relations are not able to distinguish a functionality description from its well-informed refinement. Since this concerns distinctions that are relevant in practice, this phenomenon will be analysed in more detail in another report.

The following Proposition describes the case of empirically founded domains.

Proposition 4.8

Let a domain description for signature Σ and a non-empty set of input models $\mathbf{W_{in}}$ be given by $\mathbf{W}$. Suppose α is a declarative functionality description w-covering $\mathbf{W}$ and $\mathbf{KB}$ defines a reasoning component specification such that its related well-informed functionality description covers α.
The following conditions are equivalent:
(i) The domain described by $\mathbf{W}$ is empirically founded
(ii) For every $\mathbf{M} \in \mathbf{W_{in}}$ the model $\alpha\mathbf{(M)}$ is complete
(iii) For every $\mathbf{M} \in \mathbf{W_{in}}$ and output literal $\mathbf{h}$ it holds either $\mathbf{M} \models_{\mathbf{KB}} \mathbf{h}$ or $\mathbf{M} \models_{\mathbf{KB}} \neg \mathbf{h}$
(iv) For any output literal $\mathbf{h}$ there exists a proposition $\mathbf{p}$ in terms of input literals such that $\mathbf{h}$ is true in a situation $\mathbf{M} \in \mathbf{W}$ if and only if $\mathbf{p}$ is true in $\mathbf{M}$ (*explicit definability*), i.e.: for every $\mathbf{M} \in \mathbf{W}$ it holds

$$\mathbf{M} \models \mathbf{h} \Leftrightarrow \mathbf{M} \models \mathbf{p}$$

5 Applications

In this section it is shown how the semantical framework introduced in this paper can be applied to a process model for diagnostic reasoning (see [6]) . The processes at different abstraction levels of this generic diagnostic model are given in Fig. 21. 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.

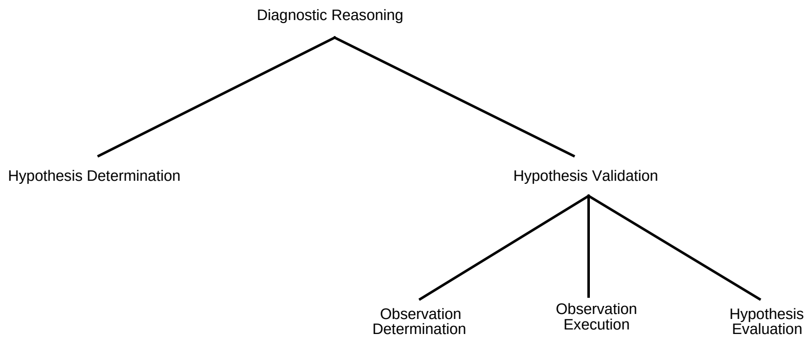

Fig. 21 Processes at different abstraction levels in the diagnostic process model

In [6] 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 [6]. To apply the generic model, instantiations of these primitive components are needed, for example selected from a library of components. The functionality of these components can be described by functionality descriptions. However, using the approach of [6], the model as a whole is only guaranteed to work properly if the properties of the primitive components presented in [6] are in some sense satisfied by these functionality descriptions. In this section we shown how these properties can be formulated as properties of functionality descriptions.

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 [6] 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:

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

$$[\ \textbf{state}_{\textbf{HD}}(\mathcal{M}, \textbf{t}, \textbf{input(HD)}) \models \textbf{assessed(h)} \Rightarrow\ \textbf{state}_{\textbf{HD}}(\mathcal{M}, \textbf{t}, \textbf{output(HD)}) \mid\neq \textbf{focus(h)}\ ]$$

Suppose a candidate component to be used for Hypothesus Determination is described by functionality description α. Then the above property can be reformulated to the following property of α:

$$\textbf{M} \models \textbf{assessed(h)} \Rightarrow\ \alpha\textbf{(M)} \mid\neq \textbf{focus(h)}\ ]$$

The second proerty 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 [6] 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:

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

$[\exists h\ \mathrm{state}_{HD}(\mathcal{M}, t, \mathrm{input(HD)}) \mid\neq \mathrm{assessed}(h) \wedge \forall h\ \mathrm{state}_{S}(\mathcal{M}, t, \mathrm{input(HD)}) \mid\neq \mathrm{confirmed}(h)]$

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

This property can be reformulated to the following property of α:

$[\ \exists h\ M\ \mid\neq \mathrm{assessed}(h)\ \&\ \forall h\ M \mid\neq \mathrm{confirmed}(h)]\ \Rightarrow\ \exists h'\ \alpha(M) \models \mathrm{focus}(h')$

In conclusion, a component can be chosen to play the role of Hypothesis Determination in the diagnostic model if these two properties hold for its functionality description.

In a similar manner the properties 'observation effectiveness' and 'observation efficiency' of the component Observation Determination reduce to static properties of functionality descriptions. Observation efficiency means that no observations are generated that already were performed:

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

$[\ \mathrm{state}_{OD}(\mathcal{M}, t, \mathrm{input(OD)}) \models \mathrm{observed}(o) \Rightarrow\ \mathrm{state}_{OD}(\mathcal{M}, t, \mathrm{output(OD)}) \mid\neq \mathrm{to\_be\_observed}(o)\ ]$

This is reformulated as

$M \models \mathrm{observed}(o) \Rightarrow\ \alpha(M) \mid\neq \mathrm{to\_be\_observed}(o)\ ]$

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.

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

$[\exists o\ \ \mathrm{state}_{OD}(\mathcal{M}, t, \mathrm{input(OD)}) \mid\neq \mathrm{observed}(o)$

$\wedge\ \mathrm{state}_{OD}(\mathcal{M}, t, \mathrm{input(OD)}) \models \mathrm{focus}(h) \Rightarrow$

$\exists o'\ \mathrm{state}_{OD}(\mathcal{M}, t, \mathrm{output(OD)}) \models \mathrm{to\_be\_observed}(o')\ ]$

This is reformulated as

$[\ \exists o,h\ \ M\ \mid\neq \mathrm{observed}(o)\ \&\ M \models \mathrm{focus}(h)]\ \Rightarrow\ \exists o'\ \ \alpha(M) \models \mathrm{to\_be\_observed}(o')$

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:

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

$[\forall o\ [\ \mathrm{state}_{HE}(\mathcal{M}, t, \mathrm{input(HE)}) \models o\ \vee\ \mathrm{state}_{HE}(\mathcal{M}, t, \mathrm{input(HE)}) \models \neg\ o\ ] \Rightarrow$

$\forall h\ [\ \mathrm{state}_{HE}(\mathcal{M}, t, \mathrm{output(HE)}) \models h\ \vee\ \mathrm{state}_{HE}(\mathcal{M}, t, \mathrm{output(HE)}) \models \neg\ h\ ]$

This can be reformulated for a functionality description as:

$$\forall \mathbf{o}\ [\ \mathbf{M} \models \mathbf{o} \ \vee\ \mathbf{M} \models \neg\, \mathbf{o}\ ] \ \Rightarrow\ \forall \mathbf{h}\ [\ \alpha(\mathbf{M}) \models \mathbf{h} \ \vee\ \alpha(\mathbf{M}) \models \neg\, \mathbf{h}\ ]$$

This property is a special case of a property that is sometimes called decisiveness: The functionality description α is called decisive if for all complete input models, also the generated model is complete:

$$\mathbf{M} \in \mathbf{W}_{\mathbf{in}} \Rightarrow \alpha(\mathbf{M}) \text{ is complete}$$

In this section it was shown in an example (based on [6]) how required properties of candidate primitive components for a generic model can be formulated as properties of their functionality description which specifies their functionality independent of specific details of internal knowledge representation, inference relations or implementation.

6 Conclusions

This paper contributes a semantical framework that provides a logical description of the functionality of interactive reasoning. The concept of functionality abstracts from specific inference relations or knowledge representation. A number of properties of a functionality description are identified, and related to (formalized) characteristics of the domain. It is characterised under which conditions a functionality description can be implemented by an inference relation and a knowledge base.

It turns out that our semantical framework may provide adequate logical descriptions for the functionality of an interactive reasoning component. In particular the relation of the conclusions that may be drawn by a component and the situation in reality that is concerned may be made more transparant by our framework. Furthermore, the formal definitions of soundness, completeness and empirically foundedness as given above enable us to establish the (meta-)logical connections between these concepts.

The semantic formalisation using information states can also be exploited to formalise the dynamics of a reasoning process. The intermediate reasoning steps can be formalised as traces of information states, as briefly sketched in Section 1.2. To specify the dynamics of such traces variants of temporal logic can be used. For specific classes of nonmonotonic reasoning methods this has been worked out in [7], [8], [9].

The semantic framework for reasoning components introduced here can be incorporated in a semantic formalisation of a compositional reasoning system, as presented in [4]. In [4] the semantic formalisation of the functionality of a primitive component was left open; it was taken as an assumed building block on top of which the compositional dynamics were defined semantically. The current paper fills this gap by providing a semantic formalisation of the building block. In [12] some results can be found on compositioon verification of agent-based reasoning systems. In this paper it has been addressed how properties of primitive components are related to (dynamic) properties of a compositional reasoning system as a whole.

In Section 5 above it has been shown how (required) properties of primitive components within a compositional system can be related to properties of declarative functionality descriptions of candidate components. Since these properties abstract from specific knowledge representation or inference, reuse and maintainability is supported: whatever is changed within such a candidate component, it does not matter as long as the functionality description remains the same or at least has the same relevant properties (information hiding). Other future investigations are planned in the relation between the semantical approach introduced above and recent work on input-output logics (cf. [15]).

Acknowledgements
The research reported here has benefit from discussions with Catholijn Jonker, Pieter van Langen, Izak van Langevelde and Leon van der Torre.

References

1. S. Blamey, Partial Logic, in: D. Gabbay and F. Guenthner (eds.), Handbook of Philosophical Logic, Vol. III, 1-70, Reidel, Dordrecht, 1986.
2. Brazier, F.M.T., Jonker, C.M., and Treur, J., 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.
3. Brazier, F. M. T., Jonker, C. M., Treur, J., and Wijngaards, N.J.E, (2000), On the Use of Shared Task Models in Knowledge Acquisition, Strategic User Interaction and Clarification Agents. *International Journal of Human-Computer Studies,* vol. 52, 2000, pp. 77-110.
4. Brazier, F.M.T., Treur, J., Wijngaards, N.J.E. and Willems, M., Temporal semantics of compositional task models and problem solving methods. *Data and Knowledge Engineering,* vol. 29(1), 1999, pp. 17-42.
5. C.C. Chang, H.J. Keisler, Model theory, North Holland, 1973

6. Cornelissen, F., Jonker, C.M., and Treur, J., 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.
7. Engelfriet, J., and Treur, J., Temporal Theories of Reasoning. *Journal of Applied Non-Classical Logics*, 5, 1995, pp. 239-261.
8. Engelfriet J., Treur J. Executable Temporal Logic for Nonmonotonic Reasoning. *Journal of Symbolic Computation*, vol. 22, 1996, pp. 615-625.
9. Engelfriet J., and Treur J. Specification of Nonmonotonic Reasoning. *Journal of Applied Non-Classical Logics*, vol. 10, 2000, pp. 7-27
10. E. Giunchiglia, P. Traverso and F. Giunchiglia, Multi-context Systems as a Specification framework for Complex Reasoning Systems, In: J. Treur and Th. Wetter (eds.), Formal Specification of Complex Reasoning Systems, Ellis Horwood, 1993, pp. 45-72.
11. C.G. Hempel, Philosophy of Science, Prentice-Hall, Englewoods Cliffs, 1966
12. Jonker, C.M. and Treur, J., 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.
13. P.H.G. van Langen and J. Treur, Representing world situations and information states by many-sorted partial models, Report PE8904, University of Amsterdam, Department of Mathematics and Computer Science, 1989.
14. T. Langholm, Partiality, Truth and Persistance, CSLI Lecture Notes No. 15, Stanford University, Stanford, 1988.
15. D. Makinson, L. van der Torre, Input/output logics. *Journal of Philosophical Logic* 29 (2000) 383-408.
16. J. Treur, Completeness and definability in diagnostic expert systems. Proc. European Conference on Artificial Intelligence, ECAI 1988, pp. 619 - 624
17. Treur, J., Heuristic reasoning and relative incompleteness. *International Journal of Approximate Reasoning*, vol. 8, 1993, pp. 51-87.
18. R.W. Weyhrauch, Prolegomena to a theory of mechanized formal reasoning, Artificial Intelligence 13 (1980), pp. 133-170

Appendix: Proofs

Lemma 2.6

Let a domain description be given by $\mathbf{W}$. Then the following conditions are equivalent:

(i) For every pair of situations $\mathbf{M, N} \in \mathbf{W}$ which satisfy the same input literals it holds that they also satisfy the same output literals, i.e.:
if $\mathbf{M, N} \in \mathbf{W}$ then

$$\mathbf{In(M) = In(N)} \;\Rightarrow\; \mathbf{Out(M) = Out(N)}$$

(ii) For all $\mathbf{M} \in \mathbf{W_{in}}$ the model $\mathbf{sc_W(M)}$ is complete.

Proof

(i) ⇒ (ii) Let $\mathbf{N}$ be a complete model in $\mathbf{W}$ refining $\mathbf{M} \in \mathbf{W_{in}}$. From (i) it follows that all complete models $\mathbf{N'}$ in $\mathbf{W}$ refining $\mathbf{M}$ are the same. Therefore $\mathbf{sc_W(M) = N} \in \mathbf{W}$.

(ii) ⇒ (i) This follows from Lemma 2.5b)(i). ■

Lemma 3.2

Suppose a signature Σ is given, a non-empty set of complete input models $\mathbf{W_{in}}$ for Σ and a mapping $\alpha\mathbf{:}\ \mathbf{P_{in}} \rightarrow \mathbf{P}$, where $\mathbf{P_{in}} = \mathbf{P(W_{in})}$ and $\mathbf{P}$ is a set of partial models for Σ
.

a) α is conservative if and only if $\mathbf{M} \le \mathbf{In(\alpha(M))}$ for all $\mathbf{M} \in \mathbf{P_{in}}$

b) If α is conservative and self-bounded then α is monotonic.

c) If α is conservative then the following are equivalent:

(i) α is self-bounded

(ii) α is monotonic and for all $\mathbf{M} \in \mathbf{P_{in}}$ it holds

$$\alpha\mathbf{(In(\alpha(M)))} = \alpha\mathbf{(M)}$$

If, moreover, $\Sigma_{\mathbf{in}} = \Sigma_{\mathbf{out}}$ then this is equivalent to

(iii) α is monotonic and $\alpha\mathbf{(\alpha(M))} = \alpha\mathbf{(M)}$ for all $\mathbf{M} \in \mathbf{P_{in}}$ (idempotency).

d) If α does not affect the inputs and α is conservative then the following are equivalent:

(i) α is monotonic

(ii) α is self-bounded.

e) If $\alpha, \beta\mathbf{:}\ \mathbf{P_{in}} \rightarrow \mathbf{P}$ are mappings such that for all $\mathbf{N} \in \mathbf{W_{in}}$ it holds $\alpha\mathbf{(N)} = \beta\mathbf{(N)}$, α is monotonic and β is well-informed, then β is better informed than α.

f) If $\alpha, \beta\ \mathbf{:}\ \mathbf{P_{in}} \rightarrow \mathbf{P}$ are mappings such that for all $\mathbf{N} \in \mathbf{W_{in}}$ it holds $\alpha\mathbf{(N)} = \beta\mathbf{(N)}$ and both mappings are monotonic and well-informed then for all $\mathbf{M} \in \mathbf{P_{in}}$ it holds

$$\mathbf{Out(\alpha(M)) = Out(\beta(M))}\ .$$

Proof

a) This follows from Lemma 2.3b).

b) If $\mathbf{M} \leq \mathbf{N}$ then by conservativeness $\mathbf{M} \leq \mathbf{N} \leq \alpha(\mathbf{N})$. By self-boundedness this implies that $\alpha(\mathbf{M}) \leq \alpha(\mathbf{N})$, i.e. α is monotonic.

c) (i) $\Rightarrow$ (ii) Since $\mathbf{In}(\alpha(\mathbf{M})) \leq \alpha(\mathbf{M})$, by self-boundedness it is trivial that

$$\alpha(\mathbf{In}(\alpha(\mathbf{M}))) \leq \alpha(\mathbf{M}).$$

On the other hand from $\mathbf{M} \leq \alpha(\mathbf{M})$ (conservative) it follows that $\mathbf{M} \leq \mathbf{In}(\alpha(\mathbf{M}))$, and again by conservativity

$$\mathbf{M} \leq \mathbf{In}(\alpha(\mathbf{M})) \leq \alpha(\mathbf{I}(\alpha(\mathbf{M}))).$$

Applying self-boundedness yields $\alpha(\mathbf{M}) \leq \alpha(\mathbf{I}(\alpha(\mathbf{M})))$ and finishes the proof.

(ii) $\Rightarrow$ (i) Suppose $\mathbf{M} \leq \alpha(\mathbf{N})$, then by Lemma 2.3b) we have

$$\mathbf{M} = \mathbf{M}|\Sigma_{\mathbf{in}} \leq \alpha(\mathbf{N})|\Sigma_{\mathbf{in}} = \mathbf{In}(\alpha(\mathbf{N}));$$

hence $\mathbf{M} \leq \mathbf{In}(\alpha(\mathbf{N}))$.

From monotonicity it follows $\alpha(\mathbf{M}) \leq \alpha(\mathbf{In}(\alpha(\mathbf{N}))) = \alpha(\mathbf{N})$. This proves (i).

The final statement of c) is trivial.

d) This follows from b) and c).

e) Let $\mathbf{M} \in \mathbf{P_{in}}$ be given. From monotonicity of α it follows that for any $\mathbf{N} \in \mathbf{W_{in}}$ with $\mathbf{M} \leq \mathbf{N}$ it holds: $\alpha(\mathbf{M}) \leq \alpha(\mathbf{N}) = \beta(\mathbf{N})$. Therefore by Lemma 2.4b) we have

$$\alpha(\mathbf{M}) \leq \mathbf{gci}(\{\beta(\mathbf{N}) \mid \mathbf{N} \in \mathbf{W_{in}},\ \mathbf{M} \leq \mathbf{N}\})$$

Finally, from well-informedness of β it follows that

$$\mathbf{Out}(\mathbf{gci}(\{\beta(\mathbf{N}) \mid \mathbf{N} \in \mathbf{W_{in}},\ \mathbf{M} \leq \mathbf{N}\}) = \mathbf{Out}(\beta(\mathbf{M}))$$

f) This follows from application of e) in two directions. ■

Theorem 3.4

Let a non-empty set of complete input models $\mathbf{W_{in}}$ for signature Σ be given and a set $\mathbf{V}$ of partial models of signature Σ such that $\mathbf{W_{in}} \subset \mathbf{P(V)}$. Define the mapping $\alpha\mathbf{:}\ \mathbf{W_{in}} \to \mathbf{P}$, where $\mathbf{P} = \mathbf{P(V)}$, by $\alpha(\mathbf{M}) = \mathbf{sc_V(M)}$ for all $\mathbf{M} \in \mathbf{P_{in}}$.

Then α is a declarative functionality description.
In particular this holds if **V** is a domain description **W**, and $\mathbf{W_{in}} = \mathbf{In(W)}$.

Proof

Apply Lemma 2.5 to conclude that α is conservative and self-bounded. ■

Proposition 3.5

Suppose a domain description is given by signature Σ and non-empty set of situation models **W**. Let α**:** $\mathbf{P_{in}} \rightarrow \mathbf{P}$ be a functionality description, where $\mathbf{P_{in} = P(In(W))}$ and $\mathbf{P = P(W)}$. The following conditions are equivalent:

(i) For all $\mathbf{M} \in \mathbf{P_{in}}$ each $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ that is true in $\alpha\mathbf{(M)}$ is also true in all complete refinements of **M** in **W**, i.e.
if $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ and $\mathbf{M} \in \mathbf{P_{in}}$ then

$$\alpha\mathbf{(M)} \models \mathbf{h} \Rightarrow \mathbf{M} \Vdash_{\mathbf{W}} \mathbf{h}$$

(ii) For all $\mathbf{M} \in \mathbf{W_{in}}$ each $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ that is true in $\alpha\mathbf{(M)}$ is also true in all complete refinements of **M** in **W**, i.e.
if $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ and $\mathbf{M} \in \mathbf{W_{in}}$ then

$$\alpha\mathbf{(M)} \models \mathbf{h} \Rightarrow \mathbf{M} \Vdash_{\mathbf{W}} \mathbf{h}$$

(iii) For all $\mathbf{M} \in \mathbf{W}$ each $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ that is true in $\alpha\mathbf{(In(M))}$ is also true in **M**, i.e.
if $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ and $\mathbf{M} \in \mathbf{W}$ then

$$\alpha\mathbf{(In(M))} \models \mathbf{h} \Rightarrow \mathbf{M} \models \mathbf{h}$$

Proof

(i) $\Rightarrow$ (ii) This follows from $\mathbf{W_{in}} \subset \mathbf{P_{in}}$.

(ii) $\Rightarrow$ (iii) If $\mathbf{M} \in \mathbf{W}$ then application of condition (ii) to **In(M)** provides $\mathbf{In(M)} \Vdash_{\mathbf{W}} \mathbf{h}$. Hence $\mathbf{M} \models \mathbf{h}$.

(iii) $\Rightarrow$ (i) Suppose $\mathbf{M} \in \mathbf{P_{in}}$ and $\alpha\mathbf{(M)} \models \mathbf{h}$. Let any $\mathbf{N} \in \mathbf{W}$ be given with $\mathbf{M} \leq \mathbf{N}$. By Lemma 2.3b) we have $\mathbf{M} \leq \mathbf{In(N)} \in \mathbf{W_{in}}$. From monotonicity of α it follows that $\alpha\mathbf{(M)} \leq \alpha\mathbf{(In(N))}$. Since $\alpha\mathbf{(M)} \models \mathbf{h}$ this implies $\alpha\mathbf{(In(N))} \models \mathbf{h}$. From condition (iii) it follows that $\mathbf{N} \models \mathbf{h}$. Summarizing: for any $\mathbf{N} \in \mathbf{W}$ with $\mathbf{M} \leq \mathbf{N}$ we have proved that $\mathbf{N} \models \mathbf{h}$, i.e. $\mathbf{M} \Vdash_{\mathbf{W}} \mathbf{h}$. This proves (i). ■

Theorem 3.7

Suppose a domain description is given by signature Σ and non-empty set of situation models **W**. Let α**:** $\mathbf{P_{in}} \rightarrow \mathbf{P}$ be a functionality description, where $\mathbf{P_{in} = P(W_{in})}$ with $\mathbf{W_{in} = In(W)}$ and $\mathbf{P = P(W)}$. Then the following hold:

a) The following conditions are equivalent:

(i) The functionality description α is sound with respect to **W**

(ii) $\mathbf{Out(\alpha(M)) \leq Out(sc_W(M))}$ for all $\mathbf{M} \in \mathbf{P_{in}}$.

(iii) $\alpha\mathbf{(M) \leq sc_W(M)}$ for all $\mathbf{M} \in \mathbf{P_{in}}$.

b) The following conditions are equivalent:

(i) The functionality description α is w-complete with respect to **W**

(ii) $\mathbf{Out(\alpha(M)) \geq Out(sc_W(M))}$ for all $\mathbf{M} \in \mathbf{W_{in}}$.

(iii) $\alpha\mathbf{(M) \geq sc_W(M)}$ for all $\mathbf{M} \in \mathbf{W_{in}}$.

c) The functionality description α is complete with respect to **W** if and only if

$$\mathbf{Out(\alpha(M)) \geq Out(sc_W(M))} \text{ for all } \mathbf{M} \in \mathbf{P_{in}}.$$

d) The following conditions are equivalent:

(i) The functionality description α w-covers **W**

(ii) $\mathbf{Out(\alpha(M)) = Out(sc_W(M))}$ for all $\mathbf{M} \in \mathbf{W_{in}}$.

(iii) $\alpha\mathbf{(M) = sc_W(M)}$ for all $\mathbf{M} \in \mathbf{W_{in}}$.

e) The functionality description α covers **W** if and only if

$$\mathbf{Out(\alpha(M)) = Out(sc_W(M))} \text{ for all } \mathbf{M} \in \mathbf{P_{in}}.$$

There exists a functionality description that covers **W**, namely $\mathbf{sc_W}$.

f) If α w-covers **W** , then the domain description given by **W** is empirically founded if and only if for every $\mathbf{M} \in \mathbf{In(W)}$ the model $\alpha\mathbf{(M)}$ is complete.

Proof

a) Let $\mathbf{M} \in \mathbf{P_{in}}$ be given. From Lemma 2.5a) it follows that for any $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ it holds $\mathbf{sc_W(M)} \vDash \mathbf{h}$ if and only if $\mathbf{M} \Vdash_{\mathbf{W}} \mathbf{h}$. Therefore soundness is equivalent to:
for each $\mathbf{M} \in \mathbf{P_{in}}$ and all $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ it holds

$$\alpha\mathbf{(M)} \vDash \mathbf{h} \;\Rightarrow\; \mathbf{sc_W(M)} \vDash \mathbf{h}$$

This proves (i) $\Leftrightarrow$ (ii) of a).
It is clear that this condition follows from $\alpha\mathbf{(M) \leq sc_W(M)}$. This proves (iii) $\Rightarrow$ (i) of a).
We will now prove (ii) $\Rightarrow$ (iii). Assume (ii) holds. We will prove that $\alpha\mathbf{(M) \leq sc_W(M)}$. By (ii), for any given $\mathbf{M} \in \mathbf{P_{in}}$ we have: for all $\mathbf{h} \in \mathbf{OutLit(\Sigma)}$ it holds

$$\alpha\mathbf{(M)} \vDash \mathbf{h} \;\Rightarrow\; \mathbf{sc_W(M)} \vDash \mathbf{h}$$

Therefore $\alpha\mathbf{(M)|\Sigma_{out} \leq sc_W(M)|\Sigma_{out}}$.

Since we assume that only input and output signatures are involved in functionality descriptions (and no internal signatures), the only thing that remains to be proved is that it also holds $\alpha\mathbf{(M)|\Sigma_{in} \leq sc_W(M)|\Sigma_{in}}$. Notice that

$$\mathbf{sc_W(M)|\Sigma_{in} = gci(\{M'|\ M \leq M' \in W\}|\Sigma_{in}}$$
$$\mathbf{= gci(\{In(M')|\ M \leq M' \in W\}}$$

If $\mathbf{M \leq M' \in W}$ then $\mathbf{M = In(M) \leq In(M')}$, and by monotonicity it follows that $\alpha\mathbf{(M)} \leq \alpha\mathbf{(In(M'))}$ and therefore by Lemma 2.3b) we have

$$\alpha\mathbf{(M)|\Sigma_{in}} \leq \alpha\mathbf{(In(M'))|\Sigma_{in}}.$$

By conservatism we have $\mathbf{In(M')} \leq \alpha\mathbf{(In(M'))}$. Therefore we derive

$$\mathbf{In(M') = In(M')|\Sigma_{in}} \leq \alpha\mathbf{(In(M'))|\Sigma_{in}} .$$

Now $\mathbf{In(M')} \in \mathbf{W_{in}}$, so it has no real refinements (except itself).

Therefore $\alpha\mathbf{(In(M'))|}\Sigma_{\mathbf{in}} = \mathbf{In(M')}$ and thus

$$\alpha\mathbf{(M)|}\Sigma_{\mathbf{in}} \leq \alpha\mathbf{(In(M'))|}\Sigma_{\mathbf{in}} = \mathbf{In(M')}$$

By Lemma 2.4 this proves $\alpha\mathbf{(M)|}\Sigma_{\mathbf{in}} \leq \mathbf{gci(\{In(M')|\ M \leq M' \in W\}) = sc_W(M)|}\Sigma_{\mathbf{in}}$.

This proves (ii) ⇒ (iii) of a).

b) The proof of b)(i) ⇔ (ii) is similar to the proof of a)(i) ⇔ (ii).
For the proof of (ii) ⇔ (iii), notice that for **M** a complete input model, since α and $\mathbf{sc_W}$ are conservative, the input parts of $\alpha\mathbf{(M)}$ and $\mathbf{sc_W(M)}$ both are equal to **M**.

c) This is similar to the proof of a)(i) ⇔ (ii).

d) This follows from a) and b).

e) The first part of this follows from a) and c). From Theorem 3.4 it follows that the mapping defined by $\mathbf{sc_W : M \to sc_W(M)}$ for all $\mathbf{M \in P_{in}}$ is a declarative functionality description; by the first part of e) it covers **W**.
f) This follows from d) and Lemma 2.6. ■

Proposition 3.8
Suppose a signature Σ is given with a non-empty set of complete input models $\mathbf{W_{in}}$ and **P** is a set of partial models for Σ. Assume the mapping $\alpha\mathbf{: P_{in} \to P}$, where $\mathbf{P_{in} = P(W_{in})}$, is a declarative functionality description for Σ. Moreover, let a domain description **W** for signature Σ with $\mathbf{In(W) = W_{in}}$ be given.

Then the following conditions are equivalent:
(i) **W** is covered by α
(ii) **W** is w-covered by α and α is well-informed.

Proof
(i) ⇒ (ii) Let $\mathbf{M \in P_{in}}$ be given. We have to prove that α is well-informed, i.e.

$$\mathbf{Out(}\alpha\mathbf{(M)) = Out(gci(V_1))}$$

where

$$\mathbf{V_1 = \{\ }\alpha\mathbf{(N)\ |\ N \in W_{in}\ \&\ M \leq N\ \}}$$

For any $\mathbf{N \in W_{in}}$ with $\mathbf{M \leq N}$, from monotonicity it follows that $\alpha\mathbf{(M)} \leq \alpha\mathbf{(N)}$.

Therefore by Lemma 2.4b)

$$\alpha\mathbf{(M)} \leq \mathbf{gci(V_1)} \qquad \mathbf{(1)}$$

This implies one side of what we have to prove. The other side is the harder part. Since **W** is covered by α by Theorem 3.7 we have

$$\mathbf{Out(}\alpha\mathbf{(M)) = Out(sc_W(M)) = Out(gci(V_2))}$$

with

$$\mathbf{V_2 = \{ N \in W \mid M \leq N \}}$$

We will apply Lemma 2.4a) to the sets $\mathbf{V_1}$ and $\mathbf{V_2}$. Suppose an arbitrary $\mathbf{N \in V_2}$ is given, then $\mathbf{N \in W}$ and $\mathbf{M \leq N}$. Take $\mathbf{M' = In(N) \in W_{in}}$; this is a complete model with respect to the input signature. By Lemma 2.3b) $\mathbf{M \leq M'}$, so $\alpha\mathbf{(M') \in V_1}$. Since α covers $\mathbf{W}$ it also w-covers $\mathbf{W}$; further $\mathbf{N}$ is complete. Therefore by Theorem 3.7d) we have

$$\alpha\mathbf{(M') = sc_W(M') \leq sc_W(N) = N}$$

This shows us that the conditions of Lemma 2.4a) are satisfied. Applying this lemma provide
s

$$\mathbf{gci(V_1) \leq gci(V_2) = sc_W(M)}$$

Therefore

$$\mathbf{Out(gci(V_1)) \leq Out(sc_W(M)) = Out(\alpha(M)).} \qquad \mathbf{(2)}$$

This is the other side of what we had to prove. From **(1)** and **(2)** it follows that
α is well-informed.
(ii) $\Rightarrow$ (i) Suppose $\mathbf{W}$ is w-covered by a well-informed α. Then by Theorem 3.7d) we have $\mathbf{sc_W(M) =}$ $\alpha\mathbf{(M)}$ for all $\mathbf{M \in W_{in}}$. By applying Lemma 3.2f) we derive that $\mathbf{Out(sc_W(M)) = Out(\alpha(M))}$ for all $\mathbf{M \in P_{in}}$. From Theorem 3.7e) it follows that (i) holds. ■

Theorem 3.9

Suppose a signature Σ is given, $\mathbf{W_{in}}$ is a non-empty set of complete input models and $\mathbf{P}$ is a set of partial models for Σ. Assume $\alpha\mathbf{: P_{in} \rightarrow P}$ is a declarative functionality description for Σ, where $\mathbf{P_{in} = P(W_{in})}$.
Then a domain description $\mathbf{W^*}$ for Σ with $\mathbf{In(W^*) = W_{in}}$ can be obtained that is w-covered by α. One can construct $\mathbf{W^*}$ from α by taking

$$\mathbf{W^* = \{ N \mid N \text{ is a complete model of signature } \Sigma \;\&\; \exists M \in W_{in} \;\; \alpha(M) \leq N\}.}$$

Moreover, $\mathbf{W^*}$ is covered by α if and only if α is well-informed.
For every domain description $\mathbf{W}$ with $\mathbf{In(W) = W_{in}}$ for which α is sound, $\mathbf{W}$ is contained in $\mathbf{W^*}$.

Proof

Take the domain description given by

$$\mathbf{W^* = \{ N \mid N \text{ is a complete model of signature } \Sigma \;\&\; \exists M \in W_{in} \;\; \alpha(M) \leq N\}}$$

It is easy to verify that $\mathbf{In(W^*) = W_{in}}$.
We will prove that $\mathbf{W^*}$ is w-covered by α ; first we treat soundness. Let an $\mathbf{M \in P_{in}}$ be given. By the choice of $\mathbf{W^*}$ above and by Lemma 2.3b) we can rewrite

$$\mathbf{sc_{W^*}(M) = gci\{ N \in W^* \mid M \leq N \} = gci(V_2)}$$

with

$$\mathbf{V_2 = \{ N \mid N \text{ is a complete model } \& \ M \le N \ \& \ \exists M' \in W_{in} \ \alpha(M') \le N \}}$$

We will prove that $\alpha(\mathbf{M})$ is less than or equal to every member of $\mathbf{V_2}$. Let a member $\mathbf{N}$ of $\mathbf{V_2}$ be given, with $\mathbf{M' \in W_{in}}$ such that $\alpha(\mathbf{M'}) \le \mathbf{N}$. Then both $\mathbf{M \le N}$ and $\mathbf{M' \le \alpha(M') \le N}$, while $\mathbf{M' = In(N)}$. From Lemma 2.3b) it follows that $\mathbf{M \le M'}$. Therefore, by monotonicity $\alpha(\mathbf{M}) \le \alpha(\mathbf{M'}) \le \mathbf{N}$. By applying Lemma 2.4b) it follows that

$$\alpha(\mathbf{M}) \ \le \ \mathbf{gci(V_2)} \ = \ \mathbf{sc_{W*}(M)} \qquad \mathbf{(3)}$$

This proves soundness of α with respect to $\mathbf{W^*}$.
Next we will treat w-completeness. We are done if we prove that $\mathbf{sc_{W*}(M)} \le \alpha(\mathbf{M})$ for all $\mathbf{M \in W_{in}}$. Let such an $\mathbf{M}$ be given. Then we have $\mathbf{M} \le \alpha(\mathbf{M})$. Notice that by our choice of $\mathbf{W^*}$ every complete refinement of $\alpha(\mathbf{M})$ is in $\mathbf{W^*}$; therefore by Lemma 2.5b) and 2.5a):

$$\mathbf{sc_{W*}(M)} \le \mathbf{sc_{W*}}(\alpha(\mathbf{M})) = \alpha(\mathbf{M}) \qquad \mathbf{(4)}$$

This proves the w-completeness of α with respect to $\mathbf{W^*}$. From **(3)** and **(4)** it follows that α w-covers $\mathbf{W^*}$.
By Proposition 3.8 α is well-informed if and only if α covers $\mathbf{W^*}$.
Finally we will prove that for any $\mathbf{W}$ such that α is sound with respect to $\mathbf{W}$ it holds $\mathbf{W \subset W^*}$. Suppose $\mathbf{M \in W}$ is given, then by Theorem 3.7a) from soundness of α with respect to $\mathbf{W}$ it follows that

$$\alpha(\mathbf{In(M)}) \le \mathbf{sc_W(In(M))} \le \mathbf{M}$$

Therefore $\mathbf{M \in W^*}$. ■

Corollary 3.10
Suppose a signature Σ is given, $\mathbf{W_{in}}$ is a non-empty set of input models for Σ and $\mathbf{P}$ is a set of partial models for Σ. Assume $\alpha\mathbf{: P_{in}} \to \mathbf{P}$, where $\mathbf{P_{in} = P(W_{in})}$, is a declarative functionality description for Σ.
Then there is a well-informed declarative functionality description $\beta\mathbf{: P_{in}} \to \mathbf{P}$ that is a well-informed refinement of α . This β is better informed than α ; in particular, it holds

$$\mathbf{Out}(\alpha(\mathbf{M})) \le \mathbf{Out}(\beta(\mathbf{M})) = \mathbf{Out(sc_{W*}(M))}$$

for all $\mathbf{M \in P_{in}}$, with $\mathbf{W^*}$ as in Theorem 3.9.

Proof

First we prove the existence of such a β. Apply Theorem 3.9 to the functionality description α. By Theorem 3.7 the resulting domain $\mathbf{W^*}$ is covered by the well-informed $\beta = \mathbf{sc_{W*}}$.
From Theorem 3.9 it follows that $\mathbf{W^*}$ is w-covered by α . From Theorem 3.7b), applied to α it follows that

$$\mathbf{Out}(\alpha(\mathbf{N})) = \ \mathbf{Out(sc_{W*}(N))} = \ \mathbf{Out}(\beta(\mathbf{M}))$$

for all $\mathbf{N} \in \mathbf{W_{in}}$. From Lemma 3.2e) it follows that β is better informed than α. This proves the existence. ■

Lemma 4.2
Suppose a signature Σ is given and $\mathbf{W_{in}}$ is a non-empty set of input models. Take for $\mathbf{P}$ the set of all partial models of signature Σ and $\mathbf{P_{in} = P(W_{in})}$. Let a consistent reasoning component specification for Σ and $\mathbf{P_{in}}$ be given by $\mathbf{KB}$.
Then the mapping $\mathbf{cons_{KB} : P_{in} \to P}$ given by $\mathbf{M} \to \mathbf{cons_{KB}(M)}$ is a well-informed (declarative) functionality description. Moreover, $\mathbf{cons_{KB} = sc_{Mod(KB)}}$.

Proof
Since $\mathbf{cons_{KB} = sc_{Mod(KB)}}$ this follows from Theorems 3.4, 3.7 and Proposition 3.8. ■

Lemma 4.6
Assume a signature Σ is given, $\mathbf{W_{in}}$ is a non-empty set of complete input models and $\mathbf{P}$ a set of models for Σ. Assume $\alpha : \mathbf{P_{in} \to P}$ where $\mathbf{P_{in} = P(W_{in})}$ is a functionality description. The non-empty set of rules $\mathbf{KB}_\alpha$ is constructed as follows. For each output literal $\mathbf{h}$, take

$$\mathbf{T(h) \;=\; \{\, M \in P_{in} \;|\; \alpha(M) \models h \,\}}$$

and $\mathbf{mT(h)}$ the set of minimal (with respect to the refinement relation) elements in $\mathbf{T(h)}$.
Define $\mathbf{KB}_\alpha$ by the following set of rules

$$\mathbf{KB}_\alpha \;=\; \{\, \mathbf{h} \;|\; \Lambda \in \mathbf{mT(h)} \,\} \;\cup\; \{\, \mathbf{Con(Lit(M))} \;\to\; \mathbf{h} \;|\; \mathbf{M} \in \mathbf{mT(h)}, \mathbf{M} \neq \Lambda \,\}$$

where **Con(..)** means taking the conjunction of a set of literals.
Then a consistent reasoning component specification is obtained with signature Σ, knowledge base $\mathbf{KB}_\alpha$, and set of partial input models $\mathbf{P_{in}}$ such that for all complete models $\mathbf{M} \in \mathbf{W_{in}}$ it holds $\alpha\mathbf{(M)} \models \mathbf{KB}_\alpha$.

Proof
Suppose a model $\mathbf{M} \in \mathbf{W_{in}}$ is given. We will prove that $\alpha\mathbf{(M)} \models \mathbf{KB}_\alpha$. This also implies consistency. First we treat the general facts in $\mathbf{KB}_\alpha$. Assume $\mathbf{h} \in \mathbf{KB}_\alpha$ so $\Lambda \in \mathbf{mT(h)}$. Therefore $\alpha(\Lambda) \models \mathbf{h}$. From monotonicity it follows that $\alpha\mathbf{(M)} \models \mathbf{h}$. Next we treat a rule that is no general fact, say

$$\mathbf{Con(Lit(M_0)) \;\to\; h}$$

with $\mathbf{M_0} \in \mathbf{mT(h)}$. Since $\mathbf{M}$ is complete, from not $\mathbf{M} \models \mathbf{Con(Lit(M_0))}$ it follows that

$$\mathbf{M} \models \neg\, \mathbf{Con(Lit(M_0))}$$

By the strong Kleene rule for implication in that case the rule is true in **M**, independent of the truth value of **h**. By conservatism the same holds for $\alpha(\mathbf{M})$. In the other case

$$\mathbf{M} \vDash \mathbf{Con(Lit(M_0))}$$

This implies that $\mathbf{M_0} \leq \mathbf{M}$. Therefore, by monotonicity we have $\alpha(\mathbf{M_0}) \leq \alpha(\mathbf{M})$. Now from $\mathbf{M_0} \in \mathbf{mT(h)}$ it follows that $\alpha(\mathbf{M_0}) \vDash \mathbf{h}$, hence $\alpha(\mathbf{M}) \vDash \mathbf{h}$. So also in this case the rule is true in $\alpha(\mathbf{M})$. ■

Theorem 4.7
Assume a signature Σ is given, $\mathbf{P_{in}}$ is a non-empty set of partial input models for Σ and let $\alpha : \mathbf{P_{in}} \rightarrow \mathbf{P(W)}$ be a well-informed functionality description.

Then the well-informed functionality description related to the consistent reasoning component specification with signature Σ, knowledge base $\mathbf{KB}_\alpha$, and a set of partial input models $\mathbf{P_{in}}$ covers α. This reasoning component specification is minimal.

Proof

We will prove that the well-informed functionality description of the reasoning component specification with $\mathbf{KB} = \mathbf{KB}_\alpha$ given in Lemma 4.6 covers α. First we prove $\mathbf{Out(cons_{KB}(M))} \leq \mathbf{Out}(\alpha(\mathbf{M}))$ for all $\mathbf{M} \in \mathbf{P_{in}}$. Let the output literal $\mathbf{h} \in \mathbf{OutLit}(\Sigma)$ and $\mathbf{M} \in \mathbf{P_{in}}$ be given with $\mathbf{M} \vdash_{\mathbf{KB}} \mathbf{h}$. By Lemma 4.6 for every $\mathbf{N} \in \mathbf{W_{in}}$ with $\mathbf{M} \leq \mathbf{N}$ the model $\alpha(\mathbf{N})$ is a model of **KB**, hence $\alpha(\mathbf{N}) \vDash \mathbf{h}$. By well-informedness $\alpha(\mathbf{M}) \vDash \mathbf{h}$.
Next we prove $\mathbf{Out(cons_{KB}(M))} \geq \mathbf{Out}(\alpha(\mathbf{M}))$ for all $\mathbf{M} \in \mathbf{P_{in}}$. Suppose we are given $\mathbf{h} \in \mathbf{OutLit}(\Sigma)$ and an $\mathbf{M} \in \mathbf{P_{in}}$ with $\alpha(\mathbf{M}) \vDash \mathbf{h}$. We will show that $\mathbf{M} \vDash_{\mathbf{KB}} \mathbf{h}$. Since $\alpha(\mathbf{M}) \vDash \mathbf{h}$ we have $\mathbf{M} \in \mathbf{T(h)}$. Take a minimal element $\mathbf{M'}$ in $\mathbf{T(h)}$ with $\mathbf{M'} \leq \mathbf{M}$. If $\mathbf{M'} = \Lambda$, then from $\mathbf{M'} \in \mathbf{mT(h)}$ it follows $\mathbf{h} \in \mathbf{KB}$, so $\mathbf{M} \vDash_{\mathbf{KB}} \mathbf{h}$ and we are done. In the other case that $\mathbf{M'} \neq \Lambda$ we have the following rule in **KB**:

$$\mathbf{Con(Lit(M'))} \rightarrow \mathbf{h}$$

From $\mathbf{M'} \leq \mathbf{M}$ it follows that $\mathbf{M} \vDash \mathbf{Con(Lit(M'))}$. Therefore for any partial model **N** with $\mathbf{N} \geq \mathbf{M}$ and $\mathbf{N} \vDash \mathbf{KB}$ it holds $\mathbf{N} \vDash \mathbf{Con(Lit(M'))}$. Since $\mathbf{N} \vDash \mathbf{KB}$ it holds $\mathbf{N} \vDash \mathbf{Con(Lit(M'))} \rightarrow \mathbf{h}$. Therefore, (by the strong Kleene truth value combination table) we have $\mathbf{N} \vDash \mathbf{h}$. This proves $\mathbf{M} \vDash_{\mathbf{KB}} \mathbf{h}$.

Therefore the well-informed functionality related to the reasoning component specification as constructed covers the given functionality description α. Finally we show it is minimal. Suppose we obtain **KB'** from **KB** by leaving out one of the conditions in the condition part of a rule

$$\mathbf{Con(Lit(M))} \rightarrow \mathbf{h}$$

with $\mathbf{M} \in \mathbf{mT(h)} \subset \mathbf{T(d)}$ and $\mathbf{M} \neq \Lambda$. The resulting condition part corresponds to a partial model $\mathbf{M'} \leq \mathbf{M}$ with $\mathbf{M'} \neq \mathbf{M}$. Since $\mathbf{M}$ was minimal in $\mathbf{T(h)}$, it holds $\mathbf{M'} \not\in \mathbf{T(h)}$; therefore $\alpha\mathbf{(M')} \not\models \mathbf{h}$, while $\mathbf{M'} \models_{\mathbf{KB'}} \mathbf{h}$. So this knowledge base **KB'** would not be equivalent to **KB**. This proves that the constructed **KB** is minimal. ■

Proposition 4.8

Let a domain description for signature Σ and a non-empty set of input models $\mathbf{W_{in}}$ be given by **W**. Suppose α is a declarative functionality description w-covering **W** and **KB** defines a reasoning component specification such that its related well-informed functionality description covers α.

The following conditions are equivalent:

(i) The domain described by **W** is empirically founded

(ii) For every $\mathbf{M} \in \mathbf{W_{in}}$ the model $\alpha\mathbf{(M)}$ is complete

(iii) For every $\mathbf{M} \in \mathbf{W_{in}}$ and output literal **h** it holds either $\mathbf{M} \models_{\mathbf{KB}} \mathbf{h}$ or $\mathbf{M} \models_{\mathbf{KB}} \neg \mathbf{h}$

(iv) For any output literal **h** there exists a proposition **p** in terms of input literals such that **h** is true in a situation $\mathbf{M} \in \mathbf{W}$ if and only if **p** is true in **M** (*explicit definability*), i.e.: for every $\mathbf{M} \in \mathbf{W}$ it holds

$$\mathbf{M} \models \mathbf{h} \Leftrightarrow \mathbf{M} \models \mathbf{p}$$

Proof

(i) $\Leftrightarrow$ (ii) This follows from Theorem 3.7b).

(ii) $\Rightarrow$ (iii) Let a complete input model $\mathbf{M} \in \mathbf{W_{in}}$ and an output literal **h** be given. Suppose $\mathbf{M} \models_{\mathbf{KB}} \mathbf{h}$ is not the case. This implies that not $\mathbf{cons_{KB}(M)} \models \mathbf{h}$. Since **KB** defines a reasoning component specification such that its related well-informed functionality description is covering α, therefore not $\alpha\mathbf{(M)} \models \mathbf{h}$. From completeness of $\alpha\mathbf{(M)}$ it follows that $\mathbf{Out(cons_{KB}(M)) = Out(}\alpha\mathbf{(M))} \models \neg \mathbf{h}$, hence we have $\mathbf{M} \models_{\mathbf{KB}} \neg \mathbf{h}$.

(iii) $\Rightarrow$ (ii) Following the lines of the above proof in the reversed order this can easily be established.

(i) $\Rightarrow$ (iv) This proof is a variant of the construction in Theorem 4.6. Let an output literal **h** be given. Take an indexing of all $\mathbf{M_i} \in \mathbf{W}$ with $\mathbf{M_i} \models \mathbf{h}$; we show that **h** is explicitly definable by the proposition

$$\mathbf{p = Dis\{Con(Lit(In(M_i))) \mid i = 1, .. , k\}}$$

where **Dis** stands for the disjunction of a set of formulas. If $\mathbf{M} \in \mathbf{W}$ is given with $\mathbf{M} \models \mathbf{p}$, then $\mathbf{M} \models \mathbf{Con(Lit(In(M_i)))}$ for some **i** , so $\mathbf{In(M) = In(M_i)}$. From (i) and

$\mathbf{M_i} \vDash \mathbf{h}$ it follows that also $\mathbf{M} \vDash \mathbf{h}$. Conversely, suppose $\mathbf{M} \vDash \mathbf{h}$, then $\mathbf{M = M_i}$ for some $\mathbf{i}$. Then $\mathbf{M} \vDash \mathbf{Con(Lit(In(M_i)))},$ so $\mathbf{M} \vDash \mathbf{p}$. We have proved that for every

$\mathbf{M} \in \mathbf{W}$ it holds that $\mathbf{M} \vDash \mathbf{p} \Leftrightarrow \mathbf{M} \vDash \mathbf{h}.$
(iv) $\Rightarrow$ (i) This is easy to verify. ∎