Default Inheritance Reasoning in Hybrid KL-ONE-style Logics *

Umberto Straccia
Istituto di Elaborazione dell'Informazione
Consiglio Nazionale delle Ricerche
Via S. Maria, 46 - 56126 Pisa, Italy
E-mail: straccia@iei.pi.cnr.it

Abstract

Hybrid KL-ONE-style logics are knowledge representation formalisms of considerable applicative interest, as they are specifically oriented to the vast class of application domains that are describable by means of taxonomic organizations of complex objects. In this paper we consider the problem of endowing such logics with capabilities for *default inheritance reasoning*, a kind of default reasoning that is specifically oriented to reasoning on taxonomies. The formalism that results from our work has a reasonable and simple behaviour when dealing with the interplay of defeasible and strict inheritance of properties of complex objects.

1 Introduction

Hybrid KL-ONE-style logics (H-logics, for short - see e.g. [Nebel, 1990]) are knowledge representation formalisms of considerable applicative interest, as they are specifically oriented to the vast class of application domains that are describable by means of taxonomic organizations of complex objects. These formalisms, that may be seen as term-oriented syntactic variants of subsets of first order logic, are usually structured into two modules: the *terminological module*, allowing the representation of "concepts" and their implicit structuring according to a partial order, and the *assertional module*, allowing to state that given individuals are instances of the concepts described by means of the terminological module.

In the last ten years H-logics have been intensively investigated, with the attention of researchers especially focusing on the analysis of their logical and computational properties. Little attention, however, has been paid to the problem of endowing these logics with default reasoning capabilities. This is despite the fact that default reasoning is an important item on the list of *desiderata* of H-logics users (see e.g. [Peltason *et al.*, 1991]), and despite the fact that in most domains that have a taxonomic nature (e.g. the natural species), default information is abundant. Some researchers have recently addressed this problem (see e.g. [Baader and Hollunder, 1992; Brewka, 1987; Nado and Fikes, 1987]), but we think that their work has been successful only to a limited extent.

In this paper we will address the problem of extending H-logics with the ability to perform *default inheritance reasoning*, a kind of default reasoning that is specifically oriented to reasoning on taxonomies and that had been used mostly within formalisms of a much smaller expressive power than H-logics (see e.g. [Touretzky, 1986]). Most systems dealing with default inheritance reasoning solve "conflicts" according to some sort of *specialization principle*, i.e. by relying on the partial order according to which the knowledge base is structured: as a first approximation we can say that, in case of conflicts, a default $a \rightarrow b$ is "preferred" to another default $c \rightarrow \neg b$ if the precondition of the former precedes the precondition of the latter in the ordering; this accounts for the fact that the conclusion derivable from the former is more reliable than the one derivable from the latter.

This paper is organized as follows. In Section 2 we summarize the main notions of H-logics. In Section 3 we describe the integration of default inheritance capabilities into H-logics, while in Section 4 we present an algorithm for computing "extensions" (i.e. sets of derivable conclusions). In Section 5 we show that, unfortunately, computing an extension is a computationally hard problem in our formalism, even if its monotonic fragment is computationally tractable. Section 6 concludes.

2 Basic Notions of H-logics

In this section we present the basic notions of H-logics. For a more general presentation see [Nebel, 1990; Donini *et al.*, 1992].

2.1 The Terminological Module

We assume two disjoint alphabets of symbols, called *atoms* and *roles*. *Concepts* (denoted below by C and D)

*This work has been partially funded by the Progetto Finalizzato "Sistemi Informatici e Calcolo Parallelo" of the Italian National Council of Research

are formed out of atoms (denoted by A and B) according to the following syntax rule:

$$C, D \quad \rightarrow \quad A \mid C \sqcap D \mid \neg C \mid \forall R.C$$

This logic is usually known as $\mathcal{ALC}$. Other H-logics are obtained by allowing different concept-forming operators. An *interpretation* $\mathcal{I} = (\Delta^{\mathcal{I}}, \cdot^{\mathcal{I}})$ consists of a set $\Delta^{\mathcal{I}}$ (the *domain* of $\mathcal{I}$) and a function $\cdot^{\mathcal{I}}$ (the *interpretation function* of $\mathcal{I}$) which maps every concept to a subset of $\Delta^{\mathcal{I}}$ and every role to a subset of $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$ in such a way that the following equations are satisfied:

$$\begin{array}{rcl} (\neg C)^{\mathcal{I}} & = & \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\ (C \sqcap D)^{\mathcal{I}} & = & C^{\mathcal{I}} \cap D^{\mathcal{I}} \\ (\forall R.C)^{\mathcal{I}} & = & \{x \in \Delta^{\mathcal{I}} \mid \forall y : \langle x, y\rangle \in R^{\mathcal{I}} \Rightarrow y \in C^{\mathcal{I}}\} \end{array}$$

Let A be an atom and C be a concept. A *terminological axiom* is an expression of type $A \doteq C$ (*concept definition*) or $A \prec C$ (*concept specialization*). When $A \doteq C$ we say that A is the *name* of C. An interpretation $\mathcal{I}$ *satisfies* a terminological axiom δ iff

$$\begin{array}{ll} A^{\mathcal{I}} = C^{\mathcal{I}} & \text{if} \quad \delta = A \doteq C \\ A^{\mathcal{I}} \subseteq C^{\mathcal{I}} & \text{if} \quad \delta = A \prec C \end{array}$$

Therefore, a concept definition lists necessary and sufficient conditions for appartenance to A, whereas a concept specialization lists necessary conditions only.

A *terminology* (TBox) is a finite set $\mathcal{T}$ of terminological axioms. An interpretation $\mathcal{I}$ is a *model* of $\mathcal{T}$ iff it satisfies all terminological axioms in $\mathcal{T}$. We say that C *subsumes* D wrt $\mathcal{T}$ (written $D \sqsubseteq_{\mathcal{T}} C$) iff $D^{\mathcal{I}} \subseteq C^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$, and that C is *equivalent* to D wrt $\mathcal{T}$ (written $C =_{\mathcal{T}} D$) iff $C^{\mathcal{I}} = D^{\mathcal{I}}$ for every model $\mathcal{I}$ of $\mathcal{T}$. We will write $C \neq_{\mathcal{T}} D$ iff C is not equivalent to D wrt $\mathcal{T}$.

2.2 The Assertional Module

We assume an alphabet of symbols called *individuals* (denoted below by a and b), disjoint from the alphabets of atoms and roles. An *individual description* is an expression of type $a{:}C$, where a is an individual and C a concept. A *relation description* is an expression of type $\langle a, b\rangle{:}R$, where a and b are individuals and R is a role. A *description* is either an individual description or a relation description. Finally, an ABox is a finite set of descriptions.

With respect to the semantics, we extend the interpretation function $\cdot^{\mathcal{I}}$ of an interpretation $\mathcal{I}$ to individuals by mapping them to elements of the domain $\Delta^{\mathcal{I}}$, in such a way that $a^{\mathcal{I}} \neq b^{\mathcal{I}}$ if $a \neq b$[1].

An interpretation $\mathcal{I}$ *satisfies* a description α iff

$$\begin{array}{ll} a^{\mathcal{I}} \in C^{\mathcal{I}} & \text{if} \quad \alpha = a{:}C \\ \langle a^{\mathcal{I}}, b^{\mathcal{I}}\rangle \in R^{\mathcal{I}} & \text{if} \quad \alpha = \langle a, b\rangle{:}R \end{array}$$

An interpretation $\mathcal{I}$ is a *model* of an ABox $\mathcal{A}$ iff it satisfies all descriptions in $\mathcal{A}$.

[1]This restriction on individuals, the so-called *unique name assumption*, ensures that different individuals denotes different elements of the domain.

2.3 Reasoning in H-logics

A knowledge base in an H-logic is defined as the pair of a TBox and an ABox. We will say that an interpretation $\mathcal{I}$ is a *model of an ABox $\mathcal{A}$ wrt a TBox $\mathcal{T}$*, iff $\mathcal{I}$ is a model of $\mathcal{A}$ and of $\mathcal{T}$. An ABox $\mathcal{A}$ is *consistent* wrt a TBox $\mathcal{T}$ iff it has a model wrt $\mathcal{T}$. An ABox $\mathcal{A}$ and a TBox $\mathcal{T}$ *imply* a description α, written $\mathcal{A} \models_{\mathcal{T}} \alpha$, iff all models of $\mathcal{A}$ wrt $\mathcal{T}$ satisfy α.

With respect to an H-logic knowledge base, the following problems are usually considered interesting. Let $\mathcal{T}$ be a TBox and $\mathcal{A}$ be an ABox.

Subsumption problem: Does a concept C subsume a concept D wrt $\mathcal{T}$?

Consistency problem: Does there exist a model of $\mathcal{A}$ wrt $\mathcal{T}$?

Instance problem: Do $\mathcal{A}$ and $\mathcal{T}$ imply an individual description[2] $a{:}C$?

Realization problem: Let a be an individual occurring in $\mathcal{A}$. The set of the *most specific concepts* of which a is an instance is defined as

$$MSC_{\mathcal{A},\mathcal{T}}(a) = \{C \mid \mathcal{A} \models_{\mathcal{T}} a{:}C, \nexists D \text{ such that } C \neq_{\mathcal{T}} D, \mathcal{A} \models_{\mathcal{T}} a{:}D \text{ and } D \sqsubseteq_{\mathcal{T}} C\}.$$

What is the set of the most specific concepts of which a is an instance?

Retrieval problem: Let C be a concept. What is the set of all the individuals a such that $\mathcal{A} \models_{\mathcal{T}} a{:}C$?

Recent works on the computational properties of H- logics (see e.g. [Hollunder, 1990]) have shown that the above problems are reducible to the consistency problem, for which there exists a well-known technique based on constraint propagation. In fact, the subsumption problem is reducible to the instance problem since $D \sqsubseteq_{\mathcal{T}} C$ iff $\{a{:}D\} \models_{\mathcal{T}} a{:}C$, whereas the instance problem is reducible to the consistency problem since $\mathcal{A} \models_{\mathcal{T}} a{:}C$ iff $\mathcal{A} \cup \{a{:}\neg C\}$ has no model wrt $\mathcal{T}$. Finally, the realization problem can be easily solved by reducing it to the instance problem and the subsumption problem, while the retrieval problem can be reduced to the instance problem.

For reasons of space we will not describe the constraint propagation method here. The interested reader can consult [Hollunder, 1990].

3 Embedding Default Inheritance Reasoning in H-Logics

Until now we have described only the standard notions of H-logics. Let us now see how we can effectively for-

[2]Observe that, in our case, $\mathcal{A} \models_{\mathcal{T}} \langle a, b\rangle{:}R$ holds iff $\langle a, b\rangle{:}R \in \mathcal{A}$, since there are no role-forming constructs in the terminological module.

malize a minimal framework in which default inheritance reasoning in H-logics can be performed.

A *default of type 1* is an expression of the form $A \mapsto C$, while a *default of type 2* is an expression of the form $A \mapsto R.C$, where A is an atom, R is a role and C is a concept. Informally, a default $A \mapsto C$ means: "if a is an A and the assumption that a is a C does not lead to a "conflict", then assume that a is a C"; for example, the default `Bird`$\mapsto$`Fly` means: "if a is a bird and the assumption that a flies does not lead to a "conflict", assume that a flies". On the other hand, a default $A \mapsto R.C$ means: "if a is an A such that b is an R-filler of a and the assumption that b is a C does not lead to a conflict, then assume that b is a C"; for example, the default `Italian-University` $\mapsto$ `Faculty-Member.Italian` means: "if a is an Italian university, b is one of its faculty members, and the assumption that b is an Italian does not lead to a conflict, assume that b is an Italian".

In all systems for default inheritance reasoning conflicts are solved (whenever possible) by applying the so-called *specialization principle*, according to which properties belonging to a subclass are preferred to properties belonging to a superclass. For example, given that Opus is a penguin, that penguins typically do not fly, whereas birds do, the specialization principle let us infer, as desired, that Opus does not fly even if his being a bird might lead us to conclude differently. This is a default that we would call "of type 1"; the same principle is applicable to "type 2" defaults too: for example, given that the faculty members of Italian universities are typically Italian, given that the faculty members of South Tyrolean universities are typically not Italian, that South Tyrolean universities are Italian universities[3], that the University of Bozen is a South Tyrolean university, and that Professor Schmidt is a faculty member thereof, the specialization principle lets us to derive, as desired, that Schmidt is not Italian.

Whenever conflicts are not solvable by means of the specialization principle, like in the well-known "Nixon Diamond" example, multiple extensions are typically allowed.

We now go on to describe how our framework allows us to perform default inheritance reasoning in the context of H-logics. Our way to handle defaults will comply with the specialization principle and with "credulous reasoning" (because of this, multiple extensions are allowed). Let us define a *Hybrid Default Inheritance Theory* (HDIT) as a triple $\langle \mathcal{A}, \mathcal{T}, \mathcal{D} \rangle$, where $\mathcal{A}$ is an ABox, $\mathcal{T}$ is a TBox and $\mathcal{D}$ is a finite set of defaults. If $\mathcal{D}$ contains only defaults of type 1 (resp. type 2), we will say that the theory is an *HDIT of type 1 (resp. type 2)*.

From a technical point of view, our definition of "extension" will be similar to the one given by Reiter for Default Logic [Reiter, 1980], i.e. an extension is a fixpoint of a consequence relation. However, unlike in Default Logic, we will see that the specialization principle is "wired" in our definition. The following definition defines a precedence relation over atoms, $\preceq_{T,\bar{\mathcal{A}},a}$, induced by a HDIT T, an individual a and an ABox $\bar{\mathcal{A}}$. This relation will be used for fixing priority to defaults.

Definition 3.1 *Given an HDIT $T = \langle \mathcal{A}, \mathcal{T}, \mathcal{D} \rangle$, an individual a and an ABox $\bar{\mathcal{A}}$, the binary relation over atoms $\preceq_{T,\bar{\mathcal{A}},a}$ is defined as the smallest relation such that*

1. *if $\bar{\mathcal{A}} \models_{\mathcal{T}} a{:}A$ and $A \sqsubseteq_{\mathcal{T}} B$, then $A \preceq_{T,\bar{\mathcal{A}},a} B$;*
2. *if $\bar{\mathcal{A}} \models_{\mathcal{T}} a{:}A$, $\bar{\mathcal{A}} \models_{\mathcal{T}} a{:}C$ and $A \mapsto C \in \mathcal{D}$, then $A \preceq_{T,\bar{\mathcal{A}},a} B$ for all B's such that $C \sqsubseteq_{\mathcal{T}} B$;*
3. *$\preceq_{T,\bar{\mathcal{A}},a}$ is transitively closed.*

We define the *transitive closure of $\mathcal{A}$ wrt $\mathcal{T}$*, written $Tcl(\mathcal{A}, \mathcal{T})$, to be the set $\{\alpha \mid \mathcal{A} \models_{\mathcal{T}} \alpha\}$. Note that $\preceq_{T,\bar{\mathcal{A}},\cdot}$ can be seen as the taxonomy induced by the strict and defeasible information, altogether, in T and $\bar{\mathcal{A}}$. The following definition, which is the main definition of this paper, deals with the notion of "extension"; the definition is rather complex, and the examples that follow it will contribute in clarifying its import.

Definition 3.2 *Let $T = \langle \mathcal{A}, \mathcal{T}, \mathcal{D} \rangle$ be an HDIT. Let Γ be an operator such that, for any ABox $\bar{\mathcal{A}}$, $\Gamma(\bar{\mathcal{A}}, T)$ is the smallest ABox satisfying the following closure conditions:*

1. *$\mathcal{A} \subseteq \Gamma(\bar{\mathcal{A}}, T)$;*
2. *$\Gamma(\bar{\mathcal{A}}, T) = Tcl(\Gamma(\bar{\mathcal{A}}, T), \mathcal{T})$;*
3. *for all defaults $A \mapsto C \in \mathcal{D}$, for all descriptions $a{:}A \in \Gamma(\bar{\mathcal{A}}, T)$, it happens that $a{:}C \in \Gamma(\bar{\mathcal{A}}, T)$, unless $\bar{\mathcal{A}} \cup \{a{:}C\}$ is not consistent wrt $\mathcal{T}$ or there exists an atom B such that $B \preceq_{T,\bar{\mathcal{A}},a} A$ and $A \not\preceq_{T,\bar{\mathcal{A}},a} B$ and either*
   - *(a) $B \mapsto D \in \mathcal{D}$, and*
   - *(b) $\bar{\mathcal{A}} \cup \{a{:}C \sqcap D\}$ is not consistent wrt $\mathcal{T}$*

   or
   - *(a) $B \mapsto R.D \in \mathcal{D}$, and*
   - *(b) $\langle a, b \rangle{:}R \in \bar{\mathcal{A}}$, and*
   - *(c) $\bar{\mathcal{A}} \cup \{a{:}C, b{:}D\}$ is not consistent wrt $\mathcal{T}$*
4. *for all defaults $A \mapsto R.C \in \mathcal{D}$, for all descriptions $a{:}A \in \Gamma(\bar{\mathcal{A}}, T)$ such that $\langle a, b \rangle{:}R \in \Gamma(\bar{\mathcal{A}}, T)$, it happens that $b{:}C \in \Gamma(\bar{\mathcal{A}}, T)$, unless $\bar{\mathcal{A}} \cup \{b{:}C\}$ is not consistent wrt $\mathcal{T}$ or there exists an atom B such that $B \preceq_{T,\bar{\mathcal{A}},a} A$ and $A \not\preceq_{T,\bar{\mathcal{A}},a} B$ and either*
   - *(a) $B \mapsto D \in \mathcal{D}$, and*
   - *(b) $\bar{\mathcal{A}} \cup \{b{:}C, a{:}D\}$ is not consistent wrt $\mathcal{T}$*

   or
   - *(a) $B \mapsto R.D \in \mathcal{D}$, and*

[3] South Tyrol is, in fact, a German-speaking region of Italy.

(b) $\bar{\mathcal{A}} \cup \{b{:}C \sqcap D\}$ *is not consistent wrt* $\mathcal{T}$

An ABox $\mathcal{E}$ *is an* extension *of the HDIT theory* T *iff* $\mathcal{E} = \Gamma(\mathcal{E}, T)$*, i.e. iff* $\mathcal{E}$ *is a fixpoint of the operator* Γ.

In Definition 3.2, points 1 and 2 are as for Default Logic; point 3 handles conflicts between type 1 and type 1 defaults, and conflicts between type 1 and type 2 defaults; point 4 (symmetrically to point 3) handles conflicts between type 2 and type 1 defaults, and conflicts between type 2 and type 2 defaults. Definition 3.2 sanctions that (i) defaults which introduce conflicts cannot be applied and (ii) ambiguous situations are handled by means of multiple extensions (see Example 3.3 below).

We now consider some examples to show how Definition 3.2 works. The following example shows how "conflicts between fillers" are solved (type 2 vs. type 2).

Example 3.1 Let $T_1 = \langle \mathcal{A}_1, \mathcal{T}_1, \mathcal{D}_1 \rangle$ be the HDIT that formalizes our "Professors" example, with $\mathcal{A}_1 = \{\mathtt{p}{:}\mathtt{IU},\ \langle \mathtt{p},\mathtt{r}\rangle{:}\mathtt{FM},\ \mathtt{b}{:}\mathtt{STU},\ \langle \mathtt{b},\mathtt{s}\rangle{:}\mathtt{FM}\}$, $\mathcal{T}_1 = \{\mathtt{STU} \sqsubset \mathtt{IU}\}$ and $\mathcal{D}_1 = \{\mathtt{IU} \mapsto \mathtt{FM.I},\ \mathtt{STU} \mapsto \mathtt{FM.}\neg\mathtt{I}\}$. Let $\mathcal{E}_1 = Tcl(\mathcal{A}_1 \cup \{\mathtt{s}{:}\neg\mathtt{I}, \mathtt{r}{:}\mathtt{I}\})$ and $\Gamma(\mathcal{E}_1, T_1) = \mathcal{E}_1$. It is not hard to show that $\Gamma(\mathcal{E}_1, T_1)$ satisfies the conditions of Definition 3.2; therefore $\mathcal{E}_1$ is an extension of T_1. In $\mathcal{E}_1$ the inference according to which Professor Schmidt is not Italian is sanctioned, as we hoped.

In the next example we show how "conflicts between classes" are solved (type 1 vs. type 1).

Example 3.2 Let $T_2 = \langle \mathcal{A}_2, \mathcal{T}_2, \mathcal{D}_2 \rangle$ be an HDIT that formalizes our "Bird" example, with $\mathcal{A}_2 = \{\mathtt{o}{:}\mathtt{P}\}$, $\mathcal{T}_2 = \{\mathtt{P} \sqsubset \mathtt{B}\}$ and $\mathcal{D}_2 = \{\mathtt{B} \mapsto \mathtt{F},\ \mathtt{P} \mapsto \neg\mathtt{F}\}$. It follows that $\mathcal{E}_2 = Tcl(\mathcal{A}_2 \cup \{\mathtt{o}{:}\neg\ \mathtt{F}\})$ is an extension of T_2; hence, the inference according to which Opus does not fly is licensed.

The next example shows, instead, how ambiguous situations are handled, yielding multiple extensions.

Example 3.3 Let $T_3 = \langle \mathcal{A}_3, \mathcal{T}_3, \mathcal{D}_3 \rangle$ be an HDIT that formalizes an extended version of the well known "Nixon Diamond", with $\mathcal{A}_3 = \{\mathtt{a}{:}\mathtt{A} \sqcap \mathtt{C},\ \mathtt{a}{:}\neg\mathtt{B}\}$, $\mathcal{T}_3 = \emptyset$ and $\mathcal{D}_3 = \{\mathtt{C} \mapsto \mathtt{D},\ \mathtt{A} \mapsto \neg\mathtt{D},\ \mathtt{A} \mapsto \mathtt{B},\ \mathtt{B} \mapsto \mathtt{C}\}$. Let $\mathcal{E}_{3_1} = Tcl(\mathcal{A}_3 \cup \{\mathtt{a}{:}\mathtt{D}\})$ and $\mathcal{E}_{3_2} = Tcl(\mathcal{A}_3 \cup \{\mathtt{a}{:}\neg\mathtt{D}\})$ be ABoxes. It follows that $\Gamma(\mathcal{E}_{3_1}, T_3) = \mathcal{E}_{3_1}$ and $\Gamma(\mathcal{E}_{3_2}, T_3) = \mathcal{E}_{3_2}$, thus T_3 has two extensions, $\mathcal{E}_{3_1}$ and $\mathcal{E}_{3_2}$. Furthermore, note that $\mathtt{A} \not\preceq_{T,\mathcal{E}_{3_i},a} \mathtt{C}$, for $1 \leq i \leq 2$, since $\mathtt{a}{:}\mathtt{B} \notin \mathcal{E}_{3_i}$. In the case $\mathtt{a}{:}\neg\mathtt{B} \notin \mathcal{A}_3$, $\mathtt{A} \preceq_{T,\mathcal{E}'_3,a} \mathtt{C}$ would be the case, with $\mathcal{E}'_3 = Tcl(\mathcal{A}_3 \cup \{\mathtt{a}{:}\mathtt{B}, \mathtt{a}{:}\neg\mathtt{D}\})$, and thus $\mathcal{E}'_3$ would be the unique extension of T_3.

In the example below we show how conflicts between conflicting defaults of different type are solved (type 1 vs. type 2).

Example 3.4 Let $T_4 = \langle \mathcal{A}_4, \mathcal{T}_4, \mathcal{D}_4 \rangle$ be an HDIT, with $\mathcal{A}_4 = \{\mathtt{j}{:}\mathtt{RE},\ \langle \mathtt{j},\mathtt{g}\rangle{:}\mathtt{C}\}$, $\mathcal{T}_4 = \{\mathtt{RE} \sqsubset \mathtt{E},\ \mathtt{GT} \sqsubset \forall\mathtt{C.G}\}$ and $\mathcal{D}_4 = \{\mathtt{E} \mapsto \mathtt{GT},\ \mathtt{RE} \mapsto \mathtt{C.}\neg\mathtt{G}\}$, where RE, C, E, GT, G, stand for Royal_Elephant, Color, Elefant, Gray_Thing and Grey, respectively. It turns out that $\mathcal{E}_4 = Tcl(\mathcal{A}_4 \cup \{\mathtt{g}{:}\neg\mathtt{G}\})$ is an extension of T_4; in $\mathcal{E}_4$, as we hoped, the fact that the colour of j is not grey is sanctioned.

The next example shows, by means of a modified version of the "Royal-Elephant" example, how strict and defeasible information interact.

Example 3.5 Let $T_5 = \langle \mathcal{A}_5, \mathcal{T}_5, \mathcal{D}_5 \rangle$ be an HDIT with $\mathcal{A}_5 = \{\mathtt{j}{:}\mathtt{RE},\ \langle \mathtt{j},\mathtt{g}\rangle{:}\mathtt{C}\}$, $\mathcal{T}_5 = \{\mathtt{GT} \sqsubset \forall\mathtt{C.G},\ \mathtt{RE} \sqsubset \mathtt{E} \sqcap \forall\mathtt{C.}\neg\mathtt{G}\}$ and $\mathcal{D}_5 = \{\mathtt{E} \mapsto \mathtt{GT}\}$. It turns out that $\mathcal{E}_5 = Tcl(\mathcal{A}_5)$ is an extension of T_5. Note that the default $\mathtt{E} \mapsto \mathtt{GT}$ is not applied in the computation of $\mathcal{E}_5$, since Royal-Elephants are not gray: the application of defaults is "blocked" by the presence of strict information conflicting with them.

Unfortunately, some HDITs may not have extensions. The following example shows that there exists an HDIT of type 2 with no extension.

Example 3.6 Let $T_6 = \langle \mathcal{A}_6, \mathcal{T}_6, \mathcal{D}_6 \rangle$ be an HDIT with $\mathcal{A}_6 = \{\mathtt{a}{:}\mathtt{A},\ \langle \mathtt{a},\mathtt{a}\rangle{:}\mathtt{R}\ \}$, $\mathcal{T}_6 = \{\ \mathtt{A} \doteq \mathtt{B} \sqcap \mathtt{C},\ \mathtt{D} \doteq \mathtt{B} \sqcap \mathtt{C} \sqcap \mathtt{E}\}$ and $\mathcal{D}_6 = \{\mathtt{A} \mapsto \mathtt{R.E},\ \mathtt{D} \mapsto \mathtt{R.}\neg\mathtt{E}\}$. It is easy to show that T_6 has no extension.

It is easy to show that the same property holds for HDITs of type 1 too.

Finally, note that, given an HDIT $T_7 = \langle \mathcal{A}_7, \mathcal{T}_7, \mathcal{D}_7 \rangle$ such that $\mathcal{A}_7 = \{\mathtt{a}{:}\exists\mathtt{R}_1\mathtt{.}(\exists\mathtt{R}_2) \sqcap \mathtt{C}\}$ [4], $\mathcal{T}_7 = \emptyset$ and $\mathcal{D}_7 = \{\mathtt{C} \mapsto \mathtt{R}_2\mathtt{.D}\}$, $\mathcal{E}_7 = Tcl(\mathcal{A}_7, \mathcal{T}_7)$ does not contain $\mathtt{a}{:}\exists\mathtt{R}_1\mathtt{.}\exists\mathtt{R}_2\mathtt{.D}$, since in our definition of extension the role-filler must be explicitly known. This is contrary to intuitions: hence, in order to handle this case too, we must modify Definition 3.2, requiring that $\bar{\mathcal{A}}$ and $\Gamma(\bar{\mathcal{A}}, T)$ are "$\exists$-closed", i.e. if $\bar{\mathcal{A}} \models_{\mathcal{T}} a{:}\exists R.C$ and either $\langle a, b\rangle{:}R \notin \bar{\mathcal{A}}$ or $b{:}C \notin \bar{\mathcal{A}}$ is false, then $\langle a, b\rangle{:}R \in \bar{\mathcal{A}}$ and $b{:}C \in \bar{\mathcal{A}}$, for some new individual b. For example, given the HDIT above, the "$\exists$-closure" of $\mathcal{A}_7$ is $\bar{\mathcal{A}}_7 = \mathcal{A}_7 \cup \{\langle \mathtt{a},\mathtt{b}\rangle{:}\mathtt{R}_1,\ \mathtt{b}{:}(\exists\mathtt{R}_2) \sqcap \mathtt{C},\ \langle \mathtt{b},\mathtt{c}\rangle{:}\mathtt{R}_2\}$. It follows that $\Gamma(\bar{\mathcal{A}}_7, T_7) = Tcl(\bar{\mathcal{A}}_7 \cup \{\mathtt{c}{:}\mathtt{D}\}) = \mathcal{E}_7$ and $\Gamma(\bar{\mathcal{E}}_7, T_7) = \mathcal{E}_7$. Therefore, $\mathtt{a}{:}\exists\mathtt{R}_1\mathtt{.}\exists\mathtt{R}_2\mathtt{.D} \in \mathcal{E}_7$, as from intuitions. This case can be seen as a special case of the Skolemization process in Reiter's Default Theories.

The examples detailed above show that our formalism has a reasonable and simple behaviour. It is important to notice that these features are due to the fact that *our formalism is not just the "H-fragment" of a standard non-monotonic logic*, as its behaviour is informed by the "specialization principle", a principle that is not present into standard non-monotonic logics and that is the reason for what we claim to be the superior (in terms of intuitivity) behaviour of our formalism.

In the Section 4 we describe an algorithm for computing an extension (whenever it exists) of an HDIT T.

4 Computing Extensions

In order to describe our algorithm, we introduce some definitions. Let $T = \langle \mathcal{A}, \mathcal{T}, \mathcal{D} \rangle$ be an HDIT, a and b

[4] Note that $\exists R \equiv \exists R.\top$, with $\top \equiv \neg(\neg A \sqcap A)$.

Let $T=\langle \mathcal{A}, \mathcal{T}, \mathcal{D}\rangle$ be an HDIT;
begin
  $\mathcal{A}_0 \leftarrow \mathcal{A}; n \leftarrow 0;$
  repeat
    $n \leftarrow n+1; \omega_0 \leftarrow \mathcal{A}; D_0 \leftarrow \emptyset; i \leftarrow 0;$
    repeat
      $A_T^i \leftarrow (A_T(\omega_i) \cap A_T(\mathcal{A}_{n-1})) \setminus D_i;$
      if not $empty(A_T^i)$
        then choose γ **from** A_T^i;
          $D_{i+1} \leftarrow D_i \cup \{\gamma\};$
          $\omega_{i+1} \leftarrow \omega_i \cup \{Consequent(\gamma)\};$
      endif
      $i \leftarrow i+1;$
    until $empty(A_T^{i-1})$;
    $\mathcal{A}_n \leftarrow \omega_{i-1};$
  until $\mathcal{A}_n = \mathcal{A}_{n-1}$;
end

Table 1: The EXT algorithm computes an extension of an HDIT.

two individuals, $\bar{\mathcal{A}}$ an ABox, $\delta \in \mathcal{D}$ a default of the form $A \mapsto C$, $\sigma \in \mathcal{D}$ a default of the form $A \mapsto R.C$. An *instantiated default* of T is either a pair $\gamma=\langle a, \delta\rangle$ or a triple $\gamma=\langle a, b, \sigma\rangle$. The function *Consequent* is defined to be such that *Consequent*($\langle a, \delta\rangle$)=a:C and *Consequent*($\langle a, b, \sigma\rangle$)=$b$:$C$. We say that an instantiated default $\langle a, \delta\rangle$ is *applicable in* $\bar{\mathcal{A}}$ *wrt* T iff $\bar{\mathcal{A}} \models_{\mathcal{T}} a{:}A$ and the "unless" conditions of Definition 3.2, point 3, do not hold. Analogously, an instantiated default $\langle a, b, \sigma\rangle$ is *applicable in* $\bar{\mathcal{A}}$ *wrt* T iff $\bar{\mathcal{A}} \models_{\mathcal{T}} a{:}A$, $\langle a, b\rangle{:}R \in \bar{\mathcal{A}}$ and the "unless" conditions of Definition 3.2, point 4, do not hold. We will write $A_T(\bar{\mathcal{A}})$ as shorthand for the set of applicable defaults in $\bar{\mathcal{A}}$ wrt T.

Let us now discuss the EXT algorithm for computing extensions, which is similar to that described in [Etherington, 1988] for Reiter's Default Theories. The extension is built by a series of subsequent approximations, each of them being an ABox. Each approximation $\mathcal{A}_n$ is built from the first component $\mathcal{A}$ of an HDIT T by using applicable defaults, one at time. At each step, the instantiated default to be applied is chosen from those which have not yet been considered and which were applicable in the previous approximation and still are in the current "state" of the current approximation. When no more instantiated defaults are applicable, the algorithm continues with the next approximation. If two successive approximations are the same set, the algorithm is said to *converge*. The EXT algorithm is detailed in Table 1.

The following correctness and completeness theorem states that all and only the extensions of an HDIT T can be computed by the algorithm.

Theorem 4.1 *Let $T = \langle \mathcal{A}, \mathcal{T}, \mathcal{D}\rangle$ be an HDIT. $\mathcal{E}$ is an extension of T iff the application of the* EXT *algorithm to T has a converging computation such that $\mathcal{A}_n = \mathcal{A}_{n-1}$ and $Tcl(\mathcal{A}_n, \mathcal{T}) = \mathcal{E}$.*

Therefore, it is not necessary to compute an entire extension $\mathcal{E}$, since the corresponding $\mathcal{A}_n$ is sufficient. Furthermore, observe that, according to Theorem 4.1, the computation converges if and only if there exists an extension. The following example shows that, if there are no extensions, the computation does not converge.

Example 4.1 Consider the HDIT T_6 of Example 3.6. It turns out that each approximation $\mathcal{A}_i$ is such that $\mathcal{A}_{2k} = \mathcal{A}_6$ and $\mathcal{A}_{2k+1} = \mathcal{A}_6 \cup \{\mathtt{a{:}E}\}$, for each $k \geq 0$. Therefore $\mathcal{A}_{2k} \neq \mathcal{A}_{2k+1}$ for each $k \geq 0$, and the computation never stops.

Note however that, since the set of instantiated defaults of an HDIT T is finite, there is only a finite number of ABoxes $\mathcal{A}_n$, computable by the EXT algorithm, such that $Tcl(\mathcal{A}_n, \mathcal{T})$ is an extension; it is thus possible to decide if there are cyclic computations like the one in Example 4.1. Therefore, it is straightforward to modify the EXT algorithm so that the computation always converges.

5 Complexity

In this section we will analyze the computational complexity of computing an extension. In order to do so, first of all we will restrict our attention to HDITs for which deciding if an instantiated default is applicable is computable in polynomial time; for all the other HDITs, the problem of computing an extension is obviously intractable. For our purpose, let $T = \langle \mathcal{A}, \mathcal{T}, \mathcal{D}\rangle$ be an HDIT and C a concept. *C is a simple concept wrt $\mathcal{T}$* iff C is a conjunction of atoms or negated atoms which are not names in $\mathcal{T}$. We will say that *T is a restricted HDIT of type 1* iff (a) $\mathcal{A}$ contains only individual descriptions of the form $a{:}C$, where C is a simple concept wrt $\mathcal{T}$; (b) $\mathcal{T}$ contains only terminological axioms of the form $A \doteq C$, where C is an atom (or the negation of an atom) which is not a name in $\mathcal{T}$, or axioms of the form $A \sqsubset B$, where B is not a name in $\mathcal{T}$, and (c) $\mathcal{D}$ is a set of defaults of the form $A \mapsto C$, where C is a simple concept wrt $\mathcal{T}$. The definition of a *restricted HDITs of type 2* is similar to the case of type 1, but for the fact that $\mathcal{D}$ contains only defaults of the form $A \mapsto S.C$, where C is a simple concept wrt $\mathcal{T}$. It may be shown that, in the case of restricted HDITs (either of type 1 or of type 2) deciding if an instantiated default is applicable is computable in polynomial time.

Unfortunately, notwithstanding the restrictions on HDITs, the problem of computing an extension is a hard problem.

Theorem 5.1 *Finding an extension of a restricted HDIT of type 1 (or determining that it has no extension) is NP-Hard.*

The proof of this theorem depends on the following reduction of the 3SAT problem [Garey and Johnson, 1979] to the problem of determining an extension. The reduction is similar to that described in [Kautz, 1989]. For a

propositional formula σ in 3CNF consider the restricted HDIT of type 1, T_σ, obtained exactly from the expressions which appear in the following rules. (A) for every symbol P which occurs in σ, the following expressions appear in T_σ, where A is a new atom and a an individual: $A \mapsto P$, $A \mapsto \neg P$, $a{:}A$. (B) for each clause $X \vee Y \vee Z$ of σ, the following expressions appear in T_σ, where A, B, F and F_{xyz} are new atoms: $A \doteq \neg X$ (we will use P instead of $\neg\neg P$), $A \mapsto F_{xyz} \sqcap \neg Y \sqcap \neg Z$ and $F_{xyz} \mapsto F \sqcap \neg B$. (C) $F \lessdot F_{xyz}$ and $F \mapsto B$ appear in T_σ. We can show that σ is satisfiable iff T_σ has an extension[5]. Therefore the problem of determining if a restricted HDIT of type 1 has an extension is also NP-Hard.

The same property holds for restricted HDITs of type 2. In fact we can consider the same HDIT T_σ as above except that expressions of the form $A \mapsto C$ are substituted with the expressions of the form $A \mapsto R.C$ and $\langle a, a\rangle{:}R$.

Theorem 5.2 *Finding an extension of a restricted HDIT of type 2 (or determining that it has no extension) is NP-Hard.*

Among the problems listed in Section 2.3, let us analyze the complexity of the instance problem with respect to our formalism (we will leave the consistency problem aside as, in our context, it corresponds to the problem of answering if there exists a consistent extension). Let us first define what the "instance problem" amounts to in our framework. One can say that, due to the possible presence of multiple extensions, there are actually two types of instance problems: the "credulous" one and the "skeptical" one. Let T be an HDIT and $a{:}C$ be an individual description; the *credulous instance problem* is the problem of determining if there is an extension $\mathcal{E}$ of T such that $a{:}C \in \mathcal{E}$ (in this case we will say that a is a credulous instance of C wrt T). The *skeptical instance problem* is defined as the problem of determining if $a{:}C$ is an element of the intersection of all the extensions of T (in this case we will say that a is a skeptical instance of C wrt T). One can easily check that the realization problem can be defined in terms of the subsumption problem and the credulous/skeptical instance problem, whereas the retrieval problem can be defined in terms of the credulous/skeptical instance problem.

Since, given an HDIT T, $a{:}\top$ is a credulous instance of C wrt T iff T has an extension, from Theorem 5.1 and Theorem 5.2 we have the following Corollary:

Corollary 5.1 *The credulous instance problem for restricted HDITs (either of type 1 or of type 2) is NP-Hard.*

Instead:

Theorem 5.3 *The skeptical instance problem for restricted HDITs (either of type 1 or of type 2) is co-NP-Hard.*

[5]Note that if σ is satisfiable it must not be the case that $a{:}F \in \mathcal{E}$, where $\mathcal{E}$ is an extension of T_σ.

The proof of Theorem 5.3 wrt HDITs of type 1 (resp. type 2) is similar to the one for Theorem 5.1 (resp. Theorem 5.2), except that in the reduction we do not consider rule (C). Therefore, an arbitrary propositional 3CNF formula σ is unsatisfiable iff $a{:}F$ holds in all extensions of T_σ.

6 Conclusion

In this paper we have shown how we can extend H-logics in such a way that they allow default inheritance reasoning, thus creating a formalism that combines the expressive power of a language for the description of taxonomic organizations of complex objects with the taxonomy-oriented style of default reasoning which is typical of "inheritance systems with exceptions".

The importance of our work lies in bringing these default reasoning capabilities into a family of logics that have gained wider and wider acceptance because of their reasonable expressive power, good computational properties, intuitive "object-oriented" syntax and wide spectrum of applicability.

We have presented an algorithm that computes extensions and shown that it is correct and complete. Moreover, we have shown that, even if the "H-fragment" of our formalism were tractable, computing an extension or deciding that there are no extensions would be NP-Hard; similarly, also the (credulous or skeptical) instance problem, the realization problem and the retrieval problem are intractable.

Our formalism has been designed with the aim of providing the *minimal* framework that would allow one to study the interaction of H-knowledge and default inheritance knowledge in a meaningful way. Quite obviously, extensions to this framework may be conceived that enable the expression of more general concepts: any H-logic that contains $\mathcal{ALC}$ may be profitably used.

For that matter, the formalism could also be straightforwardly extended to the representation of default rules of a different nature; for example, we might have defaults of the form $\alpha : \beta_1, \ldots, \beta_n/\gamma$, where α, $\beta_1, \ldots, \beta_n$ and γ are $\mathcal{ALC}$ concepts. Recently, Baader and Hollunder [1992] have embedded this types of defaults into H-logics. However, their formalism, unlike ours, does not support the taxonomy-oriented brand of non-monotonic reasoning informed by the "principle of specialization"; it can thus be seen as (and has the disadvantages of) an "H-fragment" of a standard non-monotonic logic (in their case, Reiter's Default Logic).

References

[Baader and Hollunder, 1992] Franz Baader and Bernhard Hollunder. Embedding defaults into terminological knowledge representation formalisms. In *Proceedings of KR-92, 3rd International Conference on Knowledge Representation and Reasoning*, Cambridge, MA, 1992.

[Brewka, 1987] Gerhard Brewka. The logic of inheritance in frame systems. In *Proceedings of IJCAI-87, 10th International Joint Conference on Artificial Intelligence*, pages 483–488, Milano, Italy, 1987.

[Donini *et al.*, 1992] Francesco M. Donini, Maurizio Lenzerini, Daniele Nardi, and Andrea Schaerf. From subsumption to instance checking. Technical Report 15.92, Università di Roma "La Sapienza", Dipartimento di Informatica e Sistemistica, Roma, Italy, 1992.

[Etherington and Reiter, 1983] David W. Etherington and Raymond Reiter. On inheritance hierarchies with exceptions. In *Proceedings of AAAI-83, 3rd Conference of the American Association for Artificial Intelligence*, pages 24–26, Washington, DC, 1983.

[Etherington, 1988] David W. Etherington. *Reasoning with incomplete information.* Morgan Kaufmann, Los Altos, CA, 1988.

[Garey and Johnson, 1979] Michael R. Garey and David S. Johnson. *Computers and intractability. A guide to the theory of NP-completeness.* Freeman, New York, NY, 1979.

[Hollunder, 1990] Bernhard Hollunder. Hybrid inferences in KL-ONE-based knowledge representation systems. In *Proceedings of GWAI-90, 14th German Workshop on Artificial Intelligence*, pages 38–47, Eringerfeld, FRG, 1990.

[Kautz, 1989] Henry. A Kautz and Bart Selman. Hard problems for simple defaults. In *Proceedings of KR-89, 1st International Conference on Principles of Knowledge Representation and Reasoning*, pages 189–197, Toronto, Ont., 1989.

[Nado and Fikes, 1987] Robert A. Nado and Richard E. Fikes. Semantically sound inheritance for a formally defined frame language with defaults. In *Proceedings of AAAI-87, 6th Conference of the American Association for Artificial Intelligence*, pages 443–448, Seattle, WA, 1987.

[Nebel, 1990] Bernhard Nebel. *Reasoning and revision in hybrid representation systems.* Springer, Heidelberg, FRG, 1990.

[Peltason *et al.*, 1991] Christof Peltason, Kai von Luck, and Carsten Kindermann (eds.). Proceedings of the Terminological Logics Users Workshop. KIT Report 95, Technical University Berlin, Berlin, FRG, 1991.

[Reiter, 1980] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.

[Touretzky, 1986] David S. Touretzky. *The mathematics of inheritance systems.* Pitman, London, GB, 1986.