Embedding Defaults into Terminological Knowledge Representation Formalisms

Franz Baader and Bernhard Hollunder
German Research Center for AI (DFKI)
Stuhlsatzenhausweg 3, 6600 Saarbrücken 11, Germany
e-mail: ⟨*last name*⟩@dfki.uni-sb.de

Abstract

We consider the problem of integrating Reiter's default logic into terminological representation systems. It turns out that such an integration is less straightforward than we expected, considering the fact that the terminological language is a decidable sublanguage of first-order logic. Semantically, one has the unpleasant effect that the consequences of a terminological default theory may be rather unintuitive, and may even vary with the syntactic structure of equivalent concept expressions. This is due to the unsatisfactory treatment of open defaults via Skolemization in Reiter's semantics. On the algorithmic side, we show that this treatment may lead to an undecidable default consequence relation, even though our base language is decidable, and we have only finitely many (open) defaults. Because of these problems, we then consider a restricted semantics for open defaults in our terminological default theories: default rules are only applied to individuals that are explicitly present in the knowledge base. In this semantics it is possible to compute all extensions of a finite terminological default theory, which means that this type of default reasoning is decidable.

Contents

1 Introduction

Terminological representation systems are used to represent the taxonomic and conceptual knowledge of a problem domain in a structured and well-formed way. To describe this kind of knowledge, one starts with atomic concepts (unary predicates) and roles (binary predicates), and defines more complex concepts using the operations provided by the concept language of the particular formalism. In addition to this concept description formalism, most of these systems also have an assertional component. One can for example state that an individual is an instance of a concept, or that two individuals are connected by a role.

In terminological representation formalisms, the concept descriptions are interpreted as universal statements, which means, unlike frame languages, they do not allow for exceptions. As a consequence, the system can use descriptions to automatically insert concepts at the proper place in the taxonomy (classification), and it can use the facts stated about individuals to deduce to which concepts they must belong (realization). For example, one could define the concept *Mammal* as an *Animal* that feeds its young with *Milk*, where *feeds-young-with* is used as a role. If the concept *Platypus*[1] is defined as an *Animal* that *lives-in* the *Water*, feeds its young with *Milk*, and *reproduce*s with *Eggs*, then the system will recognize that *Platypus* is a subconcept of *Mammal*.

However, commonsense reasoning is often based on assumptions that may ultimately be shown to be false. In our example, one might want to assume by default that *Mammal*s *reproduce Viviparously*. Only if it is known that a specific mammal reproduces with eggs, should this assumption be cancelled. If one wants to use terminological systems for this kind of commonsense reasoning, one needs a formalism that can handle such default assumptions, but does not destroy the definitional character of concept descriptions—because otherwise the advantage of automatic concept classification, etc., would be lost (see [5]). Besides the general arguments for the importance of reasoning with defaults, which can be found in the nonmonotonic reasoning literature, the need for embedding defaults into terminological representation formalisms is also substantiated by the fact that this is an important item on the wish list of users of terminological representation systems (see e.g. [22]).

Several existing terminological systems, such as BACK [20], CLASSIC [6], K-REP [16], LOOM [19], or SB-ONE [14], have been or will be extended to provide the user with some kind of default reasoning facilities. However, as

[1] We are taking this as our exceptional animal, in view of the fact that last IJCAI was in Australia, and not in the Antarctic.

the designers of these systems themselves point out, these approaches usually have an ad hoc character, and are not equipped with a formal semantics. For example, defaults in the FAME system, which is built using K-REP, "will not be complete (or even consistent)" ([16], p.11) unless the user is very careful when using them. In CLASSIC, "a limited form of defaults can be represented with the aid of rules and test functions." However, the user is warned to "use this trick with extreme caution" ([6], p.45,46).

Our arguments for the importance of default extensions for terminological representation languages so far were given from the viewpoint of the terminological systems community. However, these investigations may also be of interest for research in nonmonotonic reasoning itself. Most nonmonotonic reasoning formalisms (e.g. Reiter's default logic [25], Circumscription [17]) use full first-order predicate logic as their base language. In this general form, the formalisms are usually highly undecidable (see e.g. [25] Theorem 4.9). For this reason, work on decision procedures for decidable subcases was mostly restricted to propositional logic (see e.g. [13]), thus leaving the wide gap between propositional logic and full first-order logic almost unexplored. Since most terminological representation languages can be viewed as decidable subclasses of first-order logic—but are nevertheless much more expressive than propositional logic—they can serve as interesting test cases for nonmonotonic reasoning formalisms. We shall see that this not only applies for algorithmic, but also for semantic considerations.

We shall here consider the problem of integrating Reiter's default logic into a terminological representation formalism. This treatment of defaults in terminological systems has already been proposed by Brachman and Schmolze [7], but to the best of our knowledge, this proposal was never followed up. Reiter's default rule approach seems to fit well into the philosophy of terminological systems because most of them already provide their users with a form of "monotonic" rules. These rules can be considered as special default rules where the justifications—which make the behaviour of default rules nonmonotonic—are absent.

At first sight, one might think that, from a semantic point of view, the proposed integration should be unproblematic. In fact, the terminological representation language we shall consider (see Section 2) is a sublanguage of first-order logic, and Reiter's semantics has been formulated for full first-order logic. However, on closer inspection it turns out that one runs into severe problems, due to the unsatisfactory treatment of open defaults by Skolemization (see Section 3).

A similar problem arises when considering the integration from the algorithmic point of view. In the abstract of their paper on how to compute extensions for default logic, Junker and Konolige [12] write that their method

is applicable if the default theory "consists of a finite number of defaults and premises and classical derivability for the base language is decidable." A related formulation can be found in the abstract of Schwind and Risch's paper on the same topic [28]. Since our base language is decidable, and we certainly do not want to have infinitely many default rules, these methods seem to apply in our case. However, a closer look at the papers reveals that by "a finite number of defaults" it is meant "a finite number of *closed* defaults." But the default rules we want to consider are *open* defaults. In fact, as already pointed out by Reiter ([25], p.115) "the genuinely interesting cases involve open defaults." In Section 4 we shall show that, with our (decidable) terminological language as base language, a finite set of premises and open defaults may lead to an undecidable default consequence problem, if the open defaults are treated as proposed by Reiter ([25], Section 7.1).

Because of the semantic as well as algorithmic problems posed by Reiter's treatment of open defaults, we shall consider a restricted semantics for open defaults in our integration: default rules are only applied to individuals that are *explicitly* present in the assertional part (ABox) of the knowledge base. Though one may thus lose some intuitive default inferences, this treatment of default rules is akin to the treatment of the monotonic rules in terminological systems such as CLASSIC.

With this restricted semantics, a finite set of open defaults stands for a set of closed defaults that is finite as well. Thus the above-mentioned methods of Schwind and Risch and of Junker and Konolige can be applied to compute extensions (see Section 5). In order to make these methods more efficient, one has to solve certain algorithmic problems for the terminological language. For Junker and Konolige's methods one has to find minimal proofs for assertional facts—which can be seen as an abduction problem for ABoxes—and for Schwind and Risch's method one must find maximal consistent sets of assertional facts. In Section 6 we shall point out how the tableaux-based methods for assertional reasoning developed in our group ([10, 2]) can be modified to solve these problems.

2 The Representation Formalisms

First we shall briefly review the terminological language $\mathcal{ALCF}$ [11] and Reiter's default logic. Then terminological default logic is defined as the specialization of default logic to $\mathcal{ALCF}$. Finally an example will illustrate why Reiter uses Skolemization in his semantics for open default theories.

2.1 The terminological language $\mathcal{ALCF}$

Terminological knowledge representation formalisms can be used to define the relevant concepts of a problem domain (terminological knowledge), and to describe objects of this domain with respect to their relation to concepts and their interrelation with each other (assertional knowledge). Depending on which constructs are allowed for building concept descriptions we get different terminological languages. In the present paper we restrict our attention to the language $\mathcal{ALCF}$.

Definition 2.1 *The* terminological part *of the language $\mathcal{ALCF}$ consists of the following concept description formalism. The* concept terms *of this formalism are built from* concept, role and attribute names *using the constructors* conjunction *($C \sqcap D$),* disjunction *($C \sqcup D$),* negation *($\neg C$),* exists-restriction *($\exists R.C$),* value-restriction *($\forall R.C$), and* agreement *($u \doteq v$). Here C, D stand for concept terms, R for a role or attribute name, and u, v for finite sequences of attribute names.*
The assertional part *of our language allows us to assert facts concerning particular objects. These objects are referred to by* individual names*, and we can state that an object belongs to a concept (written $C(a)$), or that two objects are related by a role or attribute (written $R(a, b)$). Here a, b stand for individual names, C for a concept term, and R for a role or attribute name. A finite set of such facts is called an* ABox*.*

The semantics of an ABox can either be given directly by defining interpretations and models, or by a translation into first-order logic. In order to make the fact explicit that we are dealing with a sublanguage of first-order logic, we choose the second option.

Concept names are considered as symbols for unary predicates, and role and attribute names as symbols for binary predicates. Consequently, concept names A are translated into (atomic) formulae $A(x)$ with one free variable, and role and attribute names R into (atomic) formulae $R(x, y)$ with two free variables. The attributes have to be interpreted as partial functions, which can be expressed by a formula $\forall x, y, z\colon (f(x, y) \wedge f(x, z) \rightarrow y = z)$ for each attribute name f.

Concept terms are also translated into formulae with one free variable. The semantics of conjunction, disjunction, and negation are defined in the obvious way, i.e., $(C \sqcap D)(x) := C(x) \wedge D(x)$, $(C \sqcup D)(x) := C(x) \vee D(x)$, and $(\neg C)(x) := \neg C(x)$. For value-restrictions we define $(\forall R.C)(x) := \forall y\colon (R(x, y) \rightarrow C(y))$, and the semantics of exists-restrictions is given by $(\exists R.C)(x) := \exists y\colon (R(x, y) \wedge C(y))$. Let $u = f_1 \cdots f_m$, and $v = g_1 \cdots g_n$ be sequences of attributes. The agreement construct built from these sequences

is translated into the formula $(u \doteq v)(x) := \exists y_1, \ldots, y_m, z_1, \ldots z_n$: $(f_1(x, y_1) \wedge \ldots f_m(y_{m-1}, y_m) \wedge g_1(x, z_1) \wedge \ldots g_n(z_{n-1}, z_n) \wedge y_m = z_n)$.

The individual names of the Abox are considered as constant symbols. In terminological systems one usually has a *unique name assumption*, which can be expressed by the formulae $a \neq b$ for all distinct individual names a, b. The formula corresponding to the assertional fact $C(a)$ (resp. $R(a, b)$) is obtained by replacing the free variable(s) in the formula corresponding to C (resp. R) by a (resp. a, b). To sum up, an ABox is translated into a set of first order formulae consisting of the translations of the ABox facts, the formulae expressing unique name assumption, and the formulae expressing that attributes are partial functions.

The basic inference service for ABoxes is called *instantiation*. It answers the question of whether (the translation of) a given ABox fact $C(a)$ is a (logical) consequence of (the translation of) a given ABox $\mathcal{A}$. If the answer is yes we say that *a is an instance of C with respect to $\mathcal{A}$* ($\mathcal{A} \models C(a)$). Algorithms which solve this inference problem have, for example, been described in [10, 2].

2.2 Reiter's default logic

Reiter [25] deals with the problem of how to formalize nonmonotonic reasoning by introducing nonstandard, nonmonotonic inference rules, which he calls default rules. A *default rule* is any expression of the form

$$\frac{\alpha : \beta_1, \ldots, \beta_n}{\gamma},$$

where α, β_1, ..., β_n, γ are first-order formulae. Here α is called the *prerequisite* of the rule, $\beta_1, \ldots, \beta_n$ are its *justifications*, and γ is its *consequent*. For a set of default rules $\mathcal{D}$, we denote the sets of formulae occurring as prerequisites, justifications, and consequents in $\mathcal{D}$ by $Pre(\mathcal{D})$, $Jus(\mathcal{D})$, and $Con(\mathcal{D})$, respectively.

A default rule is *closed* iff α, $\beta_1, \ldots, \beta_n$, γ do not contain free variables. A *default theory* is a pair $(\mathcal{W}, \mathcal{D})$ where $\mathcal{W}$ is a set of closed first-order formulae (the world description) and $\mathcal{D}$ is a set of default rules. A default theory is *closed* iff all its default rules are closed.

Intuitively, a *closed* default rule can be applied, i.e., its consequent is added to the current set of beliefs, if its prerequisite is already believed and all its justifications are consistent with the set of beliefs. Formally, the consequences of a *closed* default theory are defined with reference to the notion of an *extension*, which is a set of deductively closed first-order formulae defined by a fixed point construction (see [25], p.89). In general,

a default theory may have more than one extension, or even no extension. Depending on whether one wants to employ skeptical or credulous reasoning, a closed formula δ is a *consequence of a closed default theory* iff it is in all extensions or if it is in at least one extension of the theory. In general, this consequence relation is not even recursively enumerable (see [25], Theorem 4.9).

Reiter also gives an alternative characterization of an extension, which we shall use, in a slightly modified way, as the definition of extension. Here and in the following, $Th(?)$ stands for the deductive closure of a set of formulae $?$.

Definition 2.2 *Let E be a set of closed formulae, and $(\mathcal{W}, \mathcal{D})$ be a closed default theory. We define*

$$E_0 := \mathcal{W}$$

and for all $i \geq 0$

$$E_{i+1} := E_i \cup \{\gamma \mid \alpha : \beta_1, \ldots, \beta_n/\gamma \in \mathcal{D}, \alpha \in Th(E_i), \\ \text{and } \neg\beta_1, \ldots, \neg\beta_n \notin Th(E)\}.$$

Then $Th(E)$ is an extension of $(\mathcal{W}, \mathcal{D})$ iff

$$Th(E) = \bigcup_{i=0}^{\infty} Th(E_i).$$

Note that the extension $Th(E)$ to be constructed by this iteration process occurs in the definition of each iteration step. Since we are only adding consequents of defaults during the iteration, any extension $Th(E)$ of $(\mathcal{W}, \mathcal{D})$ is of the form $Th(\mathcal{W} \cup Con(\mathcal{D}'))$ for a subset $\mathcal{D}'$ of $\mathcal{D}$. Reiter shows ([25], Theorem 2.5) that the set

$$\widehat{\mathcal{D}} = \Big\{\frac{\alpha : \beta_1, \ldots, \beta_n}{\gamma} \in \mathcal{D} \mid \ \alpha \in Th(E) \text{ and } \neg\beta_1, \ldots, \neg\beta_n \notin Th(E)\Big\}.$$

always satisfies this property. For this reason it is called *set of generating defaults* for the extension $Th(E)$. Another easy consequence of Definition 2.2 is that $(\mathcal{W}, \mathcal{D})$ has an inconsistent extension iff $\mathcal{W}$ is inconsistent.

Reiter defines *extensions of arbitrary default theories* $(\mathcal{W}, \mathcal{D})$, i.e., default theories with open defaults, as follows. First, the formulae of $\mathcal{W}$ and the consequents of the defaults are Skolemized (see [25], Section 7). Second, a set $\mathcal{D}'$ of closed default rules is generated by taking all ground instances (over the initial signature together with the newly introduced Skolem functions) of the defaults of $\mathcal{D}$. Now E is an *extension* of $(\mathcal{W}, \mathcal{D})$ iff E is an extension of the closed default theory $(\mathcal{W}', \mathcal{D}')$, where $\mathcal{W}'$ is the Skolemized form of $\mathcal{W}$. The reason for Skolemizing before building ground instances will be explained by an example in Subsection 2.4.

2.3 Terminological default theories

A *terminological default theory* is a pair $(\mathcal{A}, \mathcal{D})$ where $\mathcal{A}$ is an ABox and $\mathcal{D}$ is a finite set of default rules whose prerequisites, justifications, and consequents are concept terms. Obviously, since ABoxes can be seen as sets of closed formulae, and since concept terms can be seen as formulae with one free variable,[2] terminological default theories are subsumed by Reiter's notion of an open default theory.

However, as for ABox reasoning without defaults, we are not interested in arbitrary formulae as consequences of a terminological default theory $(\mathcal{A}, \mathcal{D})$, but only in assertional facts of the form $C(a)$, where a is an individual name occurring in the original ABox $\mathcal{A}$.

2.4 Why is Skolemization necessary ?

The following example shows that intuitively valid consequences would get lost if one did not Skolemize. Suppose that our ABox consists of the fact that *Tom* has some child who is a doctor, i.e., $\mathcal{A} = \{(\exists \mathit{child}.\mathit{doctor})(\mathit{Tom})\}$. By default we want to conclude that doctors usually are rich persons, and usually have children who are doctors. Thus $\mathcal{D}$ consists of the default rules

$$\frac{\mathit{doctor} : \mathit{rich\text{-}person}}{\mathit{rich\text{-}person}} \quad \text{and} \quad \frac{\mathit{doctor} : \exists \mathit{child}.\mathit{doctor}}{\exists \mathit{child}.\mathit{doctor}}.$$

Skolemization of the world description $\mathcal{A}$ yields $\mathcal{A}' = \{\mathit{child}(\mathit{Tom}, \mathit{Bill}), \mathit{doctor}(\mathit{Bill})\}$, where *Bill* is a new Skolem constant, whereas Skolemization of the consequent of the second default yields a unary Skolem function, say *child-of*. It is easy to see that the corresponding closed default theory has exactly one extension, and that this extension contains the assertional facts that *Tom* has a rich child and a grandchild who is a doctor, i.e., $(\exists \mathit{child}.\mathit{rich\text{-}person})(\mathit{Tom})$, and $(\exists \mathit{child}.\exists \mathit{child}.\mathit{doctor})(\mathit{Tom})$. Intuitively, this comes from the fact that the closed defaults obtained by instantiating our open defaults with the Skolem constant *Bill* are applicable. Without these ground instances, the above facts could not have been deduced by default. To deduce by default that the grandchild of *Tom* is not only a doctor, but also a rich one, the first default has to be instantiated by the term $\mathit{child\text{-}of}(\mathit{Bill})$.

[2] The concept terms occurring in one rule are assumed to have identical free variables.

3 Problems Caused by Skolemization

In addition to the problem that Skolemization usually destroys the nice compositional character of our concept formulae, it is also problematic for more severe reasons to be presented below. We shall give three examples which demonstrate that Reiter's treatment of open defaults is problematic, from an intuitive as well as a formal point of view.

Our *first example* shows that the Skolemization of the world description may lead to counterintuitive consequences of the default theory. Consider the following concept term which can be used to express that an adult man is married to a woman or is a bachelor

$$(\exists spouse.Woman) \sqcup Bachelor.$$

We assume that our ABox asserts that the individual *Tom* belongs to this concept term, and that he is married to the woman *Mary*. In addition, we take the following default (without prerequisite)

$$\frac{: \neg Woman}{\neg Woman},$$

which corresponds to a still-prevailing male chauvinism in linguistic usage. In order to know with what individuals this default has to be instantiated, we have to Skolemize our ABox facts. Translated into traditional first-order syntax, these facts yield the world description

$$\begin{gathered}\{(\exists y\colon spouse(Tom, y) \wedge Woman(y)) \vee Bachelor(Tom),\\ spouse(Tom, Mary),\\ Woman(Mary)\}.\end{gathered}$$

The Skolemized version of the first formula is

$$(*) \quad (spouse(Tom, Gordy) \wedge Woman(Gordy)) \vee Bachelor(Tom),$$

where *Gordy* is introduced as a new Skolem constant. Because of the disjunction in this formula, our Skolemized world description does not imply $Woman(Gordy)$. Thus the chauvinistic default can fire, and we get $\neg Woman(Gordy)$. Together with the formula $(*)$ this yields $Bachelor(Tom)$ as a consequence of our default theory, which is rather surprising since our ABox actually contains a female spouse of *Tom*.

As already pointed out by Poole, the reason for this strange behaviour comes from that fact that "we have lost the context of what the Skolem constants represent" ([23], p.907), in our case the context that *Gordy* was

originally introduced to stand for a female spouse of *Tom*. Poole proposes to keep track of this context by using Hilbert's ϵ-symbol.

Although Poole's approach may avoid the problem in the above example, it is of no avail in our next examples. These examples demonstrates that, due to the problems caused by Skolemization, the consequences of a default theory depend on the *syntactic* form of the world description, i.e., for identical sets of open defaults, logically equivalent world descriptions may lead to different results.

In our *second example* we consider concept terms $C_1 := \exists R.(A \sqcap B)$ and $C_2 := \exists R.A$ where R is a role name and A, B are concept names. Obviously, if we assert that an individual a is in the first term this implies that it is in the second one as well. For this reason, the ABoxes $\mathcal{A}_1 := \{C_1(a)\}$ and $\mathcal{A}_2 := \{C_1(a), C_2(a)\}$ are logically equivalent. When Skolemizing the first ABox, we get a single new Skolem constant b which is R-related to a and lies in $A \sqcap B$, whereas when Skolemizing the second ABox we get two Skolem constants c and d, both R-related to a, but where c lies in $A \sqcap B$ and d lies in A. Now consider the (open) default $A : \neg B / \neg B$. For the Skolemized version of $\mathcal{A}_1$, this default is instantiated with a, b, whereas for the Skolemized version of $\mathcal{A}_2$ it is instantiated with a, c, d. Obviously, the default rule *cannot* fire for b and c, because their being in $A \sqcap B$ is inconsistent with its justification. On the other hand, this default rule *can* be applied to d, because being in A is consistent with being in $\neg B$. For this reason, d is put into $\neg B$, which shows that the Skolemized version of $\mathcal{A}_2$ has $(\exists R.\neg B)(a)$ as a default consequence, whereas this fact cannot be deduced by default from the Skolemized version of $\mathcal{A}_1$. Technically, the reason for this behaviour is due to the fact that, before the application of the default, the individuals c and d might be identical (which is the reason why the two ABoxes are logically equivalent) whereas this is no longer possible after the default has been applied.

The *third example* is similar to the second. It is quite obvious that the concept terms $\exists R.(A \sqcup B)$ and $(\exists R.A) \sqcup (\exists R.B)$ are equivalent. Let $\mathcal{A}_1$ be an ABox where a is asserted to be in the first concept term, and $\mathcal{A}_2$ one where a is asserted to be in the second concept term. When using a standard Skolemization method, the first ABox yields one new Skolem constant, and the second ABox yields two. Now it is easy to see that the corresponding instantiations of the default rule $A \sqcup B : C / C$ can only fire for the Skolemized version of the first ABox. Consequently, we have a in $\exists R.C$ as a default consequence of the first ABox, but not of the second one, even though these two ABoxes are equivalent.

Lifschitz [15] proposes a treatment of open defaults which avoids Skolemization by working with classes of models instead of sets of formulae in the definition of default extensions. Obviously, working with models means that

logically equivalent formulae must yield the same results. This shows that Lifschitz's approach can overcome the problem pointed out in the previous two examples, even though it was not motivated by the problems connected with Skolemization (see footnote 1 in [15]: "Skolemization ... is irrelevant for this discussion.") Lifschitz's motivation was to make it possible to derive by default universally quantified formulae of the form $\forall x\colon C(x)$, which is not possible with Reiter's approach, but which is not necessary in our context (because the terminological inference service is only meant to derive new ABox facts, i.e., formulae of the form $C(a)$). From our point of view, the main problem of Lifschitz's approach is that working with models means that it becomes even harder to get algorithms for computing extensions. Another problem of his approach is that one gets rather unexpected consequences, due to the fact that models of different cardinality are treated separately. For example, assume that one has formulae ≥ 3 and ≤ 2 expressing that a model has at least 3 and at most 2 elements, respectively, which would, for example, be available in concept languages allowing for number-restrictions and a universal role, i.e., a role U that satisfies $\forall x, y\colon U(x, y)$. The default theory consisting of an empty world description and the closed defaults

$$\frac{\leq 2 :}{C(a)} \quad \text{and} \quad \frac{\geq 3 :}{C(a)}$$

has $C(a)$ as consequence, which means that this approach makes a case analysis with respect to the cardinality of models. But for other cases, Lifschitz's approach still does not make case analysis. For example, the theory consisting of an empty world description and the closed defaults

$$\frac{A(a) :}{C(a)} \quad \text{and} \quad \frac{\neg A(a) :}{C(a)}$$

does not have $C(a)$ as a consequence.

4 An Undecidability Result

In addition to the semantic problems caused by Skolemization, we shall now show that, for our base language $\mathcal{ALCF}$, this treatment of open defaults also leads to an undecidable default consequence relation, even though $\mathcal{ALCF}$ is decidable. This is achieved by reducing the word problem for semigroups [24] to the consequence problem of a default theory.

Let Σ be a finite alphabet, and let $R = \{(u_1, v_1), \ldots, (u_n, v_n)\}$ be a finite set of relations presenting a semigroup over Σ. In the following we shall treat the elements of Σ as attribute names. The semigroup presentation is used

to define a finite set of open defaults as follows. For any $f \in \Sigma$ and for any relation $(u_i, v_i) \in R$ we have defaults

$$\frac{A :}{\forall f.A} \quad \text{and} \quad \frac{A :}{u_i \doteq v_i}.$$

If we want to decide whether the words u, v are equivalent with respect to R, we take the ABox $\mathcal{A}_{u,v} := \{A(a), (u \doteq u)(a), (v \doteq v)(a)\}$ as our world description.

Proposition 4.1 *With respect to the set of defaults induced by Σ and R, the ABox fact $(u \doteq v)(a)$ is a default consequence of $\mathcal{A}_{u,v}$ iff u and v are equivalent with respect to R.*

Intuitively, the world description puts a into A, and asserts sequences of attributes u, v starting from a. The implicit individuals lying on these sequences are made explicit by Skolemization. The first type of defaults puts all individuals reachable from a by a sequence of attributes into A, and the second type identifies individuals which can be reached by the respective sequences u_i and v_i from an individual in A, thus simulating application of relations from R. (It should be noted that the consequents of this second type of defaults are also responsible for the introduction of new implicit individuals.)

Since a formal proof of the proposition is straightforward but rather tedious, we shall just illustrate it by an example. Consider the semigroup presentation $R = \{(fg, gf)\}$ over the alphabet $\Sigma = \{f, g\}$. This presentation is transformed into the default rules

$$\frac{A :}{\forall f.A}, \quad \frac{A :}{\forall g.A}, \quad \text{and} \quad \frac{A :}{fg \doteq gf}.$$

Obviously, the words fgg and ggf are equivalent with respect to R. If we want to obtain this equivalence as a consequence of applying the above default rules, we take the Abox $\mathcal{A}_{fgg,ggf} = \{A(a), (fgg \doteq fgg)(a), (ggf \doteq ggf)(a)\}$ as our world description.

Translated into first-order logic and then Skolemized, this ABox yields the world description

$$\begin{array}{l}\{\ A(a),\\ \quad f(a, b_1) \wedge g(b_1, b_2) \wedge g(b_2, b_3),\\ \quad g(a, c_1) \wedge g(c_1, c_2) \wedge f(c_2, c_3),\\ \quad \forall x, y, z\colon (f(x, y) \wedge f(x, z) \rightarrow y = z),\\ \quad \forall x, y, z\colon (g(x, y) \wedge g(x, z) \rightarrow y = z)\ \},\end{array}$$

where the last two formulae are expressing that f, g are interpreted as partial functions, and $b_1, \ldots, c_3$ are Skolem constants. Note that these formulae have already been used to simplify the rest of the ABox, and that redundant equalities have been removed. We want to show that $b_3 = c_3$ is a consequence of the default theory.

The translated and Skolemized form of the consequent $fg \doteq gf$ of the third default is $f(x, h_1(x)) \wedge g(h_1(x), h_2(x)) \wedge g(x, k_1(x)) \wedge f(k_1(x), k_2(x)) \wedge h_2(x) = k_2(x)$, where h_1, h_2, k_1, k_2 are unary Skolem functions.

Since $A(a)$ is in our world description, the third default, instantiated by a, is applicable, and yields $f(a, h_1(a)) \wedge g(h_1(a), h_2(a)) \wedge g(a, k_1(a)) \wedge f(k_1(a), k_2(a)) \wedge h_2(a) = k_2(a)$. The formulae which express that f, g are partial functions yield $h_1(a) = b_1$, $h_2(a) = b_2$, and $k_1(a) = c_1$.

Applying the second default, instantiated by a, we get $\forall y$: $(g(a, y) \rightarrow A(y))$, which in turn yields $A(c_1)$. Now we can apply the third default, instantiated by c_1, which yields $f(c_1, h_1(c_1)) \wedge g(h_1(c_1), h_2(c_1)) \wedge g(c_1, k_1(c_1)) \wedge f(k_1(c_1), k_2(c_1)) \wedge h_2(c_1) = k_2(c_1)$. Because of the formulae expressing that f, g are partial functions we get $c_2 = k_1(c_1), c_3 = k_2(c_1)$, and, using the additional fact $k_1(a) = c_1$, also $k_2(a) = h_1(c_1)$.

To sum up we have $b_2 = h_2(a) = k_2(a) = h_1(c_1)$, $c_3 = k_2(c_1) = h_2(c_1)$, and $g(b_2, b_3)$ as well as $g(h_1(c_1), h_2(c_1))$. This yields $b_3 = h_2(c_1) = c_3$, which is what we wanted to show.

Since the word problem for semigroups is in general undecidable, the proposition shows that our terminological default theories in general have an undecidable consequence problem.

Corollary 4.2 *The consequence problem for an open default theory is in general undecidable, even if one has a finite set of defaults and the base language is decidable.*

It should be noted that the default rules used in the reduction are monotonic (i.e., they do not have justifications). Consequently, the default theory has exactly one extension, which shows that the undecidability result is independent of whether one wants to employ skeptical or credulous reasoning. In addition, this shows that the consequences of rule applications in the CLASSIC system would become undecidable, if CLASSIC applied rules not only to individuals explicitly present in the ABox, but also to implicit individuals. This result for CLASSIC rules has already been mentioned by Nebel and Smolka [21], but without proof. In the next section we shall see that the restriction to explicit individuals leads to a decidable consequence relation even if one allows nonmonotonic default rules instead of CLASSIC's monotonic rules.

5 Computing Extensions

Because of the problems caused by Skolemization in Reiter's treatment of open defaults, we now propose a *restricted semantics for open default theories:* default rules are only applied to individuals that are explicitly mentioned in the ABox.

Definition 5.1 *In the restricted semantics for terminological default theories, an open default of a terminological default theory* $(\mathcal{A}, \mathcal{D})$ *is interpreted as representing the closed defaults obtained by instantiating the free variable by all individual names occurring in* $\mathcal{A}$*.*

Because the ABox $\mathcal{A}$ and the set of open defaults $\mathcal{D}$ are assumed to be finite, we end up with a *finite set of closed defaults.* Since our terminological language is decidable, the methods of Junker and Konolige, or of Schwind and Risch can be applied to compute all extensions (according to our restricted semantics).

In principle, both methods depend on the fact that any extension of a closed default theory $(\mathcal{A}, \mathcal{D})$ is of the form $Th(\mathcal{A} \cup Con(\widehat{\mathcal{D}}))$ for a subset $\widehat{\mathcal{D}}$ of $\mathcal{D}$. If $\mathcal{D}$ is finite, there are only finitely many such subsets, and the only problem is to decide which of these generate an extension. In fact, if the base language is decidable, one could even use for this purpose the iteration process described in the definition of an extension. This is so because decidability of the base language makes each iteration step effective, and the iteration process terminates because there are only finitely many consequents to be added. However, with this method one has to consider all the (exponentially many) subsets of $\mathcal{D}$. The two methods which we shall describe below try to avoid considering all subsets, thus making the search for (the sets of generating defaults of) all extensions more efficient.

5.1 Junker and Konolige's method

Junker and Konolige [12] translate a closed default theory $(\mathcal{A}, \mathcal{D})$ into a Truth Maintenance Network (TMN) à la Doyle [8]. The nodes of the TMN are the consequents $\mathcal{C}_{\mathcal{D}}$, and the prerequisites and negated justifications $\mathcal{L}_{\mathcal{D}}$ of the defaults. A default $\alpha : \beta_1, \ldots, \beta_n / \gamma$ of $\mathcal{D}$ is translated into a *nonmonotonic justification* $\langle in(\alpha), out(\neg\beta_1, \ldots, \neg\beta_n) \rightarrow \gamma \rangle$ of the TMN. In order to supply the truth maintenance system with enough information about first-order derivability in the base language, each prerequisite and negated justification of a default gives rise to several *monotonic justifications* of the TMN. These justifications are of the form $\langle in(Q) \rightarrow q \rangle$ where $q \in \mathcal{L}_{\mathcal{D}}$, and Q is a minimal

subset of $\mathcal{C}_\mathcal{D}$ such that $\mathcal{A} \cup Q$ entails q—i.e., $\mathcal{A} \cup Q \models q$ but $\mathcal{A} \cup Q' \not\models q$ for every proper subset Q' of Q.

Junker and Konolige show that there is a 1–1-correspondence between admissible labellings of the TMN thus obtained and extensions of the default theory, and they describe an algorithm which computes all admissible labellings of a TMN. Given such an admissible labelling, the set of generating defaults of the corresponding extension consists of the defaults whose consequents are labelled "in."

In order to make the translation of terminological default theories into TMNs effective, one has to show how to compute the above mentioned monotonic justifications of the TMN. First note that the elements of $\mathcal{L}_\mathcal{D} \cup \mathcal{C}_\mathcal{D}$ are admissible assertional facts. This is obvious for the prerequisites and the consequents of our instantiated defaults, and for the negated justifications it follows from the fact that the concept language has negation as an operator. For this reason, $\mathcal{A} \cup Q$ for a subset Q of $\mathcal{C}_\mathcal{D}$ is an admissible ABox of our language, and the entailment problem $\mathcal{A} \cup Q \models q$ for $q \in \mathcal{L}_\mathcal{D}$ is an ordinary instantiation problem. As mentioned in Section 2, the instantiation problem is decidable for our language. A *brute force algorithm* could just compute all subsets Q of $\mathcal{C}_\mathcal{D}$ such that $\mathcal{A} \cup Q$ entails $q \in \mathcal{L}_\mathcal{D}$, and then, for each q, eliminate the ones which are not minimal. Of course, this simple algorithm is very inefficient, and thus not appropriate for actual implementations.

Because $\mathcal{A} \cup Q$ entails an assertional fact $C(a)$ iff $\mathcal{A} \cup Q \cup \{\neg C(a)\}$ is inconsistent, we need a solution of the following problem: Let $\mathcal{A}$, $\mathcal{B}$ be ABoxes. Find all minimal subsets $\mathcal{Q}$ of $\mathcal{B}$ such that $\mathcal{A} \cup \mathcal{Q}$ is inconsistent. Since a similar algorithmic problem has to be solved for the method obtained from Schwind and Risch's characterization of an extension, we defer the description of a more efficient solution of this problem to a separate section.

A characteristic feature of Junker and Konolige's method is that—after the computation of the minimal sets Q—it is completely abstracted from derivability in the base language. This may be advantageous from a conceptual point of view, but it can be problematic from the algorithmic point of view. In fact, one has to compute the corresponding minimal sets for all elements q in $\mathcal{L}_\mathcal{D}$, even though this information may not contribute to the computation of an extension.

5.2 A method based on a theorem by Schwind and Risch

Schwind and Risch [28] give a theorem which characterizes those subsets $\widehat{\mathcal{D}}$ of $\mathcal{D}$ which are sets of generating defaults of an extension of a closed default theory $(\mathcal{W}, \mathcal{D})$. They use this characterization for computing extensions of

propositional default theories. In this subsection, we shall show how to apply the theorem to computing extensions of terminological default theories.

Before we can formulate the theorem we need one more piece of notation.

Definition 5.2 *Let $\mathcal{W}$ be a set of closed formulae, and $\mathcal{D}$ be a set of closed defaults. We define $\mathcal{D}_0 = \emptyset$ and, for $i \geq 0$,*

$$\mathcal{D}_{i+1} = \mathcal{D}_i \cup \{d = \frac{\alpha : \beta_1, \ldots, \beta_n}{\gamma} \mid d \in \mathcal{D} \text{ and } \mathcal{W} \cup Con(\mathcal{D}_i) \models \alpha\}.$$

Then $\mathcal{D}$ is called grounded *in $\mathcal{W}$ iff $\mathcal{D} = \bigcup_{i=0}^{\infty} \mathcal{D}_i$.*

This definition of groundedness differs from the one given in [28], but it is easy to see that both formulations are equivalent. The advantage of our formulation is that it can directly be used as a procedure for deciding groundedness, if $\mathcal{D}$ is finite and the entailment problem in the base language is decidable. If $\mathcal{D}$ is not grounded in $\mathcal{W}$, then $\bigcup_{i=0}^{\infty} \mathcal{D}_i$ is the largest subset of $\mathcal{D}$ that is grounded in $\mathcal{W}$.

The iteration process described above corresponds to the iteration in the definition of extensions, with the main difference that it disregards the justifications. The second condition given in the following theorem makes up for this neglect.

Theorem 5.3 (Schwind and Risch) *Let $(\mathcal{W}, \mathcal{D})$ be a closed default theory. A subset $\widehat{\mathcal{D}}$ of $\mathcal{D}$ is a set of generating defaults of an extension of $(\mathcal{W}, \mathcal{D})$ iff the following two conditions hold:*

1. *$\widehat{\mathcal{D}}$ is grounded in $\mathcal{W}$.*
2. *For all $d \in \mathcal{D}$ with $d = \alpha : \beta_1, \ldots, \beta_n / \gamma$ we have $d \in \widehat{\mathcal{D}}$ iff $\mathcal{W} \cup Con(\widehat{\mathcal{D}}) \models \alpha$ and for all $i, 1 \leq i \leq n$, $\mathcal{W} \cup Con(\widehat{\mathcal{D}}) \not\models \neg\beta_i$.*

If $\mathcal{D}$ is finite, and the entailment problem in the base language is decidable, this theorem provides us with an effective test of whether a subset $\widehat{\mathcal{D}}$ of $\mathcal{D}$ is a set of generating defaults of an extension of $(\mathcal{W}, \mathcal{D})$. We shall now describe a method based on this theorem which allows us to compute (the sets of generating defaults of) all extensions without having to consider all subsets of $\mathcal{D}$.

If $\mathcal{W}$ is inconsistent then there is only one extension, namely the set of all formulae. In the following, we shall without loss of generality assume that $\mathcal{W}$ is consistent. Now, let $\mathcal{D}_0$ be the largest subset of $\mathcal{D}$ that is grounded in $\mathcal{W}$, and let $\mathcal{D}_1, \ldots, \mathcal{D}_m$ be all maximal subsets of $\mathcal{D}_0$ such that $\mathcal{W} \cup Con(\mathcal{D}_i)$ is consistent. Since $\mathcal{W}$ is assumed to be consistent, extensions are consistent

Compute-All-Extensions$(\mathcal{W}, \mathcal{D})$
begin
(1) if $\mathcal{W}$ is inconsistent
(2) then print "Inconsistent world description"
(3) else for all maximal subsets $\mathcal{D}'$ of $\mathcal{D}_0$ such that
$\mathcal{W} \cup Con(\mathcal{D}')$ is consistent
(4) do *Remove-Defaults*$(\mathcal{W}, \mathcal{D}, \mathcal{D}')$;
end

Remove-Defaults$(\mathcal{W}, \mathcal{D}, \mathcal{D}')$
begin
(1) let $\mathcal{D}_0$ be the largest subset of $\mathcal{D}'$ that is grounded in $\mathcal{W}$;
(2) if $\mathcal{W} \cup Con(\mathcal{D}_0) \models \neg\beta_i$ for some justification $\beta_i \in Jus(\mathcal{D}_0)$
(3) then let $d = \alpha : \beta_1, \ldots, \beta_n/\gamma$ be the corresponding default;
(4) *Remove-Defaults*$(\mathcal{W}, \mathcal{D}, \mathcal{D}_0 \setminus \{d\})$;
(5) for all maximal subsets $\mathcal{D}''$ of $\mathcal{D}_0$ such that
$d \in \mathcal{D}''$ and $\mathcal{W} \cup Con(\mathcal{D}'') \not\models \neg\beta_i$
(6) do *Remove-Defaults*$(\mathcal{W}, \mathcal{D}, \mathcal{D}'')$;
(7) else if for each $\alpha : \beta_1, \ldots, \beta_n/\gamma \in \mathcal{D} \setminus \mathcal{D}_0$ either $\mathcal{W} \cup Con(\mathcal{D}_0) \not\models \alpha$
(8) or $\mathcal{W} \cup Con(\mathcal{D}_0) \models \neg\beta_i$ for some i
(9) then add $\mathcal{D}_0$ to the list of sets of generating defaults;
end

Figure 1: Procedure for computing the sets of generating defaults of all extensions of the closed default theory $(\mathcal{W}, \mathcal{D})$. *Proviso:* $\mathcal{D}$ is finite and entailment in the base language is decidable.

as well, which means that a generating set of defaults of an extension is a subset of one of the $\mathcal{D}_i$. The idea underlying our method is to start with these maximal sets $\mathcal{D}_i$, and successively eliminate defaults violating the first condition of the theorem, or the "only if" part of the second condition. If no more defaults can be eliminated, the "if" part of the second condition is tested.

Figure 1 describes the procedure for computing all extensions of a closed default theory. To show soundness and completeness of the procedure (Theorem 5.7) we need three lemmas.

Lemma 5.4 *Let $(\mathcal{W}, \mathcal{D})$ be a closed default theory and let $\mathcal{D}' \subseteq \mathcal{D}$ be such that $\mathcal{W} \cup Con(\mathcal{D}')$ is consistent. Suppose the call* Remove-Defaults$(\mathcal{W}, \mathcal{D}, \mathcal{D}')$ *returns the list $\mathcal{L}$ of sets of defaults. If $\mathcal{D}_0 \in \mathcal{L}$ then $\mathcal{D}_0$ is a set of generating*

defaults for an extension of $(\mathcal{W}, \mathcal{D})$.

Proof. We prove this lemma by showing that a set $\mathcal{D}_0$ of defaults contained in $\mathcal{L}$ satisfies Conditions *1* and *2* of Theorem 5.3.

Suppose that $\mathcal{D}_0$ is contained in $\mathcal{L}$. It is easy to see that $\mathcal{D}_0$ is a subset of $\mathcal{D}'$ that is grounded in $\mathcal{W}$ (because of line (1)), which shows that Condition *1* of Theorem 5.3 holds for $\mathcal{D}_0$.

To show that $\mathcal{D}_0$ satisfies the second condition of Theorem 5.3, first assume that $d = \alpha : \beta_1, \ldots, \beta_n / \gamma \in \mathcal{D}_0$. Recall that $\mathcal{D}_0$ is grounded in $\mathcal{W}$, which implies that $\mathcal{W} \cup \mathit{Con}(\mathcal{D}_0) \models \alpha$. Furthermore, observe that, for all i, $1 \leq i \leq n$, $\mathcal{W} \cup \mathit{Con}(\mathcal{D}_0) \not\models \neg\beta_i$ (because the condition in line (2) does not hold for $\mathcal{D}_0$). Both facts together show that the "only if" part of Condition *2* holds.

Now assume that $d = \alpha : \beta_1, \ldots, \beta_n / \gamma \in \mathcal{D} \setminus \mathcal{D}_0$. Then either $\mathcal{W} \cup \mathit{Con}(\mathcal{D}_0) \not\models \alpha$ or $\mathcal{W} \cup \mathit{Con}(\mathcal{D}_0) \models \neg\beta_i$ for some i (because the condition in lines (7) and (8) holds for $\mathcal{D}_0$). This shows that the "if" part of Condition *2* is also satisfied. □

Lemma 5.5 *Let $\mathcal{D}_0$ be a set of generating defaults for an extension of a closed default theory $(\mathcal{W}, \mathcal{D})$, and let $\mathcal{D}'$ be a subset of $\mathcal{D}$ such that $\mathcal{D}_0 \subseteq \mathcal{D}'$ and $\mathcal{W} \cup \mathit{Con}(\mathcal{D}')$ is consistent. If Remove-Defaults$(\mathcal{W}, \mathcal{D}, \mathcal{D}')$ recursively calls Remove-Defaults then there is a call with arguments $\mathcal{W}, \mathcal{D}, \mathcal{D}''$ where $\mathcal{D}_0 \subseteq \mathcal{D}'' \subset \mathcal{D}'$.*

Proof. Let $\mathcal{D}_0 \subseteq \mathcal{D}'$ be sets of defaults satisfying the assumptions of the lemma. Suppose *Remove-Defaults* is called with arguments $\mathcal{W}, \mathcal{D}, \mathcal{D}'$. Let $\mathcal{D}'_0$ be the largest subset of $\mathcal{D}'$ that is grounded in $\mathcal{W}$. Then $\mathcal{D}_0 \subseteq \mathcal{D}'_0$ because every set of generating defaults for an extension of $(\mathcal{W}, \mathcal{D})$ is grounded in $\mathcal{W}$.

If the condition in line (2) does not hold for $\mathcal{D}'_0$, *Remove-Defaults* is obviously not called recursively, and nothing has to be shown. Thus assume that the condition in line (2) holds for $\mathcal{D}'_0$. This means that there is a default $d = \alpha : \beta_1, \ldots, \beta_n / \gamma \in \mathcal{D}'_0$ such that $\mathcal{W} \cup \mathit{Con}(\mathcal{D}'_0) \models \neg\beta_i$ for some i, $1 \leq i \leq n$.

If $d \notin \mathcal{D}_0$ we have $\mathcal{D}_0 \subseteq \mathcal{D}'_0 \setminus \{d\} \subset \mathcal{D}'_0$, and the call of *Remove-Defaults* with arguments $\mathcal{W}, \mathcal{D}, \mathcal{D}'_0 \setminus \{d\}$) (cf. line (4)) satisfies the required property. Now assume that $d \in \mathcal{D}_0$. Since $\mathcal{D}_0$ is a set of generating defaults for an extension we know that $\mathcal{W} \cup \mathit{Con}(\mathcal{D}_0) \not\models \neg\beta_i$. Thus there is a maximal subset $\mathcal{D}''$ of $\mathcal{D}'_0$ with $\mathcal{W} \cup \mathit{Con}(\mathcal{D}'') \not\models \neg\beta_i$ that contains $\mathcal{D}_0$, and this means that the call *Remove-Defaults*$(\mathcal{W}, \mathcal{D}, \mathcal{D}'')$ has the required property (cf. line (5) and (6)). □

Lemma 5.6 *Let $\mathcal{D}_0$ be a set of generating defaults for an extension of a closed default theory $(\mathcal{W}, \mathcal{D})$, and let $\mathcal{D}'$ be a subset of $\mathcal{D}$ such that $\mathcal{D}_0 \subseteq \mathcal{D}'$ and $\mathcal{W} \cup Con(\mathcal{D}')$ is consistent. Suppose* Remove-Defaults *is called with arguments $\mathcal{W}, \mathcal{D}, \mathcal{D}'$. Then*

- *there is a recursive call of* Remove-Defaults*, or*
- *$\mathcal{D}_0$ is added to the list of sets of generating defaults.*

Proof. Let $\mathcal{D}_0 \subseteq \mathcal{D}'$ be sets of defaults satisfying the assumptions of the lemma. Suppose the call *Remove-Defaults*$(\mathcal{W}, \mathcal{D}, \mathcal{D}')$ does not recursively call *Remove-Defaults*. This means that the condition in line (2) does not hold for $\mathcal{D}'_0$, where $\mathcal{D}'_0$ is the largest subset of $\mathcal{D}'$ that is grounded in $\mathcal{W}$. We show that $\mathcal{D}'_0 = \mathcal{D}_0$.

Since $\mathcal{D}_0$ is grounded in $\mathcal{W}$, we get $\mathcal{D}_0 \subseteq \mathcal{D}'_0$, and thus we only have to show $\mathcal{D}'_0 \subseteq \mathcal{D}_0$. Assume to the contrary that $\mathcal{D}'_0 \setminus \mathcal{D}_0 \neq \emptyset$. First we show that $\mathcal{W} \cup Con(\mathcal{D}_0) \models \alpha$ for some default $\alpha : \beta_1, \ldots, \beta_n/\gamma \in \mathcal{D}'_0 \setminus \mathcal{D}_0$. To see this, recall that $\mathcal{D}'_0$ is grounded in $\mathcal{W}$. This means that there is a sequence $d'_1, d'_2, \ldots$ of default in $\mathcal{D}'_0$ such that $\mathcal{W} \cup Con(\{d'_1, \ldots, d'_{k-1}\}) \models \alpha'_k$ where α'_k is the prerequisite of the k-th default. Let l be the smallest number such that $d'_l \in \mathcal{D}'_0 \setminus \mathcal{D}_0$. Thus $d'_j \in \mathcal{D}_0$ for all j, $1 \leq j < l$, which shows that $\mathcal{W} \cup Con(\mathcal{D}_0) \models \alpha'_l$.

Second, we have $\mathcal{W} \cup Con(\mathcal{D}'_0) \not\models \neg\beta_i$ for all justifications $\beta_i \in Jus(\mathcal{D}'_0)$ because the condition in line (2) does not hold for $\mathcal{D}'_0$. Since $\mathcal{D}_0 \subseteq \mathcal{D}'_0$ we especially know that $\mathcal{W} \cup Con(\mathcal{D}_0) \not\models \neg\beta_i$ for all justifications $\beta_i \in Jus(\mathcal{D}_0)$.

Thus, we have shown that there is some default $d \in \mathcal{D}'_0 \setminus \mathcal{D}_0$, $d = \alpha : \beta_1, \ldots, \beta_n/\gamma$, such that $\mathcal{W} \cup Con(\mathcal{D}_0) \models \alpha$ and $\mathcal{W} \cup Con(\mathcal{D}_0) \not\models \neg\beta_i$ for all i, $1 \leq i \leq n$. Because of Theorem 5.3 this is a contradiction with our assumption that $\mathcal{D}_0$ is a set of generating defaults. Therefore the assumption $\mathcal{D}'_0 \setminus \mathcal{D}_0 \neq \emptyset$ is falsified, and we can conclude that $\mathcal{D}'_0 = \mathcal{D}_0$.

Since $\mathcal{D}_0$ is a set of generating defaults, the condition in lines (7), (8) holds for $\mathcal{D}_0$ (cf. Condition *2* of Theorem 5.3). Thus $\mathcal{D}_0$ is added to the list of sets of generating defaults. □

Now we are ready to prove soundness and completeness of our algorithm. First we observe that every set of defaults computed by the algorithm is in fact a set of generating defaults for an extension of a closed default theory $(\mathcal{W}, \mathcal{D})$ (cf. Lemma 5.4).

Now assume that $\mathcal{D}_0$ is a set of generating defaults for an extension of $(\mathcal{W}, \mathcal{D})$. Recall that $\mathcal{W} \cup Con(\mathcal{D}_0)$ is consistent. Thus there is a maximal subset $\mathcal{D}'$ of $\mathcal{D}$ such that $\mathcal{W} \cup Con(\mathcal{D}')$ is consistent and $\mathcal{D}'$ contains $\mathcal{D}_0$. This shows that *Compute-All-Extensions*$(\mathcal{W}, \mathcal{D})$ generates a call *Remove-Defaults*

with arguments $\mathcal{W}, \mathcal{D}, \mathcal{D}'$ (cf. lines (3) and (4) in the function *Compute-All-Extensions*) for some subset $\mathcal{D}'$ of $\mathcal{D}$ with $\mathcal{D}_0 \subseteq \mathcal{D}'$.

If the call *Remove-Defaults*$(\mathcal{W}, \mathcal{D}, \mathcal{D}')$ returns the list $\mathcal{L}$ of sets of defaults then $\mathcal{D}_0$ is contained in $\mathcal{L}$. This result is an immediate consequence of the previous two lemmas. In fact, Lemma 5.5 shows that there is a sequence of calls of *Remove-Defaults* such that $\mathcal{W}$, $\mathcal{D}$, $\mathcal{C}_i$ are the arguments of the i-th call where $\mathcal{C}_1 = \mathcal{D}'$, $\mathcal{C}_{i+1} \subset \mathcal{C}_i$, and $\mathcal{D}_0 \subseteq \mathcal{C}_i$ for all i. Since $\mathcal{D}$ is assumed to be finite and the $\mathcal{C}_i$'s are decreasing, there is some $m \geq i$ such that *Remove-Defaults*$(\mathcal{W}, \mathcal{D}, \mathcal{C}_m)$ does not generate a recursive call of *Remove-Defaults*. In this case $\mathcal{D}_0$ is added to the list $\mathcal{L}$ of sets of defaults (Lemma 5.6).

Theorem 5.7 *The call of the procedure Compute-All-Extensions with input* $(\mathcal{W}, \mathcal{D})$ *computes sets of generating defaults for all extensions of the closed default theory* $(\mathcal{W}, \mathcal{D})$.

The functions *Compute-All-Extensions* and *Remove-Defaults* use the following subprocedures which have not explicitly been described:

- Decide whether $\mathcal{W}$ is consistent.
- Compute all maximal subsets $\mathcal{D}'$ of $\mathcal{D}$ such that $\mathcal{W} \cup \mathit{Con}(\mathcal{D}')$ is consistent.
- Compute the largest subset $\mathcal{D}_0$ of $\mathcal{D}'$ that is grounded in $\mathcal{W}$.
- Compute all maximal subsets $\mathcal{D}''$ of $\mathcal{D}_0$ such that $\mathcal{W} \cup \mathit{Con}(\mathcal{D}'') \not\models \neg\beta_i$.

The first subprocedure is a direct application of the decision algorithm for entailment in the base language. The third subprocedure is simply obtained by implementing the definition of groundedness.

The other two procedures depend on an algorithm for the following problem, which will be considered in the next section: Let $\mathcal{A}, \mathcal{B}$ be ABoxes. Compute all maximal subsets $\mathcal{Q}$ of $\mathcal{B}$ such that $\mathcal{A} \cup \mathcal{Q}$ is consistent.

In fact, the second subprocedure is a direct application of such an algorithm. For the fourth subprocedure, note that $\mathcal{W} \cup \mathit{Con}(\mathcal{D}'') \not\models \neg\beta_i$ iff $\mathcal{W} \cup \mathit{Con}(\mathcal{D}'') \cup \{\beta_i\}$ is consistent.

6 Computing Minimal Inconsistent and Maximal Consistent ABoxes

This section is concerned with the following algorithmic problems: Given two ABoxes $\mathcal{A}, \mathcal{B}$, find all minimal (resp. maximal) subsets $\mathcal{Q}$ of $\mathcal{B}$ such that $\mathcal{A} \cup \mathcal{Q}$ is inconsistent (resp. consistent).

Since consistency of ABoxes in $\mathcal{ALCF}$ is decidable, there is the obvious "brute-force" solution which tests consistency of $\mathcal{A} \cup \mathcal{Q}$ for all subsets $\mathcal{Q}$ of $\mathcal{B}$, and then takes the minimal inconsistent (maximal consistent) ones. In the following we shall describe a more efficient method of finding these minimal (maximal) sets. The method is an extension of the tableaux-based consistency algorithms for ABoxes described in [1, 10]. The idea of employing tableaux-based methods for such purposes was already used in [18, 28], but these papers restricted themselves to propositional logic, which is a much easier case.

In order to decide whether an ABox $\mathcal{A}$ is consistent, the tableaux-based consistency algorithm tries to generate a finite model of $\mathcal{A}$. In principle, it starts with $\mathcal{A}$, and adds new assertional facts with the help of certain rules until the obtained ABox is "complete," i.e., one can apply no more rules. Because of the presence of disjunction in our language, a given ABox must sometimes be transformed into two different new ABoxes, with the intended meaning that the original ABox is consistent iff one of the new ABoxes is consistent. Formally, this means that one is working with sets of ABoxes instead of a single ABox.

For ease of presentation, we restrict ourselves in this formal description to the terminological language $\mathcal{ALC}$ where we do not have attributes and agreements. Later on, we shall point out how the algorithm can be extended to $\mathcal{ALCF}$.

Figure 2 describes the transformation rules of the tableaux-based consistency algorithm for $\mathcal{ALC}$. Without loss of generality we assume that the concept terms occurring in $\mathcal{A}_0$ are in negation normal form, i.e., negation occurs only directly in front of concept names. Negation normal forms can be generated using the fact that the following pairs of concept terms are equivalent: $\neg\neg C$ and C, $\neg(C \sqcap D)$ and $\neg C \sqcup \neg D$, $\neg(C \sqcup D)$ and $\neg C \sqcap \neg D$, $\neg(\exists R.C)$ and $\forall R.\neg C$, as well as $\neg(\forall R.C)$ and $\exists R.\neg C$.

The following facts make clear why the rules of Figure 2 provide us with a decision procedure for consistency of ABoxes of $\mathcal{ALC}$ (see [10, 1] for a proof).

Proposition 6.1

1. *If $\mathcal{A}_1$ is obtained from $\mathcal{A}_0$ by application of the conjunction, exists-restriction, or value-restriction rule then $\mathcal{A}_0$ is consistent iff $\mathcal{A}_1$ is consistent.*
2. *If $\mathcal{A}_1, \mathcal{A}_2$ are obtained from $\mathcal{A}_0$ by application of the disjunction rule then $\mathcal{A}_0$ is consistent iff $\mathcal{A}_1$ or $\mathcal{A}_2$ is consistent.*
3. *A complete ABox, i.e., an ABox to which no more rules apply, is consistent iff it does not contain an obvious contradiction, i.e., facts*

Let $\mathcal{M}$ be a finite set of ABoxes, and let $\mathcal{A}_0$ be an element of $\mathcal{M}$. The following rules replace $\mathcal{A}_0$ by an ABox $\mathcal{A}_1$ or by two ABoxes $\mathcal{A}_1$ and $\mathcal{A}_2$.

The conjunction rule. Assume that $(C \sqcap D)(a)$ is in $\mathcal{A}_0$, and that $\mathcal{A}_0$ does not contain both assertions $C(a)$ and $D(a)$. The ABox $\mathcal{A}_1$ is obtained from $\mathcal{A}_0$ by adding $C(a)$ and $D(a)$.

The disjunction rule. Assume that $(C \sqcup D)(a)$ is in $\mathcal{A}_0$, and that $\mathcal{A}_0$ contains neither $C(a)$ nor $D(a)$. The ABox $\mathcal{A}_1$ is obtained from $\mathcal{A}_0$ by adding $C(a)$, and the ABox $\mathcal{A}_2$ is obtained from $\mathcal{A}_0$ by adding $D(a)$.

The exists-restriction rule. Assume that $(\exists R.C)(a)$ is in $\mathcal{A}_0$, and that $\mathcal{A}_0$ does not contain assertions $R(a, c)$ and $C(c)$ for some individual c. One generates a new individual name b, and obtains $\mathcal{A}_1$ from $\mathcal{A}_0$ by adding $R(a, b)$ and $C(b)$.

The value-restriction rule. Assume that $(\forall R.C)(a)$ and $R(a, b)$ are in $\mathcal{A}_0$, and that $\mathcal{A}_0$ does not contain the assertion $C(b)$. The ABox $\mathcal{A}_1$ is obtained from $\mathcal{A}_0$ by adding $C(b)$.

Figure 2: Transformation rules of the consistency algorithm for $\mathcal{ALC}$.

$A(b), \neg A(b)$ *for an individual name* b *and a concept name* A.

4. *The transformation process always terminates.*

An obvious contradiction of the form $A(b), \neg A(b)$ will also be called "clash" in the following.

To check whether a given ABox $\mathcal{A}$ is consistent one thus starts with $\{\mathcal{A}\}$, and applies transformation rules (in arbitrary order) as long as possible. Eventually, this yields a finite set $\mathcal{M}$ of complete ABoxes with the property that $\mathcal{A}$ is consistent iff one of the ABoxes in $\mathcal{M}$ is consistent. Since the elements of $\mathcal{M}$ are complete their consistency can simply be decided by looking for an obvious contradiction.

Now assume that $\mathcal{A}, \mathcal{B}$ are ABoxes, and we want to find all minimal (resp. maximal) subsets $\mathcal{Q}$ of $\mathcal{B}$ such that $\mathcal{A} \cup \mathcal{Q}$ is inconsistent (resp. consistent). We start with applying the tableaux-based consistency algorithm to $\mathcal{A} \cup \mathcal{B}$. Let $\mathcal{A}_1, \ldots, \mathcal{A}_m$ be the complete ABoxes obtained this way. If one of these is not obviously contradictory, $\mathcal{A} \cup \mathcal{B}$ is consistent, and there are no minimal inconsistent sets to compute (resp. $\mathcal{B}$ is the maximal consistent

set). Otherwise, we want to know which elements of $\mathcal{B}$ can be dispensed with without destroying the property that all complete ABoxes contain an obvious contradiction (resp. which elements of $\mathcal{B}$ have to be removed to get at least one complete ABox without obvious contradiction).

For this reason, it is important to know which facts in $\mathcal{B}$ contribute to a particular obvious contradiction. To this purpose we introduce a propositional variable for each element of $\mathcal{B}$, and label assertional facts with "monotonic" boolean formulae built from these variables, i.e., propositional formulae built from the variables by using conjunction and disjunction only. In the original ABox $\mathcal{A} \cup \mathcal{B}$, the elements of $\mathcal{A}$ are labelled with "true," and the elements of $\mathcal{B}$ are labelled with the corresponding propositional variable. If, during the consistency test, n assertional facts with labels $\phi_1, \ldots, \phi_n$ give rise to a new fact, the new one is labelled by $\phi_1 \wedge \ldots \wedge \phi_n$. Since the same assertional fact may arise in more than one way, we also get disjunctions in labels. Again, we end up with complete ABoxes $\mathcal{A}_1, \ldots, \mathcal{A}_m$, but now all assertional facts occurring in these ABoxes have labels.

More formally, we shall now describe a *labelled consistency algorithm* for ABoxes $\mathcal{A} \cup \mathcal{B}$ consisting of "hard" facts $\mathcal{A}$ and of "refutable" facts $\mathcal{B}$. Without loss of generality we assume that the concept terms occurring in $\mathcal{A} \cup \mathcal{B}$ are in negation normal form. Initially, the elements of $\mathcal{A} \cup \mathcal{B}$ are labelled with monotonic boolean formulae as described above. We shall refer to the label of an assertional fact α by $\mathit{ind}(\alpha)$. Starting with the singleton set $\{\mathcal{A} \cup \mathcal{B}\}$, the transformation rules of Figure 3 are applied as long as possible.

As for the unlabelled consistency algorithm, there cannot be an infinite chain of rule applications. This can, for example, be shown by a straightforward adaptation to the labelled case of the termination ordering used in [1].

Thus the labelled consistency algorithm also terminates with a finite set of complete ABoxes, i.e., labelled ABoxes to which no rules apply. The labels occurring in these ABoxes can be used to describe which of the original facts in $\mathcal{B}$ are responsible for the obvious contradictions.

Definition 6.2 (Clash formula) *Let $\mathcal{A}_1, \ldots, \mathcal{A}_n$ be the complete ABoxes obtained by applying the labelled consistency algorithm to $\mathcal{A} \cup \mathcal{B}$. A particular clash $A(a), \neg A(a) \in \mathcal{A}_i$ is expressed by the propositional formula $\mathit{ind}(A(a)) \wedge \mathit{ind}(\neg A(a))$. Now let $\psi_{i,1}, \ldots, \psi_{i,k_i}$ be the formulae expressing all the clashes in $\mathcal{A}_i$. The clash formula associated with $\mathcal{A} \cup \mathcal{B}$ is*

$$\bigwedge_{i=1}^{n} \bigvee_{j=1}^{k_i} \psi_{i,j}.$$

Let $\mathcal{M}$ be a finite set of labelled ABoxes, and let $\mathcal{A}_0$ be an element of $\mathcal{M}$. The following rules replace $\mathcal{A}_0$ by an ABox $\mathcal{A}_1$ or by two ABoxes $\mathcal{A}_1$ and $\mathcal{A}_2$. These new ABoxes either contain additional assertional facts, or the indices of existing assertional facts are changed. In order to avoid having to distinguish between these two cases in the formulation of the rules, we introduce a new notation. An ABox is *extended by* an assertional fact with index ϕ means the following: If this fact is already present with index ψ, we just change its index to $\psi \vee \phi$. Otherwise, it is added to the ABox and gets index ϕ.

The conjunction rule. Assume that $(C \sqcap D)(a)$ is in $\mathcal{A}_0$, and that $\mathcal{A}_0$ does not contain assertions $C(a)$ and $D(a)$ whose indices are both implied by $ind((C \sqcap D)(a))$. The ABox $\mathcal{A}_1$ is obtained by extending $\mathcal{A}_0$ by $C(a)$ with index $ind((C \sqcap D)(a))$ and by $D(a)$ with index $ind((C \sqcap D)(a))$.

The disjunction rule. Assume that $(C \sqcup D)(a)$ is in $\mathcal{A}_0$, and that $\mathcal{A}_0$ does not contain $C(a)$ or $D(a)$ whose index is implied by $ind((C \sqcup D)(a))$. The ABox $\mathcal{A}_1$ is obtained by extending $\mathcal{A}_0$ by $C(a)$ with index $ind((C \sqcup D)(a))$, and the ABox $\mathcal{A}_2$ is obtained by extending $\mathcal{A}_0$ by $D(a)$ with index $ind((C \sqcup D)(a))$.

The exists-restriction rule. Assume that $(\exists R.C)(a)$ is in $\mathcal{A}_0$, and that $\mathcal{A}_0$ does not contain assertions $R(a, c)$ and $C(c)$ whose indices are both implied by $ind((\exists R.C)(a))$. One generates a new individual name b, and obtains $\mathcal{A}_1$ from $\mathcal{A}_0$ by adding $R(a, b)$ and $C(b)$, both with index $ind((\exists R.C)(a))$.

The value-restriction rule. Assume that $(\forall R.C)(a)$ and $R(a, b)$ are in $\mathcal{A}_0$, and that $\mathcal{A}_0$ does not contain an assertion $C(b)$ whose index is implied by $ind((\forall R.C)(a)) \wedge ind(R(a, b))$. The ABox $\mathcal{A}_1$ is obtained by extending $\mathcal{A}_0$ by $C(b)$ with index $ind((\forall R.C)(a)) \wedge ind(R(a, b))$.

Figure 3: Transformation rules of the labelled consistency algorithm for $\mathcal{ALC}$.

We have used conjunction when expressing a single clash because both assertional facts are necessary for the contradiction. Now recall that we need *at least one* clash *in each* of the complete ABoxes to have inconsistency. This explains why disjunction is used to combine the formulae expressing the clashes of one complete ABox, and why the formulae corresponding to

the different complete ABoxes are combined with the help of conjunction.

Proposition 6.3 *Let ψ be the clash formula associated with $\mathcal{A} \cup \mathcal{B}$, let $\mathcal{Q} \subseteq \mathcal{B}$, and let ω be the valuation which replaces the propositional variables corresponding to elements of $\mathcal{Q}$ by "true" and the others by "false." Then $\mathcal{A} \cup \mathcal{Q}$ is inconsistent iff ψ evaluates to "true" under ω.*

Before proving this proposition we point out how the clash formula can be used to find minimal (resp. maximal) subsets $\mathcal{Q}$ of $\mathcal{B}$ such that $\mathcal{A} \cup \mathcal{Q}$ is inconsistent (resp. consistent). By Proposition 6.3, such minimal (resp. maximal) sets directly correspond to minimal (resp. maximal) valuations making the clash formula ψ "true" (resp. "false"). Here "minimal" and "maximal" for valuations is meant with respect to the partial ordering $\omega_1 \leq \omega_2$ iff $\omega_1(p_i) \leq \omega_2(p_i)$ for all propositional variables p_i, where we assume that "false" is smaller than "true."

It is easy to see that the problem of finding maximal valuations making a monotonic boolean formula "false" can be reduced to the problem of finding minimal valuations making a monotonic boolean formula "true." In fact, for a given monotonic boolean formula ψ and a valuation ω, let ψ^d denote the formula obtained from ψ by replacing conjunction by disjunction and vice versa, and let ω^d denote the valuation obtained from ω by replacing "true" by "false" and vice versa. Then ω is a maximal valuation making ψ "false" iff ω^d is a minimal valuation making ψ^d "true."

It should be noted that the problem of finding minimal valuations that make a monotonic boolean formula ψ "true" is NP-complete. In fact, if ψ is in conjunctive normal form, this is just the well-known problem of finding minimal hitting sets [26, 9]. On the other hand, if ψ is in disjunctive normal form, the minimal valuations can be found in polynomial time. However, transforming a given monotonic boolean formula into disjunctive normal form may cause an exponential blow-up. To optimize the search for minimal valuations one can use the method described in [27].

The rules of the labelled consistency algorithm as described have the unpleasant property that deciding whether or not a rule is applicable is an NP-hard problem. In fact, the preconditions of the rules include an entailment test for monotonic boolean formulae, which is NP-hard. However, one can weaken the precondition by testing a necessary condition for entailment (e.g. occurrence of the index in the top-level disjunction) without destroying termination and the property stated in Proposition 6.3. In this case, the rules will in general produce longer formulae occurring as indices, but the test whether a rule applies becomes tractable.

Proof of Proposition 6.3
First we shall explain the connection between application of rules of the labelled consistency algorithm, starting with $\mathcal{A} \cup \mathcal{B}$, on the one hand, and application of rules of the unlabelled algorithm, starting with $\mathcal{A} \cup \mathcal{Q}$ for $\mathcal{Q} \subseteq \mathcal{B}$, on the other hand.

Definition 6.4 *Let $\mathcal{A}_0$ be a labelled ABox, and let ω be a valuation. The ω-projection of $\mathcal{A}_0$ (for short, $\omega(\mathcal{A}_0)$) is obtained from $\mathcal{A}_0$ by removing all facts whose labels evaluate to "false."*

Let $\mathcal{Q}$ be a subset of $\mathcal{B}$. In the following, the valuation ω is assumed to be such that it replaces the variables corresponding to elements of $\mathcal{Q}$ by "true" and the others by "false." Obviously, this means that $\omega(\mathcal{A} \cup \mathcal{B}) = \mathcal{A} \cup \mathcal{Q}$.

Now we shall show how application of a rule of the labelled consistency algorithm to a labelled ABox $\mathcal{A}_0$ corresponds to application of a rule of the unlabelled algorithm to $\omega(\mathcal{A}_0)$. To get this correspondence, the conditions on applicability of the disjunction and the exists-restriction rules have to be weakened for the unlabelled algorithm:

The modified disjunction rule. Assume that $(C \sqcup D)(a)$ is in $\mathcal{A}_0$, and that $\mathcal{A}_0$ does not contain $C(a)$ *and* $D(a)$. The ABox $\mathcal{A}_1$ is obtained from $\mathcal{A}_0$ by adding $C(a)$, and the ABox $\mathcal{A}_2$ is obtained from $\mathcal{A}_0$ by adding $D(a)$.

The modified exists-restriction rule. Assume that $(\exists R.C)(a)$ is in $\mathcal{A}_0$. One generates a new individual name b, and obtains $\mathcal{A}_1$ from $\mathcal{A}_0$ by adding $R(a, b)$ and $C(b)$.

Since the modified exists-restriction rule can be applied infinitely often to the same fact $(\exists R.C)(a)$ the modified set of rules need no longer terminate. But it is easy to see that the first two properties stated in Proposition 6.1 still hold. This will be sufficient for our purposes.

Lemma 6.5 *Let $\mathcal{A}_0, \mathcal{A}_1$ be labelled ABoxes such that $\mathcal{A}_1$ is obtained from $\mathcal{A}_0$ by application of the conjunction (resp. exists-restriction, value-restriction) rule. Then we either have $\omega(\mathcal{A}_1) = \omega(\mathcal{A}_0)$, or $\omega(\mathcal{A}_1)$ is obtained from $\omega(\mathcal{A}_0)$ by application of the (unlabelled) conjunction (resp. modified exists-restriction, value-restriction) rule.*

Proof. (1) Assume that the *conjunction rule* is applied to the assertional fact $(C \sqcap D)(a)$, and that this fact has index ϕ in $\mathcal{A}_0$.

First, consider the case where $\omega(\phi) = \mathit{false}$. In this case, we have $\omega(\mathcal{A}_1) = \omega(\mathcal{A}_0)$. In fact, if $C(a)$ (resp. $D(a)$) is not in $\mathcal{A}_0$ then this fact has index ϕ in

$\mathcal{A}_1$. Since $\omega(\phi) = \mathit{false}$ this means that $C(a)$ (resp. $D(a)$) is not in $\omega(\mathcal{A}_1)$. If $C(a)$ (resp. $D(a)$) is an element of $\mathcal{A}_0$ with index ψ then $C(a)$ (resp. $D(a)$) has index $\psi \vee \phi$ in $\mathcal{A}_1$. Since $\omega(\phi) = \mathit{false}$ we have $\omega(\psi \vee \phi) = \omega(\psi)$, which shows that $C(a)$ (resp. $D(a)$) is an element of $\omega(\mathcal{A}_1)$ iff it is an element of $\omega(\mathcal{A}_0)$.

Now assume that $\omega(\phi) = \mathit{true}$. Thus $(C \sqcap D)(a)$ is an element of $\omega(\mathcal{A}_0)$. Since $\mathcal{A}_1$ is obtained by extending $\mathcal{A}_0$ by $C(a)$ and $D(a)$, both with index ϕ, we also know that $C(a)$ and $D(a)$ are contained in $\omega(\mathcal{A}_1)$. If both facts are already present in $\omega(\mathcal{A}_0)$ we have $\omega(\mathcal{A}_1) = \omega(\mathcal{A}_0)$. Otherwise, $\omega(\mathcal{A}_1)$ can be obtained from $\omega(\mathcal{A}_0)$ by applying the conjunction rule to $(C \sqcap D)(a)$.

(2) Assume that the *value-restriction rule* is applied to the assertional facts $(\forall R.C)(a)$ and $R(a, b)$, and that these facts respectively have index ϕ_1 and ϕ_2 in $\mathcal{A}_0$.

As for the conjunction rule, $\omega(\phi_1 \wedge \phi_2) = \mathit{false}$ implies $\omega(\mathcal{A}_1) = \omega(\mathcal{A}_0)$. Thus assume that $\omega(\phi_1 \wedge \phi_2) = \mathit{true}$. Then $(\forall R.C)(a)$ and $R(a, b)$ are contained in $\omega(\mathcal{A}_0)$. Since $\mathcal{A}_1$ is obtained by extending $\mathcal{A}_0$ by $C(b)$ with index $\phi_1 \wedge \phi_2$, we know that $C(b)$ is an element of $\omega(\mathcal{A}_1)$. If this assertional fact is already present in $\omega(\mathcal{A}_0)$ then $\omega(\mathcal{A}_1) = \omega(\mathcal{A}_0)$. Otherwise, $\omega(\mathcal{A}_1)$ can be obtained from $\omega(\mathcal{A}_0)$ by applying the value-restriction rule to $(\forall R.C)(a)$ and $R(a, b)$.

(3) Assume that the *exists-restriction rule* is applied to the assertional fact $(\exists R.C)(a)$, and that this fact has index ϕ in $\mathcal{A}_0$.

The case where $\omega(\phi) = \mathit{false}$ is again trivial. Thus assume that $\omega(\phi) = \mathit{true}$. Then $(\exists R.C)(a)$ is an element of $\omega(\mathcal{A}_0)$. The labelled ABox $\mathcal{A}_1$ is obtained from $\mathcal{A}_0$ by generating a new individual b, and adding $C(b)$ and $R(a, b)$ to $\mathcal{A}_0$, both with index ϕ. For this reason, $C(b)$ and $R(a, b)$ are contained in $\omega(\mathcal{A}_1)$. We can obtain $\omega(\mathcal{A}_1)$ from $\omega(\mathcal{A}_0)$ by applying the modified exists-restriction rule to $(\exists R.C)(a)$ (without loss of generality we may assume that the newly generated individual is called b). It should be noted that the (unmodified) exists-restriction rule need not be applicable since $\omega(\mathcal{A}_0)$ may well contain an individual c and assertions $C(c)$ and $R(a, c)$. □

For the disjunction rule, we have a similar lemma.

Lemma 6.6 *Let $\mathcal{A}_0, \mathcal{A}_1, \mathcal{A}_2$ be labelled ABoxes such that $\mathcal{A}_1, \mathcal{A}_2$ are obtained from $\mathcal{A}_0$ by application of the disjunction rule. Then we either have $\omega(\mathcal{A}_1) = \omega(\mathcal{A}_0) = \omega(\mathcal{A}_2)$, or $\omega(\mathcal{A}_1), \omega(\mathcal{A}_2)$ are obtained from $\omega(\mathcal{A}_0)$ by application of the (unlabelled) modified disjunction rule.*

Proof. Assume that the disjunction rule is applied to the assertional fact $(C \sqcup D)(a)$, and that this fact has index ϕ in $\mathcal{A}_0$.

If $\omega(\phi) = \mathit{false}$ then $\omega(\mathcal{A}_1) = \omega(\mathcal{A}_0) = \omega(\mathcal{A}_2)$. This can be shown as in the corresponding cases in the proof of Lemma 6.5.

Thus assume that $\omega(\phi) = \mathit{true}$. Then $(C \sqcup D)(a)$ is an element of $\omega(\mathcal{A}_0)$. In addition, we know that $C(a)$ is contained in $\omega(\mathcal{A}_1)$ and that $D(a)$ is contained in $\omega(\mathcal{A}_2)$. If both $C(a)$ and $D(a)$ are already present in $\omega(\mathcal{A}_0)$ then $\omega(\mathcal{A}_1) = \omega(\mathcal{A}_0) = \omega(\mathcal{A}_2)$. Otherwise, we can obtain $\omega(\mathcal{A}_1), \omega(\mathcal{A}_2)$ from $\omega(\mathcal{A}_0)$ by applying the modified disjunction rule to $(C \sqcup D)(a)$. It should be noted that the (unmodified) disjunction rule need not be applicable since $\omega(\mathcal{A}_0)$ may well contain one of $C(a)$ and $D(a)$, but not both. □

Now assume that we have obtained the complete ABoxes $\mathcal{A}_1, \ldots, \mathcal{A}_n$ by starting with $\mathcal{A} \cup \mathcal{B}$, and applying the rules of the labelled consistency algorithm as long as possible. By Lemma 6.5 and 6.6, and since the (modified) rules of the unlabelled consistency algorithm preserve solvability, we know that $\omega(\mathcal{A} \cup \mathcal{B}) = \mathcal{A} \cup \mathcal{Q}$ is consistent iff one of $\omega(\mathcal{A}_1), \ldots, \omega(\mathcal{A}_n)$ is consistent. The next lemma implies that these projected ABoxes are also complete.

Lemma 6.7 *Let $\mathcal{A}_0$ be a labelled ABox to which none of the rules of the labelled consistency algorithm applies. Then none of the (unmodified) rules of the unlabelled consistency algorithm applies to $\omega(\mathcal{A}_0)$.*

Proof. We consider an assertional fact $(C \sqcap D)(a)$ in $\omega(\mathcal{A}_0)$, and show that the conjunction rule cannot be applied to this fact in $\omega(\mathcal{A}_0)$. (The other cases can be treated similarly.)

Since $(C \sqcap D)(a)$ is present in $\omega(\mathcal{A}_0)$ its index ϕ in $\mathcal{A}_0$ satisfies $\omega(\phi) = \mathit{true}$. Completeness of $\mathcal{A}_0$ implies that the (labelled) conjunction rule is not applicable to $(C \sqcap D)(a)$ in $\mathcal{A}_0$. For this reason, $\mathcal{A}_0$ contains the assertional facts $C(a)$ and $D(a)$, and their indices (say ψ_1, ψ_2) are implied by ϕ. But then $\omega(\phi) = \mathit{true}$ implies $\omega(\psi_1) = \mathit{true} = \omega(\psi_2)$. Thus $C(a)$ and $D(a)$ are contained in $\omega(\mathcal{A}_0)$, which shows that the conjunction rule is not applicable to $(C \sqcap D)(a)$ in $\omega(\mathcal{A}_0)$. □

Since $\mathcal{A}_1, \ldots, \mathcal{A}_n$ are complete we thus know that $\omega(\mathcal{A}_1), \ldots, \omega(\mathcal{A}_n)$ are complete as well. Now Proposition 6.1 implies that $\omega(\mathcal{A}_i)$ is inconsistent iff it contains a clash. A particular clash $A(a), \neg A(a) \in \mathcal{A}_i$ is still present in $\omega(\mathcal{A}_i)$ iff ω evaluates $\mathit{ind}(A(a)) \wedge \mathit{ind}(\neg A(a))$ to "true." Now let $\psi_{i,1}, \ldots, \psi_{i,k_i}$ be the formulae expressing all the clashes in $\mathcal{A}_i$. Obviously, $\omega(\mathcal{A}_i)$ contains a clash iff ω evaluates $\bigvee_{j=1}^{k_i} \psi_{i,j}$ to "true." For this reason, all the ABoxes $\omega(\mathcal{A}_1), \ldots, \omega(\mathcal{A}_n)$ contain a clash iff ω evaluates to "true" the clash formula

$$\bigwedge_{i=1}^{n} \bigvee_{j=1}^{k_i} \psi_{i,j}$$

computed by the labelled consistency algorithm. This concludes the proof of Proposition 6.3.

Extension to $\mathcal{ALCF}$

In the remaining part of this section we shall sketch how the above described algorithm can be extended to handle the attributes and agreements of $\mathcal{ALCF}$.

Attributes in exists- and value-restrictions are treated like roles. Applying the exists-restriction rule to two assertional facts $(\exists f.C)(a)$ and $(\exists f.D)(a)$ introduces two different individual names c, d with the assertional facts $f(a, c)$, $f(a, d)$. If f is an attribute, this means that c and d have to be interpreted as the same individual. This shows that we can no longer have a unique name assumption for the individuals which are introduced by rules. For this reason, we shall now distinguish between "old" individuals, i.e., individuals present in the original ABox $\mathcal{A} \cup \mathcal{B}$, and "new" individuals introduced by rule applications. New individuals are not subjected to the unique name assumption. In order to make the constraint that c, d have to be interpreted by the same individual explicit, the consistency algorithm for $\mathcal{ALCF}$ (see [11]) identifies these two individual names, e.g., by replacing every occurrence of c by d. In the labelled consistency algorithm, instead of making an actual replacement, we just introduce an equality fact $c = d$. Of course, this equality has to be equipped with an index, in the same way as other facts are. Here the fact $c = d$ gets index $\mathit{ind}(f(a, c)) \wedge \mathit{ind}(f(a, d))$ if it is newly introduced, otherwise one takes the disjunction of its old index with $\mathit{ind}(f(a, c)) \wedge \mathit{ind}(f(a, d))$. In case $\mathit{ind}(f(a, c)) \wedge \mathit{ind}(f(a, d))$ implies the old index, nothing has to be changed.

With the help of the equality facts, it is easy to formulate an *agreement rule*. In principle, the agreement rule applied to $(f_1 \cdots f_m \doteq g_1 \cdots g_n)(a)$ introduces the assertional facts $f_1(a, c_1)$, ..., $f_m(c_{m-1}, c_m)$, $g_1(a, d_1)$, ..., $g_n(d_{n-1}, d_n)$ and $c_m = d_n$, where c_1, ..., d_n are new individual names. Applicability of this rule, and the indices of the new facts (or new indices of existing facts) are defined analogously to the other rules.

The equality facts define an equivalence relation on individual names, which has to be taken into account when firing rules or looking for clashes. Premises of rules have to be read modulo this equivalence. For example, this means that the value-restriction rule may be applicable to the facts $(\forall R.C)(a)$ and $R(a', b)$, if there are equalities $a = a_0, a_1 = a_2, \ldots, a_n = a'$ in the ABox. Of course, the indices of these equalities have to contribute to the new index of $C(b)$ as well. On the other hand, this rule need not be applied if there exists an assertional fact $C(b')$ and equalities $b = b_0, b_1 = b_2, \ldots, b_m = b'$ such that $\mathit{ind}((\forall R.C)(a)) \wedge \mathit{ind}(R(a', b)) \wedge \mathit{ind}(a = a_0) \ldots \wedge \mathit{ind}(a_n = a')$ implies $\mathit{ind}(C(b')) \wedge \mathit{ind}(b = b_0) \ldots \wedge \mathit{ind}(b_m = b')$.

Similarly, there is a clash if $A(a)$ and $\neg A(a')$ is in the ABox, along with equalities $a = a_0, a_1 = a_2, \ldots, a_n = a'$. Because we still have unique name assumption for the old individuals, the equalities may cause another kind of

obvious contradiction. We have a clash if a, a' are old individuals and there are equalities $a = a_0, a_1 = a_2, \ldots, a_n = a'$ in the ABox. The index associated with this clash is $ind(a = a_0) \wedge \ldots \wedge ind(a_n = a')$.

To sum up, we thus have a solution of the two algorithmic problems described at the beginning of this section. Together with the methods of Section 5 this give us effective procedures to compute all extensions of terminological default theories.

7 Conclusion

We have investigated the integration of Reiter's default logic into a terminological representation formalism, and have shown that the treatment of open defaults by Skolemization is problematic, both from a semantic and an algorithmic point of view. For this reason, we have considered a restricted semantics where default rules are only applied to individuals explicitly present in the knowledge base. This treatment of default rules is similar to the treatment of monotonic rules in many terminological systems, which means that users of such systems are already familiar with the effects this restriction to explicit individuals has. However, because of the nonmonotonic character of default rules, this restriction may sometimes lead to more consequences than would have been obtained without it.

With respect to the restricted semantics, the methods of Junker and Konolige and of Schwind and Risch for computing all extensions of a default theory can be applied. We have shown how the algorithmic requirements for Junker and Konolige's method (i.e., the computation of minimal inconsistent sets of assertional facts) and for an optimized algorithm based on a theorem of Schwind and Risch (i.e., the computation of maximal consistent sets of assertional facts) can be solved by an extension of the tableaux-based algorithm for assertional reasoning.

As an alternative to the pragmatic solution described in the present paper, [4] proposes a new semantics for open defaults, in which defaults are also applied to implicit individuals. To make this possible without encountering the problems pointed out in Section 3, open defaults are not viewed as schemata for certain instantiated defaults. Instead, they are used to define a preference relation on models, which is then treated with a modified preferential approach.

According to Reiter's semantics the specificity of prerequisites of rules has no influence on the order in which defaults rules are supposed to fire. In [3] we describe a modification of terminological default logic in which more specific defaults are preferred.

Acknowledgements

We should like to thank Bernhard Nebel and Peter Patel-Schneider for helpful comments.

This work has been supported by the German Ministry for Research and Technology (BMFT) under research contract ITW 8903 0.

References

[1] F. Baader and P. Hanschke. A Scheme for Integrating Concrete Domains into Concept Languages. Research Report RR-91-10, DFKI Kaiserslautern, 1991.

[2] F. Baader and P. Hanschke. A scheme for integrating concrete domains into concept languages. In *Proceedings of the 12th International Joint Conference on Artificial Intelligence*, Sydney, Australia, 1991.

[3] F. Baader and B. Hollunder. How to prefer more specific defaults in terminological default logic. Research Report RR-92-58, DFKI Saarbrücken, 1992. Also to appear in *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, Chambery, France, 1993.

[4] F. Baader and K. Schlechta. A semantics for open normal defaults via a modified preferential approach. Research Report RR-93-13, DFKI Saarbrücken, 1993.

[5] R. J. Brachman. 'I lied about the trees' or, defaults and definitions in knowledge representation. *The AI Magazine*, 6(3):80–93, 1985.

[6] R. J. Brachman, D. L. McGuinness, P. F. Patel-Schneider, L. A. Resnick, and A. Borgida. Living with CLASSIC: When and how to use a KL-ONE-like language. In J. Sowa, editor, *Principles of Semantic Networks*, pages 401–456. Morgan Kaufmann, San Mateo, Calif., 1991.

[7] R. J. Brachman and J. G. Schmolze. An overview of the KL-ONE knowledge representation system. *Cognitive Science*, 9(2):171–216, 1985.

[8] J. Doyle. A truth maintenance system. *Artificial Intelligence*, 12:231–272, 1979.

[9] M. Garey and D. Johnson. *Computers and Intractability – A Guide to the Theory of NP-Completeness.* Freeman, San Francisco, Cal., 1979.

[10] B. Hollunder. Hybrid inferences in KL-ONE-based knowledge representation systems. In *14th German Workshop on Artificial Intelligence*, volume 251 of *Informatik-Fachberichte*, pages 38–47, Ebingerfeld, Germany, 1990. Springer.

[11] B. Hollunder and W. Nutt. Subsumption Algorithms for Concept Languages. Research Report RR-90-04, DFKI Kaiserslautern, 1990.

[12] U. Junker and K. Konolige. Computing extensions of autoepistemic and default logics with a truth maintenance system. In *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 278–283, Boston, Ma., 1990.

[13] H. A. Kautz and B. Selman. Hard problems for simple defaults. In *Proceedings of the 1st International Conference on Principles of Knowledge Representation and Reasoning*, pages 189–197, Toronto, Ont., 1989.

[14] A. Kobsa. The SB-ONE knowledge representation workbench. In *Preprints of the Workshop on Formal Aspects of Semantic Networks*, Two Habours, Calif., 1989.

[15] V. Lifschitz. On open defaults. In *Proceedings of the Symposium on Computational Logics*, Brüssel, Belgium, 1990.

[16] E. Mays and B. Dionne. Making KR systems useful. In *Terminological Logic Users Workshop – Proceedings*, pages 11–12, KIT-Report 95, TU Berlin, 1991.

[17] J. McCarthy. Circumscription – a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[18] D. McDermott and J. Doyle. Non-monotonic logic I. *Artificial Intelligence*, 13:41–72, 1980.

[19] R. McGregor. Statement of interest. In K. von Luck, B. Nebel, and C. Peltason, editors, *Statement of Interest for the 2nd International Workshop on Terminological Logics*. Document D-91-13, DFKI Kaiserslautern, 1991.

[20] μBACK. System presentation. In *Terminological Logic Users Workshop – Proceedings*, page 186, KIT-Report 95, TU Berlin, 1991.

[21] B. Nebel and G. Smolka. Attributive description formalisms ... and the rest of the world. In C. Rollinger O. Herzog, editor, *Text Understanding in LILOG*, LNAI 546. Springer-Verlag, Berlin, Germany, 1991.

[22] C. Peltason, K. v. Luck, and C. Kindermann (Org.). Terminological logic users workshop – Proceedings. KIT Report 95, TU Berlin, 1991.

[23] D. L. Poole. Variables in hypothesis. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, Milano, Italy, 1987.

[24] E. L. Post. Recursive unsolvability of a problem of Thue. *Journal of Symbolic Logic*, 12:1–10, 1947.

[25] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1-2):81–132, 1980.

[26] R. Reiter. A theory of diagnosis from first principles. *Artificial Intelligence*, 32:57–95, 1987.

[27] R. Rymon. Search through systematic set enumeration. In *Proceedings of the 3rd International Conference on Knowledge Representation and Reasoning*, Cambridge, Mass., 1992.

[28] C. Schwind and V. Risch. A tableau-based characterisation for default logic. In *Proceedings of the 1st European Conference on Symbolic and Quantitative Approaches for Uncertainty*, pages 310–317, Marseille, France, 1991.