

Defaults in specifications: distance functions between temporal models

February 1998

1 Introduction

Default reasoning concerns the logic of assertions which should be assumed to be true but for which there might be exceptions. Although it first appeared as a field within Artificial Intelligence, the need to be able to express default assertions in specifications of software systems is well established. For example, using defaults we may specify *fault-tolerant systems*, exhibiting a normal behaviour to which exceptions arise when a fault develops. In this case, an exception should be dealt with within the specification, given rise to appropriate corrective actions.

There are several different approaches to default reasoning; model preference logics is one branch of approaches. It is based on the semantical notions of choosing a preferred set of models of a theory. The main idea is to consider, instead of arbitrary models of a given axiom set, only models that satisfy a certain minimality condition. This minimality condition is related to the way models satisfy the defaults. One way of doing this is by defining an ordering between models, where interpretations of a logical language are ordered according to how well they satisfy some given default information; this framework is called *preference relations* [2, 3]. Another natural way of comparing interpretations is by the use of distance functions: if \mathcal{M} is a set of interpretations and $m, n \in \mathcal{M}$ then $d(m, n)$ is some notion of distance between m and n . defaults. In [4], this is generalized by considering morphisms between interpretations and an ordering between morphisms. These morphisms correspond to several ways of comparing two interpretations; distances are a particular case of a single morphism between any two interpretations. The semantics are based on this notion of ordering, selecting the models of the axioms that are as close as possible to the models of the defaults.

In order to utilize defaults in specifications of dynamic systems, we use a temporal instantiation of the default institution proposed in [4]. In this paper we briefly describe the notion of default institution and then the temporal instantiation. An example of the use of this framework in specifications is presented.

2 The temporal default institution

Default institutions were proposed in [4] as a way of extending the notion of institution [1]. Institutions provide a way of talking about an arbitrary logical system and provide a way of structuring specifications. However, with this structuring an existing specification can be enriched but not modified. Default institutions allow partial reuse of existing specifications, where the ‘default’ module can be modified by introducing exceptions to suit the application at hand.

The concept of default institution is an extension of institutions by adding a notion of distance between interpretations. The semantics of a specification with an exception are given by the models of the exception that are as close as possible to the models of the specification. This is non-monotonic since by adding exceptions it might be needed to retract some of the previous conclusions.

In general, we want to compare interpretations that may be different in nature. Therefore, we need a way to relate elements of different nature that play a similar role. This is done by morphisms between models, the morphisms of the category $Int(\Sigma)$, where Σ is a signature. In order to compare morphisms and give a precise meaning to ‘closest’, a pre-order \leq among morphisms is needed. These define the comparison category $Comp(\Sigma)$, for each signature Σ , whose objects are the morphisms in

the category $Int(\Sigma)$ of interpretations and whose morphisms are defined by the pre-order \leq . The formal definition of a default institution is the following:

2.1 Definition. [4] A default institution consists of

- a category $Sign$ of signatures;
- a functor $Sen : Sign \rightarrow Set$, giving languages linked by translations;
- a contravariant functor $Int : Sign \rightarrow Cat^{op}$, giving interpretations and their morphisms; the class of interpretations should not be empty;
- a family of satisfaction relations \Vdash_{Σ} between the interpretations of Σ and its formulas ($\Vdash_{\Sigma} \subseteq |Int(\Sigma)| \times Sen(\Sigma)$);
- a functor $Comp : Sign \rightarrow Cat^{op}$, such that
 - $Comp(\Sigma)$ is a pre-order;
 - the objects of $Comp(\Sigma)$ are the morphisms of $Int(\Sigma)$;
 - the identities of $Int(\Sigma)$ are initial (minima) in $Comp(\Sigma)$;
 - the morphisms of Int that are minima in $Comp$ are called *agreements*; they must form a subcategory Int_0 , that is, the composition of agreements must be an agreement;
 - the institution where Int is replaced by Int_0 is weakly abstract;
 - 0-symmetry: each agreement $h : M \rightarrow N$ has a reverse agreement $h^R : N \rightarrow M$ such that $(h^R)^R = h$;
 - 0-equivalence: for any morphism $h : B \rightarrow C$ and agreements $a : A \rightarrow B$, $c : C \rightarrow D$, $a; h \equiv h; c$.

The semantics of a default D and an exception E , written $D\mathbf{but}E$, are defined using the pre-order of the comparison category $Comp(\Sigma)$. Let us introduce some notation that is used in the formal definition: $Mor(E, D)$ is the set of morphisms whose domain satisfy E and whose codomain satisfy D ; $Min(E, D)$ is the set of minimal morphisms of $Mor(E, D)$. An interpretation m is a model of $D\mathbf{but}E$ if m is the domain of a minimal morphism between the morphisms whose domain satisfies E and whose codomain satisfies D . Note that a model of $D\mathbf{but}E$ always satisfy the exception E .

2.2 Definition (Semantics of **but**). $m \Vdash D\mathbf{but}E$ iff there is a morphism $h \in Min(E, D)$ such that $dom(h) = m$.

The temporal default institution is an instantiation of this definition with propositional linear temporal logic. However, the set of symbols is composed by two disjoint sets: a set of attributes Att and a set of actions Act . The need to split the symbols in these two sets comes from the fact that we are going to deal with them in different ways when comparing morphisms. A signature Σ is then a pair (Act, Att) . The language (the objects of the category $Sen(\Sigma)$) are formulae constructed in the usual way using the boolean connectives and the temporal operators U (until) and X (next); from these temporal operators it is possible to define G (always) and F (eventually).

The models are triples of the form $\mathcal{M} = (\mathbb{N}, <, V)$, where V is a valuation. It assigns to every time point $t \in \mathbb{N}$ a set of symbols $V(t) \subseteq \Sigma$, namely the set of proposition letters that are true at the instant t . Note that in these models, time is discrete, has an initial moment with no predecessors, and is infinite into the future. The semantics of the formulae are the usual for linear temporal logic, where we write $m \Vdash_t A$ if the model m satisfies the formula A at the time point t . We use the anchored version of temporal logic, where a formula A is said to be *valid* on the model m iff $m \Vdash_0 A$, and we consider the reflexive versions of the modalities. These models are the objects of the category $Int(\Sigma)$. To define morphisms between interpretations, we introduce some notation:

Let $h : \mathbb{N} \not\rightarrow \mathbb{N}$ be a partial function and $t \in \mathbb{N}$ be a natural number:

- $h(t) \uparrow$ means that h is undefined at t ;
- $h(t) \downarrow$ means that h is defined at t ;
- $\bar{h}(t) = \max\{t' \in \mathbb{N} : t' \leq t \text{ and } h(t') \downarrow\}$.

The idea of morphism is to relate two temporal models m and n that have the same initial state (they satisfy exactly the same atomic formulae at 0), in a way that, for each instant t_1 , a timepoint t_2 is chosen in a monotonic way, such that the model domain m at t_1 satisfies exactly the same atomic symbols as the model codomain n at t_2 . This choice is partial to allow cases where such t_2 does not exist. These conditions are formally expressed by the following definition:

2.3 Definition. Let $\Sigma \in \text{Sign}$ be a signature and m and n be two temporal models in $|\text{Int}(\Sigma)|$. A *morphism* $h : m \rightarrow n \in \text{Mor}(\text{Int}(\Sigma))$ is a partial function $h : \mathbb{N} \dashrightarrow \mathbb{N}$ such that:

- $h(0) = 0$;
- for all $t_1, t_2 \in \mathbb{N}$, if $t_1 < t_2$, $h(t_1) \downarrow$ and $h(t_2) \downarrow$ then $h(t_1) < h(t_2)$;
- for all $t \in \mathbb{N}$, if $h(t) \downarrow$ then for all $a \in \Sigma$, $m \Vdash_t a$ iff $n \Vdash_{h(t)} a$.

Suppose $h : m \rightarrow n$ is a morphism between two temporal models. If m was ‘near’ n , it would behave, at every time point, in the same way as n if n was at a similar state. Discrepancies are the cases when this does not happen.

2.4 Definition. Let $\Sigma = (\text{Act}, \text{Att}) \in \text{Sign}$ be a signature, $h : m \rightarrow n$ be a morphism in $\text{Int}(\Sigma)$ and $t \in \mathbb{N}$ a timepoint. We say that there is a *discrepancy* at time t between m and n according to h , written $m \overset{h}{\not\leftrightarrow}_t s$, if $h(t)$ is defined and there is an attribute $at \in \text{Att}$ such that

$$(m \Vdash_{t+1} at \text{ and } n \not\Vdash_{h(t)+1} at) \quad \text{or} \quad (m \not\Vdash_{t+1} at \text{ and } n \Vdash_{h(t)+1} at)$$

This notion of discrepancy expresses the fact that, if the model n was at the state that m is at some instant t , it would behave in a different way. This depends on the morphisms h , that chooses from n an instant to compare with m at the instant t : by definition of morphism, if $h : m \rightarrow n$ is a morphism, then m satisfies at t exactly the same symbols as n at $h(t)$. There is a discrepancy at t if they progress in different ways: at the following instants, $t + 1$ for m and $h(t) + 1$ for n , they satisfy different attributes.

It remains to define the pre-order between morphisms. If $h : m \rightarrow n$ and $h' : m' \rightarrow n'$ are morphisms, we want to know if m is closer to n (according to h) than m' to n' (according to h'). The main idea is to minimize the discrepancies. However, this is not enough. Firstly, we only compare morphisms if the domains (m and m') have the same initial state and they satisfy the same actions at the same instants (conditions 1 and 2 of the definition 2.5). The condition 2 makes possible comparisons of models only if the same actions occur at the same time points, so that we choose, from the models that have the same actions occurring, the ones that are as close as possible to the models of the defaults. It is a way of avoiding blockage of the occurrence of actions: if the effect of an action contradicts the defaults, then a model where that action would not occur would be better than the ones where that action occurs.

The minimization of the discrepancies is expressed by condition 3 of the definition: if, at some instant, there is a discrepancy in a morphism h , and there is not a discrepancy at h' , then $h \leq h'$ only if there was a previous discrepancy at h' that did not exist at h , and this discrepancy was the first one at both morphisms. This is like a lexicographic ordering, where discrepancies that occur in earlier instants have higher priority than the ones that occur later (chronological minimization). It also ensures that if $h \leq h'$ and there is a discrepancy in h that does not exist in h' , the absence of the discrepancy at h' is possible because there was an earlier one that allows the domain of h' to have a behaviour closer to a model of the defaults.

Moreover, it is also desirable that the morphisms are as defined as possible, which is expressed by condition 4: a completely undefined function does not have any discrepancy.

The last condition ensures that the morphisms are as surjective as possible; this has as a result that identities are minima, and that any minimal morphism is surjective, satisfying one of the conditions of the definition of default institution, namely that the category Int_0 is weakly abstract. This is not explained in this abstract, but the idea underlining this condition is that minimal morphisms satisfy exactly the same formulae. The formal definition of the pre-order is the following:

2.5 Definition. Let $\Sigma = (\text{Act}, \text{Att}) \in \text{Sign}$ be a signature and let $h : m \rightarrow n$ and $h' : m' \rightarrow n'$ be morphisms in $\text{Int}(\Sigma)$. We say that $h : m \rightarrow n \leq h' : m' \rightarrow n'$ iff:

1. $\forall a \in \Sigma. m \Vdash_0 a$ iff $m' \Vdash_0 a$;
2. $\forall ac \in \text{Act}. \forall t \in \mathbb{N}. m \Vdash_t ac$ iff $m' \Vdash_t ac$;
3. $\forall t_1 \in \mathbb{N}. ((m \overset{h}{\not\leftrightarrow}_{t_1} n \wedge m' \overset{h'}{\not\leftrightarrow}_{t_1} n')) \Rightarrow$
 $(\exists t_2 < t_1. (m \overset{h}{\not\leftrightarrow}_{t_2} n \wedge m' \overset{h'}{\not\leftrightarrow}_{t_2} n' \wedge \forall t_3 < t_2. (m \overset{h}{\not\leftrightarrow}_{t_3} n \text{ iff } m' \overset{h'}{\not\leftrightarrow}_{t_3} n'))))$;

4. $\{t : h(t) \uparrow\} \subseteq \{t : h'(t) \uparrow\}$;
5. $(\forall t \in \mathbb{N}. m \xrightarrow{h} t n \text{ iff } m' \xrightarrow{h'} t n') \Rightarrow$
 $\forall t \in \mathbb{N}. ((\exists t_1 \in \mathbb{N}. \bar{h}(t-1) < t_1 < h(t) \wedge \forall a \in \Sigma. (m \Vdash_t a \text{ iff } n \Vdash_{t_1} a)) \Rightarrow$
 $(\exists t_2 \in \mathbb{N}. \bar{h}'(t-1) < t_2 < h'(t) \wedge \forall a \in \Sigma. (m' \Vdash_t a \text{ iff } n' \Vdash_{t_1} a))).$

This relation is in fact a pre-order and it defines the category $Comp(\Sigma)$: the objects of $Comp(\Sigma)$ are the morphisms of $Int(\Sigma)$ and there is a morphism between $h : m \rightarrow n$ and $h' : m' \rightarrow n'$ iff $h : m \rightarrow n \leq h' : m' \rightarrow n'$. This pre-order is used to give semantics of specifications with exceptions: if D is a specification and E an exception, m is a model of $D\mathbf{but}E$ if m is the domain of a \leq -minimal morphism of $Mor(E, D)$.

3 An example: a simple lift

As an illustration of the use of these definitions, consider a lift with three floors: `floor1`, `floor2` and `floor3`. The lift goes from `floor1` to `floor2`, and from this to `floor3`, and then it goes down to `floor2` and `floor1` again, and it never stops. When it gets to `floor1`, it goes up again. The following temporal formulae are a specification of this simple lift.

Specification of a lift system:

$G((\text{floor1} \vee \text{floor2} \vee \text{floor3}) \wedge \neg(\text{floor1} \wedge \text{floor2}) \wedge \neg(\text{floor2} \wedge \text{floor3}) \wedge \neg(\text{floor1} \wedge \text{floor3}))$	
$G(\text{up} \vee \text{down}) \wedge \neg(\text{up} \wedge \text{down})$	
$G(\text{floor1} \rightarrow X\text{up})$	$G(\text{floor3} \rightarrow X\text{down})$
$G(\text{floor2} \wedge \text{up}) \rightarrow X\text{up}$	$G((\text{floor2} \wedge \text{down}) \rightarrow X\text{down})$
$G(\text{up} \wedge \text{floor1}) \rightarrow X\text{floor2}$	$G(\text{down} \wedge \text{floor1}) \rightarrow X\text{floor1}$
$G(\text{up} \wedge \text{floor2}) \rightarrow X\text{floor3}$	$G(\text{down} \wedge \text{floor2}) \rightarrow X\text{floor1}$
$G(\text{up} \wedge \text{floor3}) \rightarrow X\text{floor3}$	$G(\text{down} \wedge \text{floor3}) \rightarrow X\text{floor2}$

Suppose now that we want to add a `call` button that makes the lift go down when it reaches `floor2`, independently if it was going up or down. When it gets to `floor2`, the `call` button is reset. The following represents what we want to add to the specification:

Adding a <code>call</code> button:	$G(\text{floor2} \rightarrow X\neg\text{call})$
$G(((\text{floor1} \vee \text{floor3}) \wedge \text{call}) \rightarrow X\text{call})$	$G((\text{call} \wedge \text{floor2}) \rightarrow X(\text{floor2} \wedge \text{down}))$

Instead of rewriting the specification from the beginning, and considering explicitly whether the `call` button has been pressed or not, we add these formulae as exceptions to the specification. Let Σ be the signature with $Att = \{\text{floor1}, \text{floor2}, \text{floor3}, \text{up}, \text{down}\}$ and $Act = \{\text{call}\}$. If D is the conjunction of the formulae in the specification and E the conjunction of the formulae that describe the `call` button, then the desired models are the models of $D\mathbf{but}E$ that satisfy the state restrictions: $G((\text{floor1} \vee \text{floor2} \vee \text{floor3}) \wedge \neg(\text{floor1} \wedge \text{floor2}) \wedge \neg(\text{floor2} \wedge \text{floor3}) \wedge \neg(\text{floor1} \wedge \text{floor3}))$ and $G((\text{up} \vee \text{down}) \wedge \neg(\text{up} \wedge \text{down}))$ (these state constraints have to be rigid to avoid the satisfaction of the exception by allowing the lift to be in two floors at the same time, or to be going up and down simultaneously. These formulae should not be overridden). We have that, if m is a model of these two axioms, and $m \Vdash D\mathbf{but}E$, then m is one of the desired models: if the lift is at `floor1` or `floor3`, or at `floor2` but the `call` button has not been pressed, then it behaves as before adding the `call` button; if the lift is at `floor2` and the `call` button has been pressed, then it behaves as specified in the formulae we added.

4 Concluding remarks

Starting from the default institution in [4], we define a temporal instantiation: a propositional linear temporal default institution. This consists of the usual temporal institution, but where the notion of morphisms between interpretations is changed, and it is extended with a pre-order between these morphisms. This notion of ordering between morphism of interpretations is used in order to deal with non-monotonicity in specifications. In particular, an example where a specification is reused but where exceptions to the previous behaviour are added is described, where the first specification is seen as default and a new module is considered, avoiding the necessitation of writing the whole specification from the beginning.

References

- [1] Joseph A. Goguen and Rod M. Burstall. Institutions: Abstract model theory for specification and programming. *Journal of the Association for Computing Machinery*, 39(1):95–146, January 1992.
- [2] S. Kraus, D. Lehmann, and M. Magidor. Non-monotonic reasoning, preferential models and cumulative logics. *Artificial Intelligence*, 44:167–207, 1990.
- [3] D. Makinson. General patterns in non-monotonic reasoning. In Dov Gabbay, C. J. Hogger, and J. A. Robinson, editors, *Nonmonotonic Reasoning and Uncertain Reasoning*, volume 3 of *Handbook of Logic in Artificial Intelligence and Logic Programming*, chapter 2, pages 35–110. Clarendon Press, Oxford, 1994.
- [4] Pierre-Yves Schobbens. Exceptions for algebraic specifications: On the meaning of ‘but’. *Science of Computer Programming*, 20(1-2), 1993.