Towards Efficient Default Reasoning

Ilkka Niemelä
Department of Computer Science
Helsinki University of Technology
Otakaari 1, FIN–02150 Espoo, Finland
Ilkka.Niemela@hut.fi

Abstract

A decision method for Reiter's default logic is developed. It can determine whether a default theory has an extension, whether a formula is in some extension of a default theory and whether a formula is in every extension of a default theory. The method handles full propositional default logic. It can be implemented to work in polynomial space and by using only a theorem prover for the underlying propositional logic as a subroutine. The method divides default reasoning into two major subtasks: the search task of examining every alternative for extensions, which is solved by backtracking search, and the classical reasoning task, which can be implemented by a theorem prover for the underlying classical logic. Special emphasis is given to the search problem. The decision method employs a new compact representation of extensions which reduces the search space. Efficient techniques for pruning the search space further are developed.

1 Introduction

In this paper we develop a theorem-proving method for default logic of Reiter [1980]. Default logic is one of the most well-known nonmonotonic logics [Marek and Truszczyński, 1993]. There is a body of results indicating that default logic captures a large number of different forms of nonmonotonic reasoning. Default logic is closely related to logic programs and deductive databases [Gelfond and Lifschitz, 1990]. Connections have been established between default logic and autoepistemic logic and McDermott and Doyle style nonmonotonic modal logics [Konolige, 1988; Truszczyński, 1991], circumscription [Etherington, 1987], diagnosis [Reiter, 1987], and abductive reasoning [Poole, 1988].

In default logic knowledge is represented by a default theory, which consists of ordinary first-order formulae and nonmonotonic inference rules, *default rules*. Possible sets of conclusions from a default theory are defined in terms of *extensions* of the default theory. In this paper we consider three basic reasoning tasks: *extension existence* (whether a default theory has an extension), *brave reasoning* (whether a formula is in some extension of a default theory) and *cautious reasoning* (whether a formula belongs to every extension of a default theory).

Computational properties of nonmonotonic reasoning have received considerable attention. This research provides a valuable basis for developing nonmonotonic theorem-proving techniques. The complexity of propositional default reasoning has been located at the second level of the polynomial time hierarchy [Garey and Johnson, 1979]: extension existence and brave reasoning are Σ_2^p-complete and cautious reasoning Π_2^p-complete [Gottlob, 1992]. This means that full propositional default logic can be implemented in polynomial space and that it is strictly harder than propositional reasoning unless the polynomial time hierarchy collapses.

Developing theorem-proving techniques for default logic and other nonmonotonic logics has turned out to be difficult. Existing techniques are quite straightforward and little emphasis has been laid on efficiency considerations. For example, in recent approaches [Junker and Konolige, 1990; Risch and Schwind, 1994; Baader and Hollunder, 1992] to automating default logic exponential space is needed or the methods are based on solving subtasks that seem to be computationally harder than the original default reasoning task.

We consider first approaches where a reasoning problem in default logic is reduced into another problem like a truth maintenance problem [Junker and Konolige, 1990] or a constraint satisfaction problem [Ben-Eliyahu and Dechter, 1991]. A crucial feature of the reduction mappings is that classical deductions needed in default reasoning have to be encoded. This implies that reductions are computationally feasible only for very restricted subclasses of default logic. Typically, the reductions lead to an exponential increase in the problem size because of the exponential number of deductions to be encoded. Moreover, computing the reduction mapping appears to be harder than the original default reasoning problem. This is because even the problem of finding a single deduction, (i.e., a proof of a given formula from a set of formulae) is closely related to logic-based abduction, which is Σ_2^p-complete [Eiter and Gottlob, 1992]. For a more detailed discussion, see [Niemelä, 1994].

Risch and Schwind [1994] propose a tableau-based method for finding extensions. Also this approach suffers from the problem of exponential worst-case space complexity. Baader and Hollunder [1992] present an approach to generate all extensions of a default theory by pruning defaults in a top-down way. When eliminating defaults, this method uses heavily a subroutine computing all maximal consistent subsets, i.e., given sets Σ and H the subroutine is expected to find all maximal subsets H' of H such that $\Sigma \cup H'$ is consistent. It seems that finding *all* such maximal subsets is computationally expensive and at least as hard as the decision problems in default logic. This is because, for example, finding a maximal subset not containing a given formula in H is closely related to logic-based abduction. To see this consider a maximal subset $H' \subseteq H$ for which $\Sigma \cup H'$ is consistent but $\psi \in H - H'$. Hence, $\Sigma \cup H' \cup \{\psi\}$ is not consistent which implies that $\neg\psi$ is a logical consequence of $\Sigma \cup H'$, i.e. H' is an abductive explanation of $\neg\psi$ from the hypotheses H and the background theory Σ.

In this paper we develop a decision method for default logic that handles important subclasses of default reasoning (i.e., the full propositional case as well as closed default rules together with a decidable fragment of the underlying first-order logic) but does not suffer from the two problems: (i) exponential space requirements and (ii) the use of computationally too difficult subtasks as a part of the method. As a basis of the work we have taken a decision method for autoepistemic logic presented in [Niemelä, 1994] because this approach satisfies the two requirements. As autoepistemic logic and default logic are closely related [Konolige, 1988; Niemelä, 1992], the approach is directly applicable to default logic. However, there seems to be some room for improvements. In this paper we take the basic ideas from [Niemelä, 1994] and apply them directly to default logic in order to fully exploit the special characteristics of default reasoning.

The paper is organized as follows. Section 2 introduces default logic and develops a concise representation of extensions. This representation is based on ideas from autoepistemic reasoning and it forms the basis for the decision method. Section 3 develops a basic algorithm for default reasoning. It provides a framework for integrating optimization techniques. Section 4 presents optimizations of the basic algorithm and Section 5 contains the concluding remarks.

2 Default Logic

We are going to use intuitions from autoepistemic reasoning and to facilitate this we employ somewhat nonstandard notations for default logic. First, we introduce a new operator $\mathrm{nb}(\cdot)$: $\mathrm{nb}(\phi)$ expresses that a formula ϕ is "not believed", i.e. ϕ does *not belong* to the extension in question. Second, we write default rules using the new operator. A *default rule* is an expression of the form

$$a_1, \ldots, a_n, \mathrm{nb}(b_1), \ldots, \mathrm{nb}(b_m) \hookrightarrow c \qquad (1)$$

where $a_1, \ldots, a_n, b_1, \ldots, b_m, c$ are arbitrary (first-order) formulae. This is just an alternative notation for a default rule in the standard form

$$\frac{a_1 \wedge \cdots \wedge a_n : \neg b_1, \ldots, \neg b_m}{c}.$$

A *default theory* is a set of default rules of the form (1). In Reiter's [1980] original presentation a default theory can contain ordinary first-order formulae in addition to default rules. Here for uniformity the first-order formulae are represented as default rules, i.e., a first-order formula ϕ is represented as a rule $\hookrightarrow \phi$.

A default rule of the form (1) can be thought of as representing an autoepistemic formula

$$La_1 \wedge \cdots \wedge La_n \wedge L\neg Lb_1 \wedge \cdots \wedge L\neg Lb_m \rightarrow c.$$

Under this translation, proposed by Truszczyński [1991], an extension of a default theory is the L free part of an L-hierarchic expansion of the translated theory [Niemelä, 1992] or the L free part of a consistent S-expansion of the translated theory for a range of nonmonotonic modal logics S [Truszczyński, 1991]. Hence default rules of the form (1) provide an interesting "standard" form of autoepistemic formulae.

Next we give the definition of an extension of a default theory. Technically our definition is somewhat different from that given by Reiter [1980] but it leads to the same class of extensions. The definition is given by using the notions of a deductive closure and NB-formulae.

We call NB-formulae expressions of the form $\mathrm{nb}(\phi)$ where ϕ is a formula. For a set of formulae S, $\mathrm{nb}(S) = \{\mathrm{nb}(\phi) \mid \phi \in S\}$. By S^+ (S^-) we denote set of formulae (NB-formulae) in S. For example, if $S = \{a, \mathrm{nb}(b), \mathrm{nb}(c)\}$, $S^+ = \{a\}$ and $S^- = \{\mathrm{nb}(b), \mathrm{nb}(c)\}$.

We denote by $\mathrm{Dcl}(\Sigma, L)$ the *deductive closure* of a set of rules Σ of the form (1) and a set of formulae and NB-formulae L. $\mathrm{Dcl}(\Sigma, L)$ is the smallest set of formulae which contains L^+ and which is closed under Σ' and first-order derivations where

$$\begin{aligned} \Sigma' \;=\; & \{a_1, \ldots, a_n \hookrightarrow c \mid \\ & a_1, \ldots, a_n, \mathrm{nb}(b_1), \ldots, \mathrm{nb}(b_m) \hookrightarrow c \in \Sigma \\ & \text{and for all } i = 1, \ldots, m, \mathrm{nb}(b_i) \in L^-\}. \quad (2) \end{aligned}$$

For example, let $\Sigma = \{\mathrm{nb}(p) \hookrightarrow q;\ \neg\neg q \hookrightarrow \neg\neg r\}$ and $L = \{p, \mathrm{nb}(p)\}$. Then $\Sigma' = \{\hookrightarrow q;\ \neg\neg q \hookrightarrow \neg\neg r\}$ and $\mathrm{Dcl}(\Sigma, L) = \mathrm{Th}(\{p, q, \neg\neg r\})$ where $\mathrm{Th}(S)$ denotes the set of the first-order consequences of a set of formulae S. This means that, e.g., $r \in \mathrm{Dcl}(\Sigma, \{p, \mathrm{nb}(p)\})$.

Notice that the deductive closure is a monotonic operator also with respect to the premises L: if $L \subseteq L'$, then $\mathrm{Dcl}(\Sigma, L) \subseteq \mathrm{Dcl}(\Sigma, L')$.

For a set of rules Σ, a set of formulae Δ is called an *extension* of Σ iff $\Delta = \mathrm{Dcl}(\Sigma, \mathrm{nb}(\overline{\Delta}))$ where $\overline{\Delta}$ is the complement of Δ, i.e., the formulae not in Δ.

We develop a compact characterizing condition for extensions. The formulae that occur inside the $\mathrm{nb}(\cdot)$ operator play a crucial role. For a set of rules Σ, we denote by $\mathrm{NAnt}(\Sigma)$ the *negative antecedents* in Σ, i.e., the set of formulae b such that $\mathrm{nb}(b)$ appears in Σ. For example, $\mathrm{NAnt}(\{\mathrm{nb}(p) \hookrightarrow q; \neg\neg q \hookrightarrow \neg\neg r\}) = \{p\}$. The following proposition makes the role of $\mathrm{NAnt}(\Sigma)$ evident. It is based on the observation that for the deductive closure of a set of rules only the NB-formulae appearing in the rules are of importance, i.e., $\mathrm{Dcl}(\Sigma, \mathrm{nb}(\overline{\Delta})) = \mathrm{Dcl}(\Sigma, \mathrm{nb}(\mathrm{NAnt}(\Sigma) - \Delta))$.

Proposition 2.1 *For a set of rules Σ, a set of formulae Δ is an extension of Σ iff $\Delta = \mathrm{Dcl}(\Sigma, \mathrm{nb}(\mathrm{NAnt}(\Sigma) - \Delta))$.*

The situation is similar to that in autoepistemic logic where a stable expansion is uniquely determined by the modal subformulae in the premises [Niemelä, 1990]. For characterizing extensions, we are able to use ideas from the full set based characterization of stable expansions [Niemelä, 1990]. The novelty here is that we exploit the strong groundedness of extensions which implies that ordinary formulae appearing as antecedents of rules (prerequisites) do not play a role in determining extensions and only NB-formulae (justifications) are essential.

Our aim is to provide a compact characterizing set for each extension. The characterizing sets are called full sets and they are sets of NB-formulae built from formulae in $\mathrm{NAnt}(\Sigma)$.

Definition 2.2 *For a set of rules Σ, a set Λ of NB-formulae is called Σ-full iff the following condition holds for all $\phi \in \mathrm{NAnt}(\Sigma)$, $\mathrm{nb}(\phi) \in \Lambda$ iff $\phi \notin \mathrm{Dcl}(\Sigma, \Lambda)$.*

For example, let $\Sigma = \{\mathrm{nb}(p) \hookrightarrow \neg p;\ \mathrm{nb}(\neg p) \hookrightarrow p\}$. Then $\{\mathrm{nb}(p)\}$ is Σ-full, because $p \notin \mathrm{Dcl}(\Sigma, \{\mathrm{nb}(p)\})$ and $\neg p \in \mathrm{Dcl}(\Sigma, \{\mathrm{nb}(p)\})$ but, e.g., $\emptyset$ is not Σ-full, as $p \notin \mathrm{Dcl}(\Sigma, \emptyset)$. It turns out that for every full set there is an extension and for each extension a corresponding full set.

Theorem 2.3 *Let Σ be a set of rules.*

(i) If a set Λ is Σ-full, $\mathrm{Dcl}(\Sigma, \Lambda)$ is an extension of Σ.

(ii) If there is an extension Δ of Σ, then $\Lambda = \mathrm{nb}(\mathrm{NAnt}(\Sigma) - \Delta)$ is a Σ-full set such that $\Delta = \mathrm{Dcl}(\Sigma, \Lambda)$.

Proof. (i) Let $\Delta = \mathrm{Dcl}(\Sigma, \Lambda)$. If $\phi \in \mathrm{NAnt}(\Sigma) - \Delta$, then as Λ is Σ-full, $\mathrm{nb}(\phi) \in \Lambda$. If $\mathrm{nb}(\phi) \in \Lambda$, then as Λ is Σ-full, $\phi \in \mathrm{NAnt}(\Sigma) - \Delta$. Thus $\mathrm{nb}(\mathrm{NAnt}(\Sigma) - \Delta) = \Lambda$ holds which by Proposition 2.1 implies that $\mathrm{Dcl}(\Sigma, \Lambda)$ is an extension of Σ.

(ii) Let Δ be an extension of Σ. By Proposition 2.1 $\Delta = \mathrm{Dcl}(\Sigma, \Lambda)$ for $\Lambda = \mathrm{nb}(\mathrm{NAnt}(\Sigma) - \Delta)$. Let $\phi \in \mathrm{NAnt}(\Sigma)$. If $\phi \in \mathrm{Dcl}(\Sigma, \Lambda)$, then $\phi \in \Delta$ and $\mathrm{nb}(\phi) \notin \Lambda$. If $\phi \notin \mathrm{Dcl}(\Sigma, \Lambda)$, then $\phi \notin \Delta$ and $\mathrm{nb}(\phi) \in \Lambda$. Hence, Λ is Σ-full. □

Theorem 2.3 suggests the following straightforward method for finding all extensions. For each subset S of $\mathrm{NAnt}(\Sigma)$, test whether $\mathrm{nb}(S)$ is Σ-full. If a full set Λ is found, $\mathrm{Dcl}(\Sigma, \Lambda)$ is an extension of Σ by Theorem 2.3 (i) and by Theorem 2.3 (ii) for each extension Δ there is a corresponding full set Λ such that $\Delta = \mathrm{Dcl}(\Sigma, \Lambda)$.

Example 2.1 The straightforward method is not very practical. If there are n NB-formulae in Σ, 2^n fullness tests are needed although there can be a single extension which is easily constructible as the following set of rules Σ shows.

$$\begin{aligned}\Sigma = \{&a_0 \hookrightarrow a_1;\ \mathrm{nb}(a_1) \hookrightarrow a_0;\\ &a_1 \hookrightarrow a_2;\ \mathrm{nb}(a_2) \hookrightarrow a_1; \ldots;\\ &a_{n-1} \hookrightarrow a_n;\ \mathrm{nb}(a_n) \hookrightarrow a_{n-1};\ \hookrightarrow a_0\}.\end{aligned}$$

There are 2^n candidates for Σ-full sets but only one Σ-full set $\Lambda = \emptyset$. This can be seen by the following argument. For any set F, $a_0 \in \mathrm{Dcl}(\Sigma, F)$ which implies $a_1 \in \mathrm{Dcl}(\Sigma, F)$. Hence $\mathrm{nb}(a_1)$ cannot belong to any Σ-full set neither can $\mathrm{nb}(a_2)$ and so on. ■

3 The Basic Algorithm

In this section we develop a basic algorithm for solving default reasoning problems. The basic algorithm serves as a framework for developing further optimization methods, which are discussed in the next section. The algorithm is based on ideas introduced in [Niemelä, 1994] in the context of autoepistemic reasoning. The algorithm presented in Figure 1 is given as a function $\mathbf{extensions}_{\mathrm{DL}}$ that is the skeleton for the decision procedures for brave and cautious reasoning as well as for checking the existence of extensions.

When describing the algorithm we use the following three concepts.

(i) A set of formulae and NB-formulae Λ is *grounded* in a set of rules Σ if $\Lambda^+ \subseteq \mathrm{Dcl}(\Sigma, \Lambda^-)$. For example, $\{\mathrm{nb}(b)\}$ and $\{b, \mathrm{nb}(a)\}$ are grounded in $\{\mathrm{nb}(a) \hookrightarrow b;\ a \hookrightarrow a\}$ but $\{a\}$ is not.

(ii) We say that a set of formulae S *agrees* with a set of formulae and NB-formulae Λ if for all formulae $\phi \in \Lambda^+$, $\phi \in S$ and for all $\mathrm{nb}(\phi) \in \Lambda^-$, $\phi \notin S$. For example, the set $\{a, b\}$ agrees with $\{a, \mathrm{nb}(c)\}$ but not with $\{a, \mathrm{nb}(b)\}$.

(iii) A set of formulae and NB-formulae Λ *covers* a formula ϕ if either $\phi \in \Lambda$ or $\mathrm{nb}(\phi) \in \Lambda$. For example, the set $\{\mathrm{nb}(a), b\}$ covers a. A set of formulae S is covered by Λ if each formula in S is covered by Λ.

The function $\mathbf{extensions}_{\mathrm{DL}}$ takes as input a set of rules Σ, sets B and F which determine the common part of the extensions to be considered and a formula ϕ which is just passed as an argument to the function **test**. The aim of $\mathbf{extensions}_{\mathrm{DL}}$ is to return true iff there is an extension Δ of Σ agreeing with $B \cup F$ such that $\mathbf{test}(\Delta, \phi)$ returns true. This is accomplished by constructing Σ-full sets agreeing with $B \cup F$ until a full set Λ is found such that for the corresponding extension $\Delta = \mathrm{Dcl}(\Sigma, \Lambda)$, $\mathbf{test}(\Delta, \phi)$ returns true.

We represent a partially constructed full set using the set B that contains NB-formulae and ordinary formulae. The NB-formulae B^- are the formulae included in the partially constructed full set. An ordinary formula χ in B indicates that the corresponding NB-formula $\mathrm{nb}(\chi)$ cannot be included in the full set. The idea is to expand B until it forms a full set. The number of possibilities can be reduced by observing that if a formula is grounded in the rules Σ given the partially constructed full set B^-, i.e., $\chi \in \mathrm{Dcl}(\Sigma, B^-)$, $\mathrm{nb}(\chi)$ cannot be included in the full set and χ is added to B. The set F contains formulae χ for which $\mathrm{nb}(\chi)$ is excluded from the full set to be constructed (and χ should be included in B), but which are "frozen": $\chi \in F$ is added to B only when the groundedness condition is satisfied, i.e., $\chi \in \mathrm{Dcl}(\Sigma, B^-)$.

The function $\mathbf{extensions}_{\mathrm{DL}}$ uses three functions

- **expand** (for expanding B)
- **conflict** (for detecting conflicts), and
- **test** (for testing extensions).

By changing the function **test** the various decision procedures are obtained. For the first two functions we present minimal requirements (E1–E4, C1–C2) that the implementations of these functions have to satisfy in order to guarantee the soundness and completeness of the decision procedures. These two functions form the crucial

```
function extensions_DL(Σ, B, F, φ)
B := expand(Σ, B, F);
if conflict(Σ, B, F) returns true then
    return false
else if NAnt(Σ) is covered by B ∪ F then
    if F ⊆ B then
        return test(Dcl(Σ, B⁻), φ)
    else
        return false
    endif
else
    take some χ ∈ NAnt(Σ) not covered by B ∪ F;
    if extensions_DL(Σ, B ∪ {nb(χ)}, F, φ)
            returns true then
        return true
    else
        return extensions_DL(Σ, B, F ∪ {χ}, φ)
    endif
endif
```

Figure 1: The Skeleton for the Decision Procedures for Default Logic

points in the algorithm where optimization techniques can be applied. In the next section such optimization methods are developed.

We first introduce the requirements on the functions **expand** and **conflict** and then explain their role in guaranteeing the correctness of $\mathbf{extensions}_{\mathrm{DL}}$. The function **expand** is assumed to fulfill the following conditions where $B_e = \mathbf{expand}(\Sigma, B, F)$.

E1: $B \subseteq B_e$.

E2: If B is grounded in Σ, then B_e is grounded in Σ.

E3: If there is an extension Δ of Σ such that Δ agrees with $B \cup F$, then Δ agrees with $B_e \cup F$.

E4: For all $\chi \in \mathrm{NAnt}(\Sigma)$, $\chi \in \mathrm{Dcl}(\Sigma, B_e^-)$ implies that χ is covered by B_e.

The function **conflict** returns either true or false and satisfies the following conditions.

C1: $\mathbf{conflict}(\Sigma, B, F)$ returns true if for some $\mathrm{nb}(\chi) \in B$, $\chi \in \mathrm{Dcl}(\Sigma, B^-)$.

C2: If $\mathbf{conflict}(\Sigma, B, F)$ returns true, then there exists no extension Δ of Σ such that Δ agrees with $B \cup F$.

The function $\mathbf{extensions}_{\mathrm{DL}}$ starts by expanding cautiously the set B. In order to ensure the correctness of the decision procedures we must insist that the expansion $B_e = \mathbf{expand}(\Sigma, B, F)$ extends B (E1) in such a way that groundedness is preserved (E2) and that no extensions agreeing with $B \cup F$ are lost (E3). To preserve completeness we have to require that each formula in $\mathrm{NAnt}(\Sigma)$ that is grounded but not already covered by B is included in B (E4). Then a conflict test is performed. Here it must be the case that all direct conflicts are detected (C1) and if a conflict is reported, then there is no extension agreeing with $B \cup F$ (C2). If there are no conflicts and $B \cup F$ covers every NB-formula in the premises, then the status of frozen formulae F is examined. If all of them have been included in B, B^- is a full set. The corresponding extension is $\mathrm{Dcl}(\Sigma, B^-)$ and $\mathbf{extensions}_{\mathrm{DL}}$ returns what $\mathbf{test}(\mathrm{Dcl}(\Sigma, B^-), \phi)$ returns.

If there is an NB-formula $\mathrm{nb}(\chi)$ in the premises such that χ is not covered by $B \cup F$, then either χ belongs to the extension to be constructed or not. The two alternatives are handled by backtracking. First B is extended by $\mathrm{nb}(\chi)$ (assuming that χ is not in the extension). If $\mathbf{extensions}_{\mathrm{DL}}(\Sigma, B \cup \{\mathrm{nb}(\chi)\}, F, \phi)$ returns false, then the other alternative is examined by taking χ as a frozen formula (assuming that χ is in the extension). Then $\mathbf{extensions}_{\mathrm{DL}}$ returns what $\mathbf{extensions}_{\mathrm{DL}}(\Sigma, B, F \cup \{\chi\}, \phi)$ returns.

Assuming that the functions **expand** and **conflict** satisfy the conditions (E1–E4, C1–C2), it can be proved by induction on the number of formulae in $\mathrm{NAnt}(\Sigma)$ but not covered by $B \cup F$ that $\mathbf{extensions}_{\mathrm{DL}}$ is *sound* (if it returns true, there is an extension of Σ agreeing with $B \cup F$ for which the function **test** returns true) and *complete* (if there is an extension of Σ agreeing with $B \cup F$ for which **test** returns true, then $\mathbf{extensions}_{\mathrm{DL}}(\Sigma, B, F, \phi)$ returns true). Hence we have the following soundness and completeness theorem.

Theorem 3.1 *Let Σ be a set of rules. Let B be a set of formulae and NB-formulae, F a set of formulae, and B be grounded in Σ. Then* $\mathbf{extensions}_{\mathrm{DL}}(\Sigma, B, F, \phi)$ *returns true iff there exists an extension Δ of Σ which agrees with $B \cup F$ and for which* $\mathbf{test}(\Delta, \phi)$ *returns true.*

The theorem implies that decision procedures for the different reasoning tasks can be obtained by (i) taking the empty set as the common part of the extensions to be constructed and (ii) choosing an appropriate **test** function as shown in the following corollary.

Corollary 3.2 *Let Σ be a set of rules and ϕ a formula.*

1. *If* $\mathbf{test}(\Delta, \phi)$ *returns true for all Δ and ϕ, then* $\mathbf{extensions}_{\mathrm{DL}}(\Sigma, \emptyset, \emptyset, \phi)$ *returns true iff Σ has an extension.*
2. *If* $\mathbf{test}(\Delta, \phi)$ *returns true iff $\phi \in \Delta$, then* $\mathbf{extensions}_{\mathrm{DL}}(\Sigma, \emptyset, \emptyset, \phi)$ *returns true iff there exists an extension of Σ containing ϕ.*
3. *If* $\mathbf{test}(\Delta, \phi)$ *returns true iff $\phi \notin \Delta$, then* $\mathbf{extensions}_{\mathrm{DL}}(\Sigma, \emptyset, \emptyset, \phi)$ *returns true iff there exists an extension of Σ not containing ϕ (and, in other words,* $\mathbf{extensions}_{\mathrm{DL}}(\Sigma, \emptyset, \emptyset, \phi)$ *returns false iff ϕ is contained in every extension of Σ).*

To illustrate the conditions E1–E4, C1–C2 we present the following "minimal" implementation of **expand** and **conflict** satisfying E1–E4, C1–C2.

E-I0 Let $\mathbf{expand}(\Sigma, B, F)$ return $B \cup \{\chi \in \mathrm{NAnt}(\Sigma) \mid \chi$ not covered by $B, \chi \in \mathrm{Dcl}(\Sigma, B^-)\}$.

C-I0 Let $\mathbf{conflict}(\Sigma, B, F)$ return true iff there exists $\mathrm{nb}(\chi) \in B$ such that $\chi \in \mathrm{Dcl}(\Sigma, B^-)$.

Example 3.1 Although the minimal implementation is taken directly from the minimal requirements, the resulting decision method is already quite powerful. Let us use the implementation to decide brave reasoning. Hence, we let $\mathbf{test}(\Delta, \phi)$ return true iff $\phi \in \Delta$.

(i) Consider whether the set of rules Σ in Example 2.1 has an extension containing a formula b. Hence, we examine the execution of $\textbf{extensions}_{\mathrm{DL}}(\Sigma, \emptyset, \emptyset, b)$. Here $\textbf{expand}(\Sigma, \emptyset, \emptyset)$ returns $\{a_1, \ldots, a_n\}$ and false is returned by $\textbf{conflict}(\Sigma, \{a_1, \ldots, a_n\}, \emptyset)$. Because $B \cup F = \{a_1, \ldots, a_n\}$ covers $\mathrm{NAnt}(\Sigma) = \{a_1, \ldots, a_n\}$ and $F = \emptyset \subseteq \{a_1, \ldots, a_n\} = B$, we have found a Σ-full set $B^- = \emptyset$ and the corresponding extension is $\mathrm{Dcl}(\Sigma, \emptyset)$. Now $\textbf{test}(\mathrm{Dcl}(\Sigma, \emptyset), b)$ returns false and $\textbf{extensions}_{\mathrm{DL}}(\Sigma, \emptyset, \emptyset, b)$ returns false. Hence the minimal implementation is able to determine that Σ does not have an extension containing b without backtracking.

(ii) Consider whether b follows in the brave sense from

$$\Sigma = \{\mathrm{nb}(a) \hookrightarrow b;\ \mathrm{nb}(b) \hookrightarrow a;\ b \hookrightarrow a\}. \tag{3}$$

Hence, we examine the execution of

$$\textbf{extensions}_{\mathrm{DL}}(\Sigma, \emptyset, \emptyset, b).$$

First $\textbf{expand}(\Sigma, \emptyset, \emptyset)$ returns $\emptyset$ and $\textbf{conflict}(\Sigma, \emptyset, \emptyset)$ returns false. Since $\mathrm{NAnt}(\Sigma) = \{a, b\}$ is not covered by $B \cup F = \emptyset$, we must choose either a or b in order to continue. Let us take a. Then

$$\textbf{extensions}_{\mathrm{DL}}(\Sigma, \{\mathrm{nb}(a)\}, \emptyset, b)$$

is executed and here first $\textbf{expand}(\Sigma, \{\mathrm{nb}(a)\}, \emptyset)$ returns $\{\mathrm{nb}(a), b\}$. Then $\textbf{conflict}(\Sigma, \{\mathrm{nb}(a), b\}, \emptyset)$ returns true as $a \in \mathrm{Dcl}(\Sigma, \{\mathrm{nb}(a)\})$. So $\textbf{extensions}_{\mathrm{DL}}(\Sigma, \{\mathrm{nb}(a)\}, \emptyset, b)$ returns false and

$$\textbf{extensions}_{\mathrm{DL}}(\Sigma, \emptyset, \{a\}, b)$$

is executed. Now $\textbf{expand}(\Sigma, \emptyset, \{a\})$ returns $\emptyset$ and $\textbf{conflict}(\Sigma, \emptyset, \{a\})$ returns false. Because $\mathrm{NAnt}(\Sigma)$ is not covered by $B \cup F = \{a\}$, we must choose b in order to continue. Then

$$\textbf{extensions}_{\mathrm{DL}}(\Sigma, \{\mathrm{nb}(b)\}, \{a\}, b)$$

is executed and here $\textbf{expand}(\Sigma, \{\mathrm{nb}(b)\}, \{a\})$ returns $\{\mathrm{nb}(b), a\}$ and then $\textbf{conflict}(\Sigma, \{\mathrm{nb}(b), a\}, \{a\})$ returns false. Because $\mathrm{NAnt}(\Sigma)$ is covered by $B \cup F = \{\mathrm{nb}(b), a\}$ and $F = \{a\} \subseteq \{\mathrm{nb}(b), a\} = B$, we have found an extension $\mathrm{Dcl}(\Sigma, B^-) = \mathrm{Dcl}(\Sigma, \{\mathrm{nb}(b)\})$ of Σ and $\textbf{test}(\mathrm{Dcl}(\Sigma, \{\mathrm{nb}(b)\}), b)$ is performed but false is returned. Hence

$$\textbf{extensions}_{\mathrm{DL}}(\Sigma, \emptyset, \{a, b\}, b)$$

is executed. Now $\textbf{expand}(\Sigma, \emptyset, \{a, b\})$ returns $\emptyset$ and $\textbf{conflict}(\Sigma, \emptyset, \{a, b\})$ returns false. Because $\mathrm{NAnt}(\Sigma)$ is covered by $B \cup F = \{\mathrm{nb}(b), a\}$ but $F = \{a, b\} \not\subseteq \emptyset = B$, $\textbf{extensions}_{\mathrm{DL}}(\Sigma, \emptyset, \{a, b\}, b)$ returns false and hence $\textbf{extensions}_{\mathrm{DL}}(\Sigma, \emptyset, \emptyset, b)$ returns false. Thus, there is no extension of Σ containing b. ■

4 Optimizations

The basic algorithm divides default reasoning into two major subtasks. One is the *search problem* of examining all the alternatives for full sets. This is implemented using (chronological) backtracking and in the worst case the algorithm can search through 2^n alternatives where n is the number of different NB-formulae in the premises. The other is the *classical reasoning problem* of deciding whether a formula ϕ is in the deductive closure of a set of rules Σ given some formulae (and NB-formulae) as premises L, i.e., whether $\phi \in \mathrm{Dcl}(\Sigma, L)$ holds. The membership in the deductive closure is needed in the functions **expand** and **conflict**.

The two difficult tasks are in accordance with the recent results on the complexity of default reasoning showing that default reasoning is a complete problem with respect to the second level of the polynomial time hierarchy [Gottlob, 1992]. The result implies that there are two orthogonal sources of complexity in default reasoning, too. This suggests that we have not introduced additional sources of computational complexity in the basic algorithm.

In this section we present optimization techniques which lead to more efficient methods for solving the two subtasks. As there is quite a lot of research on classical reasoning, the emphasis is on the search problem. But before moving to it we make a couple of remarks about the classical reasoning problem.

- First, notice that the classical reasoning problem is reducible to deciding logical consequence for the underlying (first-order) logic. Testing the membership in the deductive closure of a set of n rules can be implemented using at most n^2 tests for logical consequence in the following way. First construct the set of rules Σ' (2). Then apply rules in Σ' until no new rule fires as follows.

  $C := L^+$
  repeat
  $\quad C' := C;$
  $\quad C := C \cup \{c \mid a_1, \ldots, a_n \hookrightarrow c \in \Sigma'$ and $a_1 \wedge \cdots \wedge a_n \in \mathrm{Th}(C)\}$
  until $C = C'$

  For the resulting C, $\phi \in \mathrm{Dcl}(\Sigma, L)$ iff $\phi \in \mathrm{Th}(C)$.

- Second, the tests for the membership in the deductive closure have a very regular pattern where the set of premises gradually grows. This pattern can be exploited.
- Third, when dealing with a large set of rules, it is important to develop methods for testing the membership in the closure in a goal-directed way so that only a relevant subset of rules is used.

Now we turn to the search problem. The potential search space is exponential and even for a set of rules with 50 different NB-formulae, its size is over 10^{15}. Hence it is essential that the search space is pruned effectively. In $\textbf{extensions}_{\mathrm{DL}}$ the functions **expand** and **conflict** handle the pruning of the search space.

Expanding B

The function **expand** extends the current common part B of the full sets to be constructed. The more formulae are added, the fewer choice points for backtracking are left; every new formula included to B cuts the remaining search space by half.

Although the implementation E-I0 can reduce the search dramatically like in the case of Example 2.1, its basic weakness is that it cannot detect when a formula cannot be in an extension. An optimized version of the implementation should be able to detect simple case like

function **expand**(Σ, B, F)
repeat

$$
\begin{aligned}
&B' := B;\\
&B := B \cup \{\phi \in \mathrm{NAnt}(\Sigma) \mid \phi \text{ is not covered by } B,\\
&\qquad\qquad\qquad\qquad \phi \in \mathrm{Dcl}(\Sigma, B^-)\};\\
&B := B \cup \{\mathrm{nb}(\phi) \mid \phi \in \mathrm{NAnt}(\Sigma), \phi \text{ is not covered by}\\
&\qquad\qquad B, \phi \notin \mathrm{Dcl}(\Sigma, \mathrm{nb}(\mathrm{NAnt}(\Sigma) - (B \cup F)))\}
\end{aligned}
$$

until $B = B'$;
return B.

Figure 2: Implementation E-I1 of **expand**.

if $\Sigma = \{a_0 \hookrightarrow a_1;\ \mathrm{nb}(a_1) \hookrightarrow a_0;\ \mathrm{nb}(b) \hookrightarrow a_0\}$, then b is in no extension of Σ and $\mathrm{nb}(b)$ can be included in B. But also more complicated situations should be covered. For example, consider the set of rules (3) in Example 3.1 and a situation where we are looking for an extension containing a, i.e., $F = \{a\}$. It is evident that no such extension of Σ contains b and $\mathrm{nb}(b)$ could be deduced.

In [Niemelä, 1994] a technique is presented for determining that a formula cannot be in an extension. An attractive feature of this approach is that only a polynomial number of logical consequence tests are needed. We adopt this technique which is based on the following idea. We use as an upper bound for the formulae included in any extension of Σ agreeing with $B \cup F$ the deductive closure of Σ when assuming every NB-formula not excluded by $B \cup F$, i.e., the upper bound is $\mathrm{Dcl}(\Sigma, \mathrm{nb}(\mathrm{NAnt}(\Sigma) - (B \cup F)))$. Proposition 4.1 shows that the formulae not in the upper bound cannot be in any extension of Σ agreeing with $B \cup F$.

Proposition 4.1 *Let Δ be an extension of Σ and let $B \cup F$ agree with Δ. Then $\phi \in \Delta$ implies $\phi \in \mathrm{Dcl}(\Sigma, \mathrm{nb}(\mathrm{NAnt}(\Sigma) - (B \cup F)))$.*

Proof. Let $\phi \in \Delta$. Then by Proposition 2.1 $\phi \in \mathrm{Dcl}(\Sigma, \mathrm{nb}(\mathrm{NAnt}(\Sigma) - \Delta))$. If $\chi \in \mathrm{NAnt}(\Sigma) - \Delta$, then $\chi \notin B \cup F$ since Δ agrees with $B \cup F$. Hence, $\mathrm{nb}(\mathrm{NAnt}(\Sigma) - \Delta) \subseteq \mathrm{nb}(\mathrm{NAnt}(\Sigma) - (B \cup F))$ which implies $\phi \in \mathrm{Dcl}(\Sigma, \mathrm{nb}(\mathrm{NAnt}(\Sigma) - (B \cup F)))$. $\square$

A new implementation E-I1 of **expand** including this technique is described in Figure 2. This implementation is able to handle also the more complicated situations and, e.g, for the set of rules Σ (3), $B = \emptyset$, and $F = \{a\}$ this implementation returns $\{\mathrm{nb}(b), a\}$ as the new set B. It is quite simple to prove that this implementation of **expand** satisfies the conditions E1–E4. A critical point is the condition E3, which can be shown by Proposition 4.1.

Proposition 4.2 *The implementation E-I1 of* **expand** *satisfies the conditions E1–E4.*

Detecting Conflicts

The function **conflict** tries to detect conflicts, i.e., situations where there is no extension agreeing with $B \cup F$. Every time a conflict is found a part of the search space is removed and the earlier the conflict is detected the larger part is eliminated.

The conflict test in the implementation (C-I0) is quite weak in detecting conflicts. Consider, e.g., a set of rules $\Sigma = \{\mathrm{nb}(p) \hookrightarrow q;\ p \hookrightarrow q;\ \mathrm{nb}(q) \hookrightarrow r\}$ and let $B = \{\mathrm{nb}(q)\}$ and $F = \{p\}$. Then there is a conflict since any extension agreeing with $B \cup F$ contains q. In [Niemelä, 1994] a conflict test which detects such conflicts is proposed. The idea is to exploit the frozen formulae F in detecting conflicts. The same idea can be used here: if there exists $\mathrm{nb}(\phi) \in B$ such that $\phi \in \mathrm{Dcl}(\Sigma, B \cup F)$, there is a conflict and **conflict**(Σ, B, F) can return true.

We can strengthen the conflict test on the basis of the following observation: there is a conflict if a frozen formula $\phi \in F$ cannot belong to any extension of the premises agreeing with $B \cup F$. Consider, e.g., a set of rules $\Sigma = \{\mathrm{nb}(p) \hookrightarrow q;\ \mathrm{nb}(q) \hookrightarrow r\}$ and let $B = \emptyset$ and $F = \{p, q\}$. There is a conflict since no extension of Σ agreeing with $B \cup F$ contains q. This kind of conflicts can be detected using the aforementioned notion of an upper bound for extensions. The following implementation includes both optimizations for conflict detection.

C-I1 Let **conflict**(Σ, B, F) return true iff for some $\mathrm{nb}(\phi) \in B$, $\phi \in \mathrm{Dcl}(\Sigma, B \cup F)$ or for some $\phi \in F$, $\phi \notin \mathrm{Dcl}(\Sigma, \mathrm{nb}(\mathrm{NAnt}(\Sigma) - (B \cup F)))$.

Proposition 4.3 *The implementation C-I1 of* **conflict** *satisfies the conditions C1–C2.*

Proof. The condition C1 is clearly satisfied because $\mathrm{Dcl}(\Sigma, B^-) \subseteq \mathrm{Dcl}(\Sigma, B \cup F)$.

(C2) Let Δ be an extension of Σ agreeing with $B \cup F$. We show that **conflict**(Σ, B, F) cannot return true. Let $\mathrm{nb}(\phi) \in B$. Then $\phi \notin \Delta = \mathrm{Dcl}(\Sigma, \mathrm{nb}(\overline{\Delta}))$. Since $B^- \subseteq \mathrm{nb}(\overline{\Delta})$, $\mathrm{Dcl}(\Sigma, B \cup F) \subseteq \mathrm{Dcl}(\Sigma, \mathrm{nb}(\overline{\Delta}) \cup B^+ \cup F)$. Because $B^+ \cup F \subseteq \Delta$, $\mathrm{Dcl}(\Sigma, \mathrm{nb}(\overline{\Delta}) \cup B^+ \cup F) = \mathrm{Dcl}(\Sigma, \mathrm{nb}(\overline{\Delta}))$. So $\mathrm{nb}(\phi) \in B$ implies $\phi \notin \mathrm{Dcl}(\Sigma, B \cup F)$.

If $\phi \in F$, then $\phi \in \Delta$ and by Proposition 4.1 $\phi \in \mathrm{Dcl}(\Sigma, \mathrm{nb}(\mathrm{NAnt}(\Sigma) - (B \cup F)))$. Hence, the condition C2 holds. $\square$

5 Conclusions

In this paper we develop a decision method for default logic which solves the extension existence problem as well as the brave and cautious reasoning problems. It handles the full propositional case and the first-order subclasses of default theories with closed default rules and a decidable fragment of the underlying first-order logic. The method differs from other recent approaches [Junker and Konolige, 1990; Risch and Schwind, 1994; Baader and Hollunder, 1992] to automating default logic in two major respects: (i) in the propositional case it can be implemented in polynomial space and (ii) it does not rely on solutions of subtasks which appear to be computationally harder than the original default reasoning problem. The method partitions default reasoning into two major subtasks: the search problem of examining every alternative for extensions, which is solved by backtracking search, and the classical reasoning task, which can be implemented by a theorem prover for the underlying classical logic. Special emphasis is given to the search problem. The method employs a new compact characterization of extensions based on considering only the justifications of the rules. This reduces the search space for alternatives for extensions. Techniques for pruning the search space are developed. Initial experiments indicate that using the implementations E-I1 and C-I1 of

expand and **conflict** the search space is often kept relatively small and we have been able to handle default theories with a few hundred default rules.

The method developed in this paper is closely related to that presented in [Niemelä, 1994] for autoepistemic reasoning. The key difference is that the novel method uses the new compact characterization of extensions which leads to a smaller initial search space. We exploit the strong groundedness of default extensions which implies that extensions are determined by the justifications of the default rules, while in the earlier approach [Niemelä, 1994] both prerequisites and justifications of the default rules are employed in the characterization. Optimization techniques that prune the search space are discussed already in [Niemelä, 1994]. The technique of expanding the set B (E-I1) is proposed in [Niemelä, 1994] but the conflict detection method (C-I1) is novel.

There are interesting areas for further research. The decision method in this paper uses heavily classical reasoning for pruning the search space. This implies that the development of efficient theorem-proving techniques for implementing the needed classical reasoning in a goal-directed way is important. The potential search space is very large and further work is needed for developing new pruning techniques. The basic algorithm with the requirements for the functions **expand** and **conflict** offers a framework for developing these kinds of optimizations. The decision method can be used in a goal-directed manner by initializing the sets B and F accordingly. For example, if we are interested in extensions containing p but not q, we can start the method with $B = \{\mathrm{nb}(q)\}$ and $F = \{p\}$. An interesting topic for further research is to develop goal-directed techniques where a default reasoning task is analyzed and divided to appropriate subtasks. In the method there is a heuristic choice when a new formula $\chi \in \mathrm{NAnt}(\Sigma)$ not covered by $B \cup F$ is selected. A further area of study is the development of efficient search heuristics.

References

[Baader and Hollunder, 1992] F. Baader and B. Hollunder. Embedding defaults into terminological knowledge representation formalisms. In *Proceedings of the 3rd International Conference on Principles of Knowledge Representation and Reasoning*, pages 306–317, Cambridge, MA, USA, October 1992. Morgan Kaufmann Publishers.

[Ben-Eliyahu and Dechter, 1991] R. Ben-Eliyahu and R. Dechter. Default logic, propositional logic and constraints. In *Proceedings of the 9th National Conference on Artificial Intelligence*, pages 370–385. The MIT Press, July 1991.

[Eiter and Gottlob, 1992] T. Eiter and G. Gottlob. The complexity of logic-based abduction. Technical Report CD-TR 92/35, Christian Doppler Labor für Expertensysteme, Institut für Informationssysteme, Technische Universität Wien, Vienna, Austria, July 1992.

[Etherington, 1987] D.W. Etherington. Relating default logic and circumscription. In *Proceedings of the 10th International Joint Conference on Artificial Intelligence*, pages 489–494, Milan, Italy, August 1987. Morgan Kaufmann Publishers.

[Garey and Johnson, 1979] M.R. Garey and D.S. Johnson. *Computers and Intractability.* W.H. Freeman and Company, San Francisco, 1979.

[Gelfond and Lifschitz, 1990] M. Gelfond and V. Lifschitz. Logic programs with classical negation. In *Proceedings of the 7th International Conference on Logic Programming*, pages 579–597, Jerusalem, Israel, June 1990. The MIT Press.

[Gottlob, 1992] G. Gottlob. Complexity results for nonmonotonic logics. *Journal of Logic and Computation*, 2(3):397–425, June 1992.

[Junker and Konolige, 1990] U. Junker and K. Konolige. Computing the 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, USA, July 1990. The MIT Press.

[Konolige, 1988] K. Konolige. On the relation between default and autoepistemic logic. *Artificial Intelligence*, 35:343–382, 1988.

[Marek and Truszczyński, 1993] W. Marek and M. Truszczyński. *Nonmonotonic Logic: Context-Dependent Reasoning.* Springer-Verlag, Berlin, 1993.

[Niemelä, 1990] I. Niemelä. Towards automatic autoepistemic reasoning. In *Proceedings of the European Workshop on Logics in Artificial Intelligence—JELIA'90*, pages 428–443, Amsterdam, The Netherlands, September 1990. Springer-Verlag.

[Niemelä, 1992] I. Niemelä. A unifying framework for nonmonotonic reasoning. In *Proceedings of the 10th European Conference on Artificial Intelligence*, pages 334–338, Vienna, Austria, August 1992. John Wiley.

[Niemelä, 1994] I. Niemelä. A decision method for nonmonotonic reasoning based on autoepistemic reasoning. In *Proceedings of the 4th International Conference on Principles of Knowledge Representation and Reasoning*, pages 473–484, Bonn, Germany, May 1994. Morgan Kaufmann Publishers.

[Poole, 1988] D. Poole. A logical framework for default reasoning. *Artificial Intelligence*, 36:27–47, 1988.

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

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

[Risch and Schwind, 1994] V. Risch and C. Schwind. Tableau-based characterization and theorem proving for default logic. *Journal of Automated Reasoning*, 13:223–242, 1994.

[Truszczyński, 1991] M. Truszczyński. Embedding default logic into modal nonmonotonic logics. In *Proceedings of the 1st International Workshop on Logic Programming and Non-monotonic Reasoning*, pages 151–165, Washington, D.C., USA, July 1991. The MIT Press.