Hard Problems for Simple Default Logics

Bart Selman
Department of Computer Science
University of Toronto
Toronto, Ontario, Canada M5S 1A4

October 28, 2002

Abstract

Kautz, H.A. and B. Selman, Hard problems for simple default logics, Artificial Intelligence 49 (1991) 243–279.

We investigate the complexity of reasoning with a number of limited default logics. Surprising negative results (the high complexity of simple three literal default rules) as well as positive results (a fast algorithm for skeptical reasoning with binary defaults) are reported, and sources of complexity are discussed. These results impact on work on defeasible inheritance hierarchies as well as default reasoning in general.

1 Introduction

It has been suggested that some kind of default inference can be used to simplify and speed commonsense reasoning. Researchers have appealed to default logics as a solution to the problem of generating and reasoning with large numbers of "frame axioms"; as a way of simplifying complex probabilistic calculations; and recently as a way of "vivifying" (filling out) an incomplete knowledge base, thus suppressing the complexities of reasoning with uncertainty [10, 11].

While current formal theories of default inference are computationally much worse than ordinary logic, it has been tacitly assumed that this additional complexity arises from their use of consistency tests. Our interest in fast, special purpose inference mechanisms led us to investigate very simple propositional, disjunction-free systems of default reasoning, where consistency checking is trivial. Here, we thought, default reasoning should shine.

This paper reports a number of surprising complexity results involving restricted versions of Ray Reiter's default logic [13]. We define a partially-ordered space of propositional default theories of varying degrees of generality. For each we determine the complexity of solving the following three problems: finding an extension; determining if a given proposition is true in some extension; and determining if a given proposition is true in all extensions.

All of these problems are NP-hard for propositional, disjunction-free default logic. This shows that consistency checking is *not* the only source of complexity in default reasoning. We show that a condition called "ordering" (which is related to stratification in logic programming) makes finding an extension tractable. The extension membership problems, however, remain intractable for most of the restricted logics. In particular, these questions are NP-complete for the logic that most naturally represents acyclic inheritance hierarchies. Systems whose rules are similar in form to Horn clauses do admit a tractable algorithm for testing membership in some extension. Finally, we present a polynomial algorithm for testing the membership of a proposition in all extensions of the very restricted class of "normal unary" theories, thus settling an open question in work on inheritance.

The next part of the paper presents general reductions of finding an extension to testing membership in some extension, and that to testing membership in all extensions. This shows that for a large class of default theories, it is at least as hard to test the status of a single proposition as to compute a complete extension.

The final part of the paper provides some intuitive characterizations of the sources of complexity in default reasoning. It suggests that the most efficient use of default information is to "flesh out" the missing detail in a knowledge base in a "brave" manner, a process that corresponds to finding an extension.

A note on notation: throughout this paper, the symbols p, q, r, s, and t are used for propositional letters (also called positive literals). The symbols a, b, c, x, y, and z are used for literals (propositional letters and their negations). The greek letters α, β, and γ are used for formulas. The sign $\sim$ is a meta-language operator that maps a positive literal to a negative literal and vice versa. For example, the expression

$$\sim x \in E$$

where E is a set of literals, is equivalent to the lengthy expression

> if $x = p$ for some letter p, then $\neg x \in E$; otherwise, where $x = \neg p$ for some letter p, it is the case that $p \in E$.

Use of this operator avoids the need to explicitly invoke a rule of negation elimination to convert formulas of the form $\neg\neg p$ to p.

2 Reiter's default logic

Reiter formalized default reasoning by extending first-order logic with default inference rules. This paper will not consider the other nonmonotonic formalisms based on modal logic, circumscription, or model-preference rules, although many of the results it presents have counterparts in those systems. (See [14, 15] for a similar analysis of model-preference theories.)

A *default theory* is a pair (D, W) where D is a set of default rules and W a set of ordinary first-order formulas. This paper examines theories containing

only *semi-normal default rules*, which are of the form

$$\frac{\alpha : \beta \wedge \gamma}{\beta}$$

where α is the *prerequisite*, β the *conclusion*, and $\beta \wedge \gamma$ the *justification* of a rule, each of them formulas. The rule is intuitively understood as meaning that if α is known, and $\beta \wedge \gamma$ is consistent with what is known, then β may be inferred. If γ is missing, then the rule is *normal*. Default rules are sometimes written as $\alpha : \beta \wedge \gamma / \gamma$ for typographic clarity.

An *extension* is a maximal set of conclusions that can be drawn from a theory. But care must be taken that the justification of each rule used in the construction of an extension be consistent with the complete contents of the extension, not just with the non-default information.

Definition:Extension. Where E is a set of formulas, $\mathrm{Th}(E)$ is the deductive closure of E. $\mathcal{E}$ is an extension for the theory (D, W) if and only if it satisfies the following equations:

$$E_0 = W,$$

and for $i > 0$,

$$E_{i+1} = \mathrm{Th}(E_i) \cup \left\{ \gamma \;\middle|\; \frac{\alpha : \beta}{\gamma} \in D, \;\; \alpha \in E_i, \;\; \text{and} \;\; \neg\beta \notin \mathcal{E} \right\};$$

$$\mathcal{E} = \bigcup_{i=0}^{\infty} E_i.$$

Note the explicit reference to $\mathcal{E}$ in the definition of E_{i+1}. A theory can have several, one, or no extensions.

Although normal theories have a number of nice theoretical and computational properties, semi-normal rules are often needed to establish a priority among the defaults. For example, two default rules may have conflicting conclusions, yet have their preconditions satisfied in the same situation. If normal rules were used, this kind of situation would lead to two different extensions. One may know, however, that the first rule should always take priority over the second when both apply. This can be encoded by adding the negation of the precondition of the first rule to the justification of the second rule. Formally, given rules δ_1 and δ_2, where

$$\delta_1 = \frac{\alpha_1 : \beta_1 \wedge \gamma_1}{\beta_1}, \qquad \delta_2 = \frac{\alpha_2 : \beta_2 \wedge \gamma_2}{\beta_2},$$

in order to establish δ_1 as being of higher priority than δ_2, replace δ_2 by δ_2':

$$\delta_2' = \frac{\alpha_2 : \beta_2 \wedge \gamma_2 \wedge \neg\alpha_1}{\beta_2}.$$

One kind of priority that this scheme can encode is the "specificity" ordering that intuitively should appear in an inheritance hierarchy. For example, W may

include the fact that "penguins are birds",[1] and D defaults that assert that penguins don't fly, and that birds do fly. The first, more specific default can be given priority over the second by encoding the pair as

$$\frac{Penguin : \neg Fly}{\neg Fly}, \qquad \frac{Bird : Fly \wedge \neg Penguin}{Fly}.$$

3 Complexity

Following [8], we shall refer to a problem class as "tractable" if a polynomial-time algorithm can solve all its instances. It is not yet possible to prove that any of the problem classes considered in this paper require exponential time, but many are as hard as any solvable in polynomial time by a nondeterministic computer. Such "NP-hard" problems are considered to be intractable.

This paper only considers worst-case complexity. Since the problem instances that cause a particular algorithm to run the longest time may rarely arise, it would be useful to follow this worst-case analysis by some kind of "average-case" analysis. Such an analysis would require some characterization of "average" commonsense theories—a significant task in its own right.

Nonetheless, this worst-case analysis is useful in revealing different sources of complexity in default reasoning, and in providing efficient algorithms for certain problem classes. For example, Section 5 includes a polynomial-time algorithm for computing extensions of the special class of ordered default theories. This algorithm is not necessarily correct for more general problem classes; on the other hand, the obvious general algorithm for computing extensions can take exponential time on an ordered theory. Once tractable algorithms are known for a number of useful classes of default theories, a general algorithm can be constructed that first tests to see if any of the special case algorithms apply, and if none does, invokes the intractable general method.

This paper uses the standard terminology of NP-completeness, which is summarized in Appendix A.

4 A taxonomy of default theories

Two sources of complexity in default theories are readily apparent: the inherent complexity of the first-order component (W), and the complexity of determining whether the justification of a default rule is consistent with the currently-derived set of formulas. We will restrict our attention to finite propositional theories in which W is simply a set (conjunction) of literals. The precondition, justification, and consequence of each default rule is also a conjunction of literals. We will call such a theory "disjunction-free" (abbreviated "DF"). Thus determining whether a default rule is applicable to W is trivial: the precondition must be a

[1] It remains an open problem to determine if a default theory *must* include this assertion, although a survey of the literature lends strong evidence to the conjecture. Certainly it is true that every *paper* on nonmonotonic reasoning must include this example [5].

subset of W, and the intersection of W with the negation of each literal in the justification must be empty. The extended theory is again a set (conjunction) of literals. Although an extension is, by definition, an infinite, deductively closed set of formulas, any extension of a disjunction-free theory is equivalent to the deductive closure of a finite set of literals. Henceforth, when we speak of "computing an extension", we will mean computing such a finite set of literals.

The following functions access the components of default rules of this restricted form.

Definition:pre, concl, just*, just. Where

$$\delta = \frac{a_1 \wedge \cdots \wedge a_l : b_1 \wedge \cdots \wedge b_m \wedge c_1 \wedge \cdots \wedge c_n}{b_1 \wedge \cdots \wedge b_m}$$

and none of the c_i are the same as any of the b_i, let

$$\begin{aligned} \text{pre}(\delta) &= \{a_1, \ldots, a_l\}, \\ \text{concl}(\delta) &= \{b_1, \ldots, b_m\}, \\ \text{just}^*(\delta) &= \{c_1, \ldots, c_n\}, \\ \text{just}(\delta) &= \text{just}^*(\delta) \cup \text{concl}(\delta). \end{aligned}$$

Any inferential power such systems possess resides in the default rules; the only non-default inference rules that apply are negation elimination and conjunction-in and -out (to convert, e.g., $\{\alpha, \beta\}$ to $\alpha \wedge \beta$ and vice versa). The reader should remember, in particular, that because the default rules are in fact rules and not axioms, the principle of reasoning by cases does not apply. For example, given a theory with empty W and rules

$$\frac{p : q}{q}, \qquad \frac{\neg p : q}{q},$$

one may *not* conclude q.

Further restrictions on the form of the default rules leads to the hierarchy shown in Fig. 1. The black arrows lead from the more restricted classes to the more general classes. A negative complexity result (that is, a transformation from an NP-hard problem) for a class in the hierarchy applies also to all elements above it. A positive complexity result (that is, a polynomial-time algorithm) for a class applies also to all classes below it. The classes of theories are as follows:

- *Unary*: These theories restrict the prerequisite to a single letter and the consequence to a single literal. In the case of a positive consequence, the justification may include a single additional negative literal. Unary theories have a simple graphical notation, as shown in Fig. 2. Positive and negative default arcs appear, where optional cancel links may be attached to positive arcs. Note that only positive information enables or cancels the default. Unary theories are a simple example of the kind of graph-based representational systems inspired by Fahlman's work on NETL [7], and are a restricted case of Etherington's "network theories" [6, p. 91].

– *Disjunction-free ordered*: "Ordering" is a syntactic property of default theories developed in [6, p. 86] as a sufficient (but not necessary) condition for a theory to have an extension. The basic idea is to make sure that the application of a default rule can never enable another rule whose conclusion is inconsistent with the justification of the first rule. Formally, given a disjunction-free theory (D, W) and a set *lits* containing the literals in the theory, define $\ll$ and $\underline{\ll}$ to be the smallest relations over *lits* $\times$ *lits* such that

  - $\underline{\ll}$ is reflexive,
  - $\underline{\ll}$ is a superset of $\ll$,
  - $\ll$ and $\underline{\ll}$ are transitive,
  - $\ll$ is transitive through $\underline{\ll}$; that is, for literals x, y, and z in *lits*:

$$[(x \ll y \wedge y \underline{\ll} z) \vee (x \underline{\ll} y \wedge y \ll z)] \supset x \ll z,$$

  - for every $\delta \in D$, and every $a \in \text{pre}(\delta)$, $b \in \text{concl}(\delta)$, and $c \in \text{just}^*(\delta)$:

$$a \underline{\ll} b, \qquad \sim c \ll b.$$

  Then (D, W) is ordered if and only if it contains no literal x such that $x \ll x$.[2] Ordered theories are quite expressive, but as we will see also have some nice computational properties. Later we will describe how ordering is a generalization of the notion of *stratification* in logic programming.

– *Ordered unary*: These theories have no cycles involving cancel arcs, as shown in Fig. 2. Of all the classes considered here, ordered unary theories possess the minimum amount of machinery necessary to represent inheritance hierarchies with some notion of priority between rules.

– *Disjunction-free normal*: Normal theories are formally well-behaved, and possess a resolution-based proof procedure. Normal theories are ordered.

– *Horn*: Horn clause non-default theories have proven useful for applications in databases and expert systems. Satisfiability of propositional Horn clauses can be determined in linear time [3]. Therefore in the search for "easy" default theories it is natural to consider default theories whose rules are similar in form to Horn clauses: the literals in the prerequisite are all positive, and the justification and consequence are the same single literal.

– *Normal unary*: This final category falls in the intersection of all the others. Its graphical representation contains only positive and negative default implication arcs. Normal unary theories can represent inheritance hierarchies with no "preemption strategy" between competing paths [16], but are more general, in that the graph need not be acyclic.

[2] The definition of $\underline{\ll}$ given in [6] does not require that relation to be reflexive or a superset of $\ll$. But the definitions agree on $\ll$, and on whether any particular theory is ordered or not.

Table 1: Forms of default rules in the various classes of theories.

Unary	$p : q/q, \qquad p : q \wedge \neg r/p, \qquad p : \neg q/\neg q,$
Disjunction-free ordered	$a_1 \wedge \cdots \wedge a_l : b_1 \wedge \cdots \wedge b_m \wedge c_1 \wedge \cdots \wedge c_n / b_1 \wedge \cdots \wedge b_m$ and for no literal x is $x \ll x$
Ordered unary	$p : q/q, \qquad p : q \wedge \neg r/p, \qquad p : \neg q/\neg q,$ and for no literal x is $x \ll x$
Disjunction-free normal	$a_1 \wedge \cdots \wedge a_l : b_1 \wedge \cdots \wedge b_m / b_1 \wedge \cdots \wedge b_m$
Horn	$p_1 \wedge \cdots \wedge p_n : q/q$ $p_1 \wedge \cdots \wedge p_n : \neg q/\neg q$
Normal unary	$p : q/q, \qquad p : \neg q/\neg q$

Table 1 summarizes the forms of the rules that appear in each kind of theory. In every case, the elements of a rule are optional. For example, the precondition of a rule may be empty.

5 Finding an extension

It is obvious that the question of whether a first-order default theory has an extension is undecidable, because the question of whether the justification of a rule is consistent with an extension is equally undecidable. In the case of disjunction-free theories, however, this consistency test, as well as the test that the precondition of a rule is satisfied, reduce to simple set operations. Furthermore, the fact that the theories are finite allows an extension to be constructed by the application of one rule at a time. It is straightforward to rewrite the definition of an extension for this special case:

Lemma 1 (Extension of a disjunction-free theory) *Let (D, W) be a disjunction-free default theory. Then $\mathcal{E}$ is an extension of (D, W) if and only if there exists a sequence of rules $\delta_1, \delta_2, \ldots, \delta_n$ from D, and a series of sets $E_0, E_1, \ldots, E_n$ such that for all $i > 0$:*

$$E_0 = W,$$

$$E_i = E_{i-1} \cup \mathrm{concl}(\delta_i),$$

$$\mathrm{pre}(\delta_i) \subseteq E_{i-1},$$

$$\neg\exists c \in \mathrm{just}(\delta_i)\,.\ \sim c \in E_n,$$

$$\neg\exists \delta \in D\,.\quad \mathrm{pre}(\delta) \subseteq E_n \wedge \mathrm{concl}(\delta) \not\subseteq E_n \\ \wedge \neg\exists c \in \mathrm{just}(\delta).\ \sim c \in E_n$$

and $\mathcal{E}$ is the deductive-closure of E_n.

This observation makes it possible to construct a nondeterministic algorithm to decide if a disjunction-free theory has an extension. The machine guesses an extension. It then tries to verify the extension by trying to construct it starting with W, and adding the conclusion of any rule whose precondition is contained in the current approximation and whose justification is consistent with the guessed extension. When the loop halts the guess is correct just in case the final approximation is the same as the extension. The first algorithm in Fig. 3 does just this. It takes as input not only the theory but two additional arguments, *In* and *Out*, which restrict the extensions that can be guessed. The set operations performed in the subroutine **applicable** run in polynomial time, and in the worst case the inner loop cycles $|D|$ times and in each cycle $|D|$ or fewer rules are checked for applicability, so the algorithm also runs in nondeterministic polynomial time. Therefore the extension existence decision problem is in NP.

The second algorithm in the figure actually computes an extension, building it from the conclusions of rules one rule at a time. The *In* parameter of **ND-Exists-Extension-Containing** is passed to the current approximation together with the conclusion of the next rule under consideration. If the answer is "yes" then the conclusion is added to the approximation. The main loop in this algorithm iterates $|D|$ times, thus proving our first theorem:

Theorem 1 *The problem of computing an extension of a disjunction-free default theory (or determining that none exists) is NP-easy.*

So, finding an extension of a DF propositional theory is not harder than the hardest problem in NP. The question then becomes: is there a *deterministic* polynomial algorithm to compute an extension of a disjunction-free theory? Unless P is NP, the answer is no. In fact, 3SAT can be reduced to the extension existence problem for unary theories. Suppose σ is a formula in 3CNF. We can construct a default theory whose extension, if any, is a model of σ. Four sets of rules are needed. The first adds every letter or its negation to the "candidate" extension. The second adds special letters to stand for negative literals, since negative literals cannot appear in the preconditions of rules. The third group checks that every clause is satisfied. If the negation of every literal in some clause is present in the candidate extension, then a special "failure" letter $\mathcal{F}$ is added. The fourth group contains a special "killer" rule. The precondition of this rule is $\mathcal{F}$, but its conclusion, $\mathcal{Z}$, is inconsistent with the justification of the rule which added $\mathcal{F}$. This kind of "vicious cycle" undermines the candidate extension: it can't be a "real" extension after all! Thus, σ is satisfiable if and only if the theory has an extension; that is, when no sequence of applications of default rules can ever conclude $\mathcal{F}$.

The following makes this reduction precise.

Definition:Mappings from 3CNF to defaults. Let σ be a propositional 3CNF formula. The function π maps each positive literal to itself, and each negative literal $\neg p$ to a new letter p'. Consider the following groups of default rules:

(A) for each letter p that appears in σ, the rules:

$$\frac{: p}{p}, \qquad \frac{: \neg p}{\neg p};$$

(B) for each letter p that appears in σ, the rules:

$$\frac{p : \neg p'}{\neg p'}, \qquad \frac{: p' \wedge \neg p}{p'};$$

(C) for each clause $x \vee y \vee z$ of σ, the following three rules, where $\mathcal{F}_{xy}$, $\mathcal{F}_{xyz}$, $\mathcal{F}$, and $\mathcal{Z}$ are new letters:

$$\frac{\pi(\sim x) : \mathcal{F}_{xy} \wedge \neg\pi(y)}{\mathcal{F}_{xy}},$$

$$\frac{\mathcal{F}_{xy} : \mathcal{F}_{xyz} \wedge \neg\pi(z)}{\mathcal{F}_{xyz}},$$

$$\frac{\mathcal{F}_{xyz} : \mathcal{F} \wedge \neg\mathcal{Z}}{\mathcal{F}};$$

(D) the single rule:

$$\frac{\mathcal{F} : \mathcal{Z}}{\mathcal{Z}}.$$

Thus we see that a 3CNF formula is satisfiable if and only if the default theory consisting of an empty W and a D made up of groups (A), (B), (C), and (D) has an extension. This proves the next theorem:

Theorem 2 *The problem of determining whether a unary default theory has an extension is NP-complete. The corresponding problem of computing a set of literals equivalent to an extension (or determining that none exists) is NP-hard.*

As noted earlier, ordered theories cannot fall victim to the kind of vicious cycle used in this reduction. In fact, the extension existence problem is trivial for ordered theories: they always have extensions. One might think that it is possible to construct an extension of an ordered theory by simply applying *any* rule which applies to W, then *any* rule which applies W and the conclusions of the first rule, and so on, until no rules apply. But this is not the case. Consider a theory containing an empty W and just two rules:

$$\delta_1 = \frac{: q \wedge \neg p}{q}, \qquad \delta_2 = \frac{: p}{p}.$$

The rule δ_1 applies to W, but there is no sequence of rule applications beginning with δ_1 that leads to an extension. Intuitively, δ_2 is of higher priority; that rule must be considered for application before δ_1. So what is needed is a way to derive a priority ordering on the *rules* of an ordered theory, given the ordering on its *literals*. The following definition does just that.

Definition:$\prec$ over D. Let (D, W) be a disjunction-free ordered theory, and $\underline{\ll}$ be defined over the literals of the theory as described above. Then for any $\delta_1, \delta_2 \in D$,

$$\delta_1 \prec \delta_2$$

if and only if

$$\exists b \in \text{concl}(\delta_1),\ c \in \text{just}^*(\delta_2)\ .\ b \underline{\ll} \sim c.$$

Lemma 3 in the appendix proves that $\prec$ is in fact a partial order. In the example just given, the theory orders $q \underline{\ll} p$, so that $\delta_2 \prec \delta_1$, as desired. One finds an extension by computing the partial order over the rules, topologically sorting the rules by the order, and then repeatedly firing the lowest ranked rule which is applicable. Figure 4 presents the algorithm, whose proof of correctness appears in Appendix B. The computationally most intensive part of the process turns out to be the transitive closure operation needed to compute $\underline{\ll}$, which requires cubic time. This leads to the following theorem:

Theorem 3 *There is an* $O(n^3)$ *algorithm that finds an extension of a disjunction-free ordered theory, where* n *is the length of theory.*

This result is significant for several reasons. As we noted before, ordered unary theories can represent default inheritance hierarchies, as was demonstrated by [4]. This gives an efficient algorithm for finding some extension, that is, some consistent interpretation, of such inheritance hierarchies. This form of default inheritance has been called "credulous" reasoning by Touretzky et al. [18]. It is of further interest that the efficiency comes from ordering, and *not* from the fact that the theories are unary, *nor* from the fact that inheritance hierarchies are *completely* acyclic. The requirement that the graphical representation of the inheritance hierarchy be acyclic (a condition imposed by Touretzky [19] and followed in the literature ever since) is a sufficient condition for ordering, but is not necessary. For example, the theory containing just the rules

$$\frac{Penguin : \neg Flier}{\neg Flier}, \qquad \frac{Flier : \neg Penguin}{\neg Penguin}$$

is ordered, but would not be admitted by most definitions of an inheritance hierarchy.

This result is also important because of its relation to logic programming. It has been known for some time [2, 12] that stratified logic programs (without "cut") can be mapped into default logic theories, by turning clauses of the form:

$$b \rightarrow a_1, \ldots, a_m, \neg c_1, \ldots, \neg c_n$$

into default rules of the form:

$$\frac{a_1 \wedge \cdots \wedge a_m : \neg c_1, \ldots, \neg c_n}{b}.$$

These rules are not semi-normal, and therefore not ordered. But it is not difficult to show that translation into rules of the form

$$\frac{a_1 \wedge \cdots \wedge a_m : b \wedge \neg c_1 \wedge \cdots \wedge \neg c_n}{b}$$

yields an ordered default theory with the same unique extension. Therefore we also have a polynomial algorithm for propositional stratified logic programming.

Although ordered theories are still quite expressive, some natural situations do map into unordered theories. Consider the "corrupt city government" example illustrated in Fig. 5. We are using default rules to represent the concept "most". This year, most Republican councilmen are running for office, as are most Democratic councilmen. Furthermore, most councilmen running for office are under indictment. The District Attorney is Democratic, and will push the cases against the Republicans much harder than the cases against the Democrats. Therefore most Republican councilmen who are under indictment are *not* running for office. This final condition is most naturally represented by a justification on the default rule for Republicans running for office, that is,

$$\frac{Republican : Running \wedge \neg UnderIndictment}{Running}.$$

The alternative of making "not Republican" a justification on the "under indictment" rule would leave the theory ordered but would change the meaning of the theory. It is easy to verify that there are worlds where most Republicans who are running for office are under indictment, and yet most Republicans who are under indictment are not running for office.

In summary, finding an extension is tractable for ordered theories and intractable for the non-ordered ones considered in this paper, as shown by the top-most horizontal line in Fig. 1. Intractability is caused by the apparent need to consider all possible sequential orderings of rule applications to see if any do not lead to situations where the conclusion of an applicable rule contradicts the justification of a previously applied rule.

6 Membership in some extension

An extension can be thought of as a complete set of beliefs which is consistent with a given set of defaults. Often one is concerned, however, with the status of only a particular proposition. Asking if a proposition p is a member of *some* extension of a theory is equivalent to asking if it is *reasonable* to believe p; that is, whether there is a good argument for p. The same theory may provide good arguments for both p and $\neg p$; but the complementary literals must appear in different extensions.

Reiter [13] showed that p holds in some extension of a normal theory just in case there is a top-down default proof of p. (A top-down default proof is, roughly, a sequence of non-default proofs; the first proves the goal given W and the conclusions of some set of the default rules; the next proves the antecedents

of those defaults, perhaps given the conclusions of another set of default rules; and so on, until a proof that only depends on W is reached.) As we noted above, Touretzky's notion of "credulous" reasoning is similar to finding an extension; he has no notion similar to determining membership in some extension.

The nondeterministic algorithm given in Fig. 3 that solves the generalized version of the extension existence decision problem also solves this problem. The function call

$$\textbf{ND-Exists-Extension-Containing}(\{x\}, \emptyset, D, W)$$

returns "yes" whenever x appears in some extension . Thus:

Theorem 4 *The problem of determining if a given literal appears in some extension of a disjunction-free theory is in NP.*

One might think that checking the status of a single literal is easier than computing an entire extension. Unfortunately, this is not the case. Default logic is "non-local" in the sense that to determine the status of any proposition, one must consider all interactions between all rules and axioms. Is the problem then of equivalent complexity to computing an extension? Surprisingly, the answer is again in general no. While finding an extension is tractable for ordered theories, determining membership in some extension is NP-complete. In fact, we will prove two stronger results, for two special cases of ordered theories: ordered unary and disjunction-free normal.

First, consider the ordered unary case. We will use a reduction like the one used in the proof of Theorem 2 above, but will eliminate the "killer" rule (D), which makes the theory unordered. Then we add the following rule, which makes sure that an extension contains a new letter $\mathcal{T}$ whenever it does *not* contain the "failure" letter $\mathcal{F}$:

(E)

$$\frac{: \mathcal{T} \wedge \neg \mathcal{F}}{\mathcal{T}}.$$

The reader can verify that the theory generated by applying mappings (A), (B), (C), and (E) to a 3CNF formula σ is ordered unary. Furthermore, σ is satisfiable if and only if this theory has an extension containing $\mathcal{T}$. Thus:

Theorem 5 *Determining if a given literal appears in some extension of an ordered unary theory is NP-complete.*

Next, consider the case of disjunction-free normal theories. Normal theories allow negative literals to appear in the precondition which simplifies the reduction. The default rules in set (A) again are used to guess a truth assignment. A second set of rules checks that each clause in σ is satisfied by the extension:

(F) Let $x_i \vee y_i \vee z_i$ stand for the ith clause of σ. Then for each clause i in σ, the following three rules appear, where T_i is a new letter:

$$\frac{x_i : T_i}{T_i}, \qquad \frac{y_i : T_i}{T_i}, \qquad \frac{z_i : T_i}{T_i}.$$

The third group contains a single rule which simply checks that every clause is simultaneously satisfied; that is, that some extension contains all of the T_i:

(G) where n is the number of clauses in σ, the rule:

$$\frac{T_1 \wedge T_2 \wedge \cdots \wedge T_n : \mathcal{T}}{\mathcal{T}}.$$

A 3CNF clause σ is satisfiable if and only if the theory given by mappings (A), (F), and (G) has an extension containing $\mathcal{T}$. In other words:

Theorem 6 *Determining if a given literal appears in some extension of a disjunction-free normal default theory is NP-complete.*

These reductions demonstrate that in order to determine if a literal appears in some extension it is generally necessary (unless P is NP) to search through *all* possible extensions. This should give pause to those who would consider using default rules to extend ordinary backward-chaining theorem proving, as suggested in Reiter's original paper. Default rules can expand the search space exponentially. If the theorem prover chains backward from the given goal, applying default rules as needed, it can reach a state where some "wrong" default has been applied earlier on, which blocks completion of the proof. The system cannot be sure that there is *no* default proof until it tries all different sequences of the defaults.

Is there any interesting class of default theories which does admit a tractable algorithm? Recall that the preconditions of Horn default rules contain only positive literals. This means that no default rule is enabled by applying a different rule which has a negative conclusion. Therefore, in order to construct a default proof of a positive literal p, you do not need to consider any rules with negative conclusions. Because the justification and conclusions of the remaining rules are all positive, none of them can be mutually inconsistent. It is never necessary to "undo" the application of a default rule during the attempt to prove p. The situation where the literal to be tested is negative differs only in that one also uses a rule whose conclusion is the negative goal literal itself.

The following lemma (whose proof appears in Appendix B) shows how to translate the membership problem for Horn default theories into a deduction problem for a consistent classical Horn theory, but eliminating some of the negative default rules.

Lemma 2 *Where (D, W) is a Horn default theory and x is a literal, let H be the following Horn theory:*

$$H = W \cup \left\{ \alpha \supset y \;\middle|\; \begin{array}{l} \alpha : y/y \in D \text{ and} \\ \sim y \notin W \text{ and} \\ [(y \neq \sim x \text{ and } y \text{ is positive}) \text{ or } y = x] \end{array} \right\}$$

Then x appears in some extension of (D, W) iff $H \vdash x$.

By the results of [3] the problem of determining if a literal follows from a propositional Horn theory can be solved in $O(n)$ time, where n is the length of the theory. The translation can also be done in linear time, so therefore:

Theorem 7 *There is an $O(n)$ algorithm which determines if a given literal appears in any extension of a Horn default theory, where n is the length of the theory.*

Horn default theories may have some practical applications in artificial intelligence, as a language for logic programming with default information. It would be useful to let W contain Horn clauses, instead of simply a set of literals, so that both default and non-default information could be represented. Unfortunately, Stillman [17] shows that this extension makes the membership decision problem NP-complete.

The middle horizontal line in Fig. 1 summarizes the results of this section. Horn and normal unary theories are tractable, and the others intractable.

7 Skeptical reasoning

The final kind of reasoning we examine is determining if a proposition holds in all extensions of a theory. This task has been called "skeptical" reasoning in the inheritance literature [18], because it is the most cautious form of default inference. Intuitively, one may skeptically conclude p only when p appears in all sets of beliefs which are consistent with the default axioms. Skeptical reasoning possesses several attractive properties absent from the other two tasks. First, the set of skeptical conclusions of a theory is closed under ordinary logical deduction, and the composition of this set is fixed for any given theory. This leads to the practical advantage of allowing decomposition in problem solving. For example, a system could employ several processors to compute different parts of the set of skeptical conclusions of a theory in parallel. The answers returned by the processors could be simply conjoined. If the processors were computing what held in different arbitrary extensions, however, it would not make sense to conjoin their answers. Second, the conclusions of skeptical reasoning often match our intuitions more closely than the conclusions reached by the other methods. Consider a case where our default knowledge is truly ambiguous; suppose we believe that berries are by default edible, green fruit is by default poisonous, and we encounter a green berry. It seems more reasonable to

withhold judgement until more information is gathered, rather than jump to an arbitrary conclusion, which could leave us either hungry or poisoned.[3] Finally, skeptical reasoning provides the strongest notion of consistency. If the non-default part of the theory is consistent, then one cannot skeptically conclude both p and $\neg p$. On the other hand, there may be *some* extension containing p, and *some other* extension containing $\neg p$.

Note that skeptical reasoning cannot be defined in terms of the test for membership in some extension; that is, one cannot skeptically affirm p if no extension contains $\neg p$. This is because some extensions may contain neither p nor $\neg p$.

Skeptical reasoning falls in the class co-NP, rather than NP. The nondeterministic algorithm for the generalized extension existence problem solves this problem as well. To determine if every extension of a theory contains a literal x, we ask if there is any which does not contain x. That is, if

$$\textbf{ND-Exists-Extension-Containing}(\emptyset, \{x\}, D, W)$$

returns "yes", then the answer is "no, x does not appear in all extensions".

Theorem 8 *The problem of determining if a given literal appears in every extension of a disjunction-free theory is in co-NP.*

One might expect the complexity results for skeptical reasoning to mirror those for the membership problem. Indeed, just as membership in some extension is NP-complete for ordered unary theories, membership in all extensions is co-NP-complete for those theories. The reduction uses the rules in groups (A), (B), and (C) from the analysis of the extension existence problem. Recall that these rules were set up to assert the "failure" letter $\mathcal{F}$ just in case the potential extension did not satisfy the 3CNF formula σ. In other words, σ is unsatisfiable if and only if $\mathcal{F}$ appears in all extensions of the theory containing just those rules. This shows that:

Theorem 9 *Determining if a given literal appears in every extension of an ordered unary theory is co-NP-complete.*

The analogy between membership in all and in some extensions breaks down, however, when we come to the class of Horn theories. We were able to obtain a polynomial algorithm for testing membership in some extension by throwing out all the default rules with negative conclusions (except those which matched the literal to be tested). This cannot be done when one wants to know if a literal holds in all extensions. We need to consider extensions which contain neither the literal nor its negation; extensions where all proofs of the literal are blocked by the application of rules with negative conclusions. Appendix B includes the proof of the following theorem:

[3] As we will see below, skeptical reasoning is computationally the most demanding form of default reasoning, so in practice one would like to have some idea of the "cost" of jumping to the wrong conclusion in order to be able to decide what default reasoning strategy is most appropriate.

Theorem 10 *Determining if a given literal appears in every extension of a Horn default theory is co-NP-complete.*

Intuitively, it is harder to find extensions which leave the truth value of a letter undecided than it is to find ones which assign it true or false. This theorem also illustrates the tradeoff between "caution" and speed in default reasoning: the most conservative kind of reasoning in default logic is also the most complex. The next section of this paper includes a general proof of this observation.

The difficulty in devising complete and tractable algorithms for this kind of skeptical reasoning has led some researchers to suppose that any formulation of reasoning based on an intersection of extensions is intractable. (In particular, the polynomial form of skeptical reasoning developed in [9] is *not* correct according to an intersection of extensions or expansions semantics. Whether it is correct according to our intuitions is, of course, another matter.) An example which demonstrates this point is a version of the "extended Nixon diamond", shown in Fig. 6. Nixon inherits from "Voter" in all three extensions, but through Republican in one, Quaker in the other, and both in the third. (Note that in the default logic formulation, unlike in Touretzky's "path-based" system, no special status given to the links that lead directly out of a leaf node such as Nixon. Touretzky treats such links as representing strict implications, rather than as defaults.)

This problem and others like it can be encoded entirely in a normal unary theory. We have devised the first sound and complete polynomial algorithm for skeptical reasoning in this logic. We will illustrate the central idea behind the algorithm by first considering the restricted case where W is empty, and the literal to be tested is positive. A polynomial algorithm for this case is given in Fig. 7. To determine if a proposition p holds in all extensions, the algorithm attempts to find a complete set of literals containing $\neg p$ which is consistent with some extension that does not contain p.

The reader may gain some understanding of the **Normal-Unary-All-Extensions-Pos** algorithm by "running" it on the extended Nixon diamond example. The set of rules D, where each proposition is abbreviated by its initial letter, is:

$$\frac{n : r}{r}, \qquad \frac{n : q}{q}, \frac{r : \neg q}{\neg q}, \qquad \frac{q : \neg r}{\neg r}, \frac{r : v}{v}, \qquad \frac{q : v}{v}.$$

Rather than including n in W, we will simply add a default rule which always adds n. Since no rule adds $\neg n$, this yields exactly the same set of extensions,

$$\frac{: n}{n}.$$

We wish to determine if v holds in all extensions. The three extensions of the theory are:

$$\mathcal{E}_1 \equiv \{n, r, q, v\},$$
$$\mathcal{E}_2 \equiv \{n, r, \neg q, v\},$$

$$\mathcal{E}_3 \equiv \{n, \neg r, q, v\}.$$

Therefore we expect the algorithm to return "yes".

The complete set of literals L is initially set to

$$L_1 = \{n, r, q, \neg v\}.$$

L_1 is positive consistent, and all its elements are grounded. However, $(r, \neg v)$ is negative inconsistent, because a rule with precondition r adds v, and no rule whose precondition holds in L adds $\neg v$. So r is replaced by $\neg r$, yielding the next version of L:

$$L_2 = \{n, \neg r, q, \neg v\}.$$

Now the algorithm notices that $(q, \neg v)$ is negative inconsistent, so q is replaced:

$$L_3 = \{n, \neg r, \neg q, \neg v\}.$$

But now $(n, \neg r)$ is negative inconsistent, so n must be replaced by $\neg n$.

$$L_4 = \{\neg n, \neg r, \neg q, \neg v\}.$$

L_4 is not positive consistent, because n is fixed positive. Therefore the algorithm returns "yes"; v holds in all extensions.

An algorithm for the more general case of normal unary skeptical reasoning appears in Fig. 8. It transforms the input theory to the simpler case by replacing literals in W by new rules (as in the example above) and by substituting a new letter for a negative query. A proof of correctness and a complexity analysis of the two algorithms appears in Appendix B. Thus:

Theorem 11 *Given a normal unary theory and a literal x, the* Normal-Unary-All-Extensions algorithm *returns "yes" if and only if every extension of the theory contains x. The time complexity of the algorithm is $O(n^2)$, where n is the length of the theory.*

In summary, all kinds of skeptical reasoning other than for normal unary theories are intractable, as shown by the lowest horizontal line in Fig. 1. Stein [16] uses normal unary theories to capture so-called "ideally skeptical" inheritance, which is default inheritance without priorities. Thus, our algorithm can be directly applied to handle this, very conservative, form of inheritance. It remains to be seen if there are interesting practical applications of this kind of reasoning.

8 Comparing the reasoning tasks

The complexity results for the specific classes of default theories we considered showed that the task of finding an extension is no harder than determining if a literal holds in some extension of a theory, which in turn is no harder than skeptical reasoning. This section develops general results that show this is true

for very broad classes of disjunction-free theories. (These general theorems do not make the previous results redundant; the results limited to the specific classes are stronger.)

First we compare finding an extension to determining if a literal holds in some extension. The first algorithm presented in Fig. 9 reduces the former problem to the latter. The algorithm works by maintaining an approximation to an extension in the variable E. It creates a new default rule whose precondition is the conjunction of all the literals in E, and whose conclusion is a new letter p. The oracle **Some-Extension** determines that some extension of the original theory augmented with this new rule contains p just when some extension of the original theory contains all of E. The main loop of the algorithm makes E maximal, by trying to add each literal to it. Note that a disjunction-free normal default rule is added to the theory. A requirement of this reduction is therefore that the class of default theories under consideration be closed under the addition of such rules. Note that this reduction is not too surprising given our earlier result which showed that the membership question is NP-complete for disjunction-free normal theories.

Theorem 12 *For any class of disjunction-free theories that is closed under the addition of a single disjunction-free normal default rule, the problem of finding an extension is Turing-reducible to the problem of determining whether a given literal appears in some extension.*

The second algorithm in Fig. 9 reduces membership in some extension to membership in all extensions. Where x is the literal to be tested, the reduction adds a new default rule with no precondition whose conclusion is $\sim x$. This rule can only fail to be applied to an extension which contains x. Therefore, we see that some extension of the original theory contains x if and only if it is not the case that every extension of the modified theory contains $\sim x$. This reduction applies to any class of theories closed under the addition of the new default rule, which can be characterized as follows:

Theorem 13 *For any class of default theories that is closed under the addition of a rule of the form $: \sim x / \sim x$, determining if a given literal appears in some extension is Turing-reducible to the task of determining whether a given literal appears in all extensions.*

These reductions can be used to derive complexity results for classes of default theories not specifically examined in this paper. For example, by Theorem 12, any NP-completeness result for the problem of finding an extension will carry over to that of membership in some extension, provided the class of theories under consideration is disjunction-free and closed under the addition of the simple default specified above. Going in the other direction, a polynomial algorithm for finding an extension can be obtained from one for membership in some extension.

The second reduction is even more general, and even applies to infinite and first-order theories. Any lower bound result, such as NP-hardness, PSPACE-

completeness, undecidability, and so on, carries over from the problem of membership in some to membership in all extensions. Likewise, a polynomial algorithm or decision procedure for skeptical reasoning immediately gives one for membership in some extension.

9 Conclusions

We have examined a wide range of simple default theories and have uncovered some surprising worst-case complexity results. The problems of deduction and consistency checking are not the only source of difficulties in reasoning with defaults. In the study of finding an extension, the source of complexity can be characterized as the problem of detecting incoherent cycles in the rules, which make it hard to detect if a sequence of rule applications is actually leading toward an extension. In the membership problems, the source of complexity can be characterized as the exponential number of different extensions that can be generated by a set of defaults. One can think of the rules as specifying a nondeterministic computation, and the test for membership of a literal in some or all extensions as picking out a successful computation, or determining that there are none.

Yet we also developed a number of interesting positive results. We presented a polynomial algorithm to find an extension of a propositional ordered theory, and these theories appear to have many uses in AI and logic programming. In particular, this class includes "credulous" reasoning in default inheritance hierarchies, but is strictly more general, in that rules may have any number of positive literals in the preconditions, and the graphic form of rules may include (certain kinds of) cycles. As noted earlier, the syntactic constraints traditionally placed on inheritance hierarchies do not correspond to the constraints actually needed for efficient reasoning. Ordered theories also subsume stratified logic programs, but allow both negative and positive literals to appear anywhere in a rule.

Horn default theories nicely generalize classical Horn theories and retain linear complexity for the problem of membership in some extension. Finally, we developed the first polynomial algorithm for determining the contents of the intersection of all extensions of a default theory—albeit a very restricted class of theories.

Last but not least, the general reductions between the reasoning tasks suggest how default logic may be most efficiently used in problem solving. The riskiest, most credulous form of default reasoning is also the fastest. At least in the propositional case, it is possible to construct an efficient problem solver which simply applies all its default rules to an input problem description, forward-chaining to a complete extension. The abundance of detail in the extension would, one might hope, simplify or trivialize further inference. This is the use of defaults suggested by Levesque in his proposal for "vivid reasoning" [10].

Reiter identified the notion of a "default logic proof" with determining if a formula held in some extension of a theory. The much greater complexity of the

problem of determining membership in some extension over finding an arbitrary extension indicates that great care will be needed in augmenting traditional backward-chaining theorem provers with default rules, in order to not fall victim to an exponential expansion of the search space.

Finally, the most conservative use of default logic, skeptical reasoning, may prove too computationally intensive for any application. If default logic is your tool of choice, and you are concerned with the complexity of inference, it appears that you should design your theory so that any extension in fact yields a reasonable set of conclusions, and you should not depend on taking an intersection of extensions to filter out the good from the bad.

A Terminology of NP-completeness

For an introduction to the theory of NP-completeness, see [8]. The class *NP* consists of decision problems (ones whose solution is either "yes" or "no") that can be solved by a nondeterministic algorithm that given a problem instance (1) guesses a data structure and (2) deterministically checks in polynomial time whether the answer is "yes" or "no". The algorithm is said to solve the problem if and only if for any "yes"-instance of the decision problem, there exists a data structure that leads to a "yes" answer after checking; whereas for any "no"-instance of the problem, no such structure exists. An *NP-complete* problem is a member of an NP to which *any* problem in NP can be transformed in polynomial time. A problem is in *co-NP* if its complement is in NP, and any such problem can be transformed in polynomial time into a *co-NP-complete* problem. A problem (not necessarily a decision problem) is *NP-hard* if an NP-complete problem can be solved by a deterministic polynomial algorithm that employs an oracle that solves the NP-hard problem. Conversely, a problem is *NP-easy* if it can be solved by a deterministic polynomial algorithm that employs an oracle that solves a problem in NP. Hence an NP-easy problem is essentially "no harder" than any problem in NP. (That is, if P = NP, then any NP-easy problem is polynomial. But in terms of the complexity hierarchy, NP-easy properly includes both NP and co-NP.)

The NP-complete problem *3SAT* is that of determining the satisfiability of a conjunction of three-element clauses in propositional logic (*3CNF*); that is, of a formula of the form:

$$\sigma = (x_1 \vee y_1 \vee z_1) \wedge (x_2 \vee y_2 \vee z_2) \wedge \cdots .$$

The corresponding co-NP-complete problem is determining the unsatisfiability of such a formula.

B Proofs

Proof: Proof of Lemma 1(Sketch). It is easy to see that the lemma's definition of an extension is equivalent to the original definition (presented in Section 2) if

the logical closure operation is applied to the E's on the right-hand side of the expressions in the lemma. For example, $\sim c \in E_n$ becomes $\sim c \in \mathrm{Th}(E_n)$. The original definition allows the application of a number of defaults at each step, while the lemma effectively stretches the applications out into a single sequence. The difference is not significant for finite propositional theories. (The lemma fails for infinite theories, because there may not be any maximal n.)

Let E be a set of literals and x a single literal. Observe that if E is consistent, then $x \in E$ if and only if $x \in \mathrm{Th}(E)$. Furthermore, if W is a consistent set of literals, then applications of semi-normal default rules will maintain consistency. Therefore the definitions are equivalent for consistent disjunction-free theories. On the other hand, if W is an inconsistent set of literals, then both by the lemma and by the original definition the inconsistent theory $\mathrm{Th}(W)$ is the unique extension. So the definitions are fully equivalent for disjunction-free theories.

Note that the lemma fails for non-semi-normal finite propositional theories containing no disjunctions. This is because the application of a non-semi-normal default rule to a consistent set of literals can yield an inconsistent set. *Q.E.D.*

Lemma 3 *The relation $\prec$ as defined over D in a disjunction-free ordered theory is a partial order.*

Proof: We show that $\prec$ is transitive and irreflexive over D.

(*transitive*) Suppose $\delta_1 \prec \delta_2$ and $\delta_2 \prec \delta_3$. Then in must be the case that

$$\exists b_1 \in \mathrm{concl}(\delta_1), c_2 \in \mathrm{just}^*(\delta_2) \,.\, b_1 \underline{\ll} \sim c_2,$$

$$\exists b_2 \in \mathrm{concl}(\delta_2), c_3 \in \mathrm{just}^*(\delta_3) \,.\, b_2 \underline{\ll} \sim c_3.$$

The rule δ_2 induces the literal ordering

$$\sim c_2 \ll b_2.$$

So $b_1 \underline{\ll} \sim c_3$, which entails that $\delta_1 \prec \delta_3$.

(*irreflexive*) Suppose it were the case that $\delta \prec \delta$. Then it must be the case that

$$\exists b \in \mathrm{concl}(\delta),\ c \in \mathrm{just}^*(\delta) \,.\, b \underline{\ll} \sim c.$$

But this rule induces the literal ordering

$$\sim c \ll b$$

which would imply that $b \ll b$, violating the definition of an ordered theory.

Proof: Proof of Theorem 3

(*correctness*) Let E_{FINAL} be the value returned by **Ordered-Find-Extension**(D, W). We claim that the following assertion is true at the end of the **then** clause in the algorithm:

$$\neg\exists c \in \mathrm{just}(D[i]) \,.\, \sim c \in E_{\mathrm{FINAL}}.$$

Correctness of the algorithm follows immediately from this assertion and Lemma 1.

So suppose the assertion were false. Let E_j represent the value of the variable E after cycle j of the outer loop, and $D[i_j]$ be the rule selected by the **if** statement in cycle j. Suppose the assertion fails when $j = j_0$. Plainly $c \notin \text{concl}(D[i_{j_0}])$, so

$$c \in \text{just}^*(D[i_{j_0}])$$

and there must be some $j_1 > j_0$ such that for $j = j_1$,

$$\sim c \in \text{concl}(D[i_{j_1}]).$$

Then by the definition of $\prec$,

$$D[i_{j_1}] \prec D[i_{j_0}]$$

because $\sim c \underline{\ll} \sim c$. This implies that

$$i_{j_1} \ll i_{j_0}.$$

Now suppose that $\text{pre}(D[i_{j_1}]) \subseteq E_{(j_0-1)}$. Then the inner loop in cycle j_0 should choose $D[i_{j_1}]$ rather than $D[i_{j_0}]$. But since this is not the case, there must exist some literal a such that

$$a \in \text{pre}(D[i_{j_1}]), \qquad a \notin E_{(j_0-1)}.$$

This a must have been added in either cycle j_0 or in some cycle which follows j_0. First, consider the possibility that a is added in cycle j_0. In that case, $a \in \text{concl}(D[i_{j_0}])$ and thus $\sim c \ll a$. Also, since $a \in \text{pre}(D[i_{j_1}])$ and $\sim c \in \text{concl}(D[i_{j_1}])$, we have that $a \underline{\ll} \sim c$. Thus, $\sim c \ll \sim c$, which contradicts the fact that our theory is ordered. Therefore, a must have been added in some cycle j_2 which follows j_0 and precedes j_1:

$$j_0 < j_2 < j_1.$$

Note that $D[i_{j_2}] \prec D[i_{j_0}]$ because

$$a \underline{\ll} \sim c, \qquad a \in \text{concl}(D[i_{j_2}]), \qquad c \in \text{just}^*(D[i_{j_0}]).$$

Now by the previous argument $\text{pre}(D[i_{j_2}]) \not\subseteq E_{(j_0-1)}$, so there must be some literal a' such that

$$a' \in \text{pre}(D[i_{j_2}]), \qquad a' \notin E_{(j_0-1)}.$$

Again, we can show that a' cannot have been added in cycle j_0. For suppose it was. Then, $a' \in \text{concl}(D[i_{j_0}])$ and thus $\sim c \ll a'$. Also, since $a' \in \text{pre}(D[i_{j_2}])$ and $a \in \text{concl}(D[i_{j_2}])$, we have that $a' \underline{\ll} a$, and, again from $D[i_{j_1}]$, we have $a \underline{\ll} \sim c$. Thus, $\sim c \ll \sim c$, which contradicts the fact that our theory is ordered.

Therefore, a must have been added by some rule $D[i_{j_3}]$ which fires at a cycle j_3, where

$$j_0 < j_3 < j_2 < j_1.$$

As before, $D[i_{j_3}] \prec D[i_{j_0-1}]$ and $\mathrm{pre}(D[i_{j_3}]) \not\subseteq E_{j_0}$. The argument can be repeated any number of times, leading to an infinite sequence

$$j_0 < \cdots < j_4 < j_3 < j_2 < j_1.$$

But since there are a finite number of rules in D, this is impossible.

(*complexity*) Let us suppose that the propositional letters of the input are represented by the odd integers 1 through $m-1$, and the corresponding negative literals by the integers 2 through m. Note that $n \leq m \leq 2n$, where n is the length of the theory. The variable E is represented by a vector of length m, with $E[i] = 1$ when the literal represented by i is in E. The precondition, justification (just*), and conclusion of each default rule is stored as a list of integers.

The first task is to compute the ordering $\underline{\ll}$ on the literals. This relation can be stored in an $m \times m$ table, with entry (i, j) equal to 1 just in case literal $i \mathrel{\underline{\ll}}$ literal j. The table is first set to all 0's except for the diagonal (i, i) which is set to all 1's (because $\underline{\ll}$ is reflective), in $\mathrm{O}(m^2)$ steps. Next the constraints derived from each default rule ($a \mathrel{\underline{\ll}} b$ and $\sim c \ll b$) initialize the table. The constraints on $\underline{\ll}$ induced by each rule δ can be calculated in $|\delta|^2$ time, so the initialization step requires the following time:

$$\sum_{i=1}^{|D|} |\delta_i|^2 \leq \left(\sum_{i=1}^{|D|} |\delta_i| \right)^2 \leq n^2.$$

Finally the transitive closure of $\underline{\ll}$ is taken in $\mathrm{O}(m^3)$ time [1]. Thus this task requires $\mathrm{O}(m^2 + n^2 + m^3) = \mathrm{O}(n^3)$ time.

The next task is to compute the ordering $\prec$ on default rules. For each pair of rules, compare each literal in the conclusion of the first rule with each literal in the proper justification (just*) of the second rule. This takes the following time:

$$\begin{aligned} \sum_{i,j=1}^{|D|} |\delta_i||\delta_j| &= \sum_{i=1}^{|D|} |\delta_i| \left(\sum_{j=1}^{|D|} |\delta_j| \right) \\ &\leq \sum_{i=1}^{|D|} |\delta_i| n = n \sum_{i=1}^{|D|} |\delta_i| \leq n^2. \end{aligned}$$

The result of this task is a list of length less than $|D|^2$ describing the relation $\prec$.

Now we come to the proper algorithm. The topological sort of D is linear in the number of rules plus the number of pairs describing $\underline{\ll}$. So the sort is $\mathrm{O}(|D| + |D|^2)$ or more simply $\mathrm{O}(n^2)$.

Checking that rule $D[i]$ is applicable to E takes $|D[i]|$ time. Checking all the rules to find an applicable one takes

$$\sum_{i=1}^{|D|} |D[i]| \leq n$$

steps. Each rule applies at most once, so this check has to be performed at most $|D|$ times. Therefore all the calls to **applicable** require $O(|D|n) = O(n^2)$ time. Each union of E with $\text{concl}(D[i])$ also takes $|D[i]|$ time, and again this step is performed at most $|D|$ times, so again the time is $O(n^2)$.

The total time for the algorithm is therefore $O(n^3 + n^2 + n^2) = O(n^3)$. It is interesting to note that the most expensive part of the algorithm is taking the transitive closure of the literal ordering.$Q.E.D.$

Proof: Proof of Lemma 2

($\Rightarrow$) Suppose x appears in some extension $\mathcal{E}$ of (D, W). By Lemma 1 there are a finite number of approximations E_i to $\mathcal{E}$. Let E_j be the lowest numbered approximation such that $x \in E_j$. If $j = 0$ then $x \in W$ so of course $H \vdash x$. Otherwise $x \in \text{concl}(\delta_j)$ which implies that $\text{pre}(\delta_j) \supset x \in H$. It is apparent from the construction of E_j that one can extract a forest of default rules all with positive conclusions rooted at $\text{pre}(\delta_j)$ and with leaves in W. All of the Horn clauses corresponding to these rules must be in H as well. Then this forest together with δ_j constitutes a proof of x from H.

($\Leftarrow$) Suppose $H \vdash x$. Note that H must be consistent; therefore there exists a linear resolution style proof tree T of x from H. Traverse T in an order with visits a node after visiting all of its children; the result is a linearization of the Horn clauses used in the proof of x. Eliminate any clause which appears in W or earlier in the sequence. Replace each clause by the corresponding default rule which generates it. It is clear then that the resulting sequence is a prefix of a sequence of rules whose application to W leads to an extension containing $x.Q.E.D.$

Proof: Proof of Theorem 10 An arbitrary 3CNF formula σ is unsatisfiable if and only if $\mathcal{F}$ holds in every extension of the theory containing rules in groups (H), (I), and (J) below:

(H) for each letter p which appears in σ, with $p' = \pi(p)$, the four rules:

$$\frac{: p}{p}, \qquad \frac{: p'}{p'}, \qquad \frac{p' : \neg p}{\neg p}, \qquad \frac{: \neg p'}{\neg p'};$$

(I) likewise for each letter p, the following rule, where $\mathcal{F}$ is a new letter:

$$\frac{p \wedge p' : \mathcal{F}}{\mathcal{F}};$$

(J) for each clause $x \vee y \vee z$ of σ, the rule:

$$\frac{\pi(\sim x) \wedge \pi(\sim y) \wedge \pi(\sim z) : \mathcal{F}}{\mathcal{F}}.$$

We prove the equivalent statement, that σ is satisfiable if and only if some extension does not contain $\mathcal{F}$.

(*if*) Let $\mathcal{E}$ be an extension not containing $\mathcal{F}$. By the rules in group (H), for any letter p, every extension contains either $\{p, p'\}$, $\{p, \neg p'\}$, or $\{\neg p, p'\}$. Since none of the rules in group (I) applied in $\mathcal{E}$, the first alternative never occurred for any p. Thus p' can be taken to stand for $\neg p$. Thus the fact that no rule in group (J) could have applied in $\mathcal{E}$ means that one of the literals in each clause of σ appeared in $\mathcal{E}$. So $\mathcal{E}$ is a model for σ, and σ is satisfiable.

(*only if*) Suppose M is a truth assignment for σ. Let $\mathcal{E}$ be the deductive closure of the set of literals which hold in M, together with p' or $\neg p'$ for every literal $\neg p$ or p respectively which holds in M. Then $\mathcal{E}$ is an extension of the default theory. Note that $\mathcal{E}$ is grounded by the rules in groups (H), and that none of the rules in groups (I) or (J) apply. In particular, $\mathcal{F}$ does not appear in $\mathcal{E}$.*Q.E.D.*

Proof: Proof of Theorem 11 The correctness proof of the **Normal-Unary-All-Extensions-Pos** algorithm is based on the following loop invariant:

Lemma 4 *Given a set of normal unary defaults D and a positive literal p_k, the following property is maintained each time through the* while *loop in* **Normal-Unary-All-Extensions-Pos(p_k,D)**:

> INV: *If L contains $\neg q$ and $\mathcal{E}$ is an extension of D that does not contain p_k, then $\mathcal{E}$ does not contain q.*

Proof:(By induction on the number of times through the loop).

Base case (upon entering the **while** statement). $L = \{p_1, p_2, ..., \neg p_k, ..., p_n\}$. The only negative literal in L is $\neg p_k$. So, INV holds.

Induction step. Let L be the complete set of literals after l times through the body of the loop, and L' the updated L after one additional time through. By the induction hypothesis, L has property INV. Clearly, if the condition in the **if** statement is false, we have $L' = L$. And thus, INV holds for L'. Otherwise, $L' = (L - \{p\}) \cup \{\neg p\}$ where p in L is such that (1) *not* grounded(p,L, D) or (2) neg-inconsistent(p,$\neg q$,L, D) for some $\neg q$ in L. We will now show by contradiction that INV holds for L'. Assume that L' does not satisfy INV, i.e., there exists an extension $\mathcal{E}^\star$ of D that does not contain p_k but does contain some letter s such that $\neg s$ in L'. If $s \neq p$, then INV would not hold for L either, violating the induction hypothesis. So, $s = p$.

Case 1. *not* grounded(p,L,D). Since p in $\mathcal{E}^\star$, there must exist a sequence of one or more rules that adds p to the extension, i.e., there exists a sequence $q_0, q_1, ..., q_m = p$ such that

(a) q_j in $\mathcal{E}^\star$, $0 \leq j \leq m$,

(b) : q_0/q_0 in D, and
(c) $p_{j-1} : p_j/p_j$ in D, $1 \leq j \leq m$.
Now, by the induction hypothesis, we have p_j in L with $0 \leq j \leq m$, since these positive literals are in an extension $\mathcal{E}^\star$ of D that does not contain p_k. Therefore, grounded(p,L,D). Contradiction.

Case 2. neg-inconsistent(p,$\neg q$,L,D). From neg-inconsistent(p,$\neg q$,L,D), it follows that
(a) p and $\neg q$ in L,
(b) $p : q/q$ in D,
(c) $:\neg q/\neg q$ not in D, and
(d) for each rule $r : \neg q/\neg q$ in D, we have $\neg r$ in L.
By the induction hypothesis it follows that $\mathcal{E}^\star$ does not contain q (since L contains $\neg q$), and neither does $\mathcal{E}^\star$ contain an r with a rule $r : \neg q/\neg q$ in D (since, if $\mathcal{E}^\star$ would contain such an r then, by the induction hypothesis, L contains r and therefore (d) is false). Contradiction. Moreover, since : $\neg q/\neg q$ not in D, it follows that $\mathcal{E}^\star$ does not contain $\neg q$ either. Now, since p in $\mathcal{E}^\star$, it follows that the rule $p : q/q$ in D is applicable. But since neither q nor $\neg q$ is in $\mathcal{E}^\star$, $\mathcal{E}^\star$ violates the fixed point property of a default logic extension [13, Theorem 2.5]. So, no such $\mathcal{E}^\star$ exists. Contradiction.

From the base case and the induction step, it follows by finite induction that INV is maintained.

We will now prove the correctness of the **Normal-Unary-All-Extensions-Pos** algorithm.

Lemma 5 *Given a set of normal unary defaults D and a positive literal p_k, the algorithm* Normal-Unary-All-Extensions-Pos *returns "yes" iff every extension of D contains p_k. The time complexity of the algorithm is* $O(n^2)$*, where n is the length of D.*

Proof: First, we will show that:

The algorithm returns "yes" iff every extension of D contains p_k.

($\Rightarrow$) By contradiction. Assume the algorithm returns "yes" while there exists some extension $\mathcal{E}^\star$ of D that does not contain p_k. Note that L is not pos-consistent upon exiting. So, there exists some q such that $\neg q$ in L with fixed-pos(q,L,D), i.e.,
(a) : q/q in D,
(b) : $\neg q/\neg q$ not in D, and
(c) for each rule of the form $r : \neg q/\neg q$ in D, we have $\neg r$ in L.
Also, by Lemma 4, L has the property INV. Now, since $\mathcal{E}^\star$ is an extension of D that does not contain p_k, it follows from INV that $\mathcal{E}^\star$ does not contain q. Moreover, by (b), (c), and INV it follows that $\mathcal{E}^\star$ does not contain $\neg q$ either (no rule present or applicable to add $\neg q$ to $\mathcal{E}$). It follows that : q/q in D is applicable. But since $\mathcal{E}^\star$ does not contain q, $\mathcal{E}^\star$ violates the fixed point property of a default logic extension [13, Theorem 2.5]. Therefore, no such $\mathcal{E}^\star$ exists. Contradiction.

($\Leftarrow$) Assume the algorithm returns "no". Therefore, L upon exiting is such that

(a) pos-consistent(L,D),

(b) for each letter q in L, we have grounded(q,L,D),

(c) there do not exist q and $\neg r$ in L such that neg-inconsistent(q,$\neg r$,L,D), and

(d) $\neg p_k$ in L.

We will construct a set of literals from L, and show that the deductive closure of this set is an extension of D that does not contain p_k. Thereby, we will have shown the contrapositive of the $\Leftarrow$-direction.

Let neg-supported($\neg p$,L,D) iff : $\neg p/\neg p$ in D or there exists a q in L and a rule $q : \neg p/\neg p$ in D, and let $\mathcal{E} = \mathrm{Th}(\{p \mid p \text{ in } L\} \cup \{\neg p \mid \neg p \text{ in } L \text{ and neg-supported}(\neg p,L,D)\ \})$. We will now show that $\mathcal{E}$ is an extension of D that does not contain p_k.

First, by (d) and the definition of $\mathcal{E}$ it follows that $\mathcal{E}$ does not contain p_k. From (b) and the definition of $\mathcal{E}$ it also follows that each positive literal in $\mathcal{E}$ is grounded, i.e., for each p in $\mathcal{E}$ there is a sequence of one or more rules starting with a rule of the form : q_0/q_0 that brings in p. Now, consider starting off with the empty set and applying all the rules (and only those) that bring in all positive letters of $\mathcal{E}$, to obtain E_0. Now, by the definition of $\mathcal{E}$ it follows that all negative letters in $\mathcal{E}$ can subsequently be brought in by rules in D that are applicable at E_0. After applying those rules, and only those, in a sequence starting at E_0, we obtain a set E and its deductive closure $\mathcal{E}$. It now remains to be shown that: (1) no subsequent application of rules can undermine the justification of any rule applied so far (i.e., make some previously applied rule non-applicable), and (2) no additional literals can be brought in by any of the rules in D.

Case 1. Follows immediately from the fact that we have only normal defaults in D.

Case 2. By contradiction. Assume more literals can be added to $\mathcal{E}$ by further rule applications. Let r or $\neg r$ be the first such literal that can thus be added. By definition of $\mathcal{E}$ and the fact that L is a complete set of literals, it follows that $\neg r$ must be in L, and therefore, the first new literal that can be brought in must be a positive one, i.e., r (if $\neg r$ could be brought in then we would have neg-supported($\neg r$,L,D) and $\neg r$ would already be in $\mathcal{E}$, contradiction). Note that since r is added by a normal default, $\mathcal{E}$ cannot contain $\neg r$. The literal r can only be brought in via one of the following rule applications:

- Application of a rule : r/r. Since there are no rules to bring in $\neg r$ (from definition of $\mathcal{E}$), we have : $\neg r/\neg r$ not in D, and for each rule $s : \neg r/\neg r$ in D, we have $\neg s$ in L. From : r/r in D it follows that fixed-pos(r,L,D). And thus, we have that L is not pos-consistent. Contradiction with (a).

- Application of a rule $t : r/r$. As argued above, we again have that : $\neg r/\neg r$ not in D and for each rule $s : \neg r/\neg r$ in D, we have $\neg s$ in L. Now, since

$t : r/r$ is applicable in $\mathcal{E}$, t is in $\mathcal{E}$ and thus, t in L. Therefore, we have neg-inconsistent(t,$\neg r$,L,D) for t and $\neg r$ in L. Contradiction with (c).

It follows that $\mathcal{E}$ is a fixed point of the defaults and grounded. So, $\mathcal{E}$ is an extension of D not containing p_k, and thus, it is not the case that every extension of D contains p_k. This completes the correctness proof of the algorithm.

Finally, we will determine the time complexity of the algorithm. Since the number of positive literals in L is decreased by one each time through the body of the **while** loop with the possible exception of the last time through, it follows from the definitions of grounded and neg-inconsistent that the loop body is executed at most N times (N is the number of distinct propositional letters in the theory). Computing pos-consistent(L,D) can be done in O(n), where n is the length of the theory. And, a pair of letters p, $\neg q$ such that (*not* grounded(p, L, D)) or neg-inconsistent($p, \neg q, L, D$)) can also be found in time O(n). Therefore, the time complexity of the **Normal-Unary-All-Extensions-Pos** algorithm is O(n^2).*Q.E.D.*

Proof: Proof of Theorem 11 (*Continued*). It is not difficult to see that the set of defaults D' is such that a set of formulas $\mathcal{E}$ is an extension of (D, W) if and only if $\mathcal{E}$ is an extension of $(D', \emptyset)$. (New defaults are introduced that add the literals from W into each extension; note that rules which add literals inconsistent with W have to be removed—such rules are not applicable in the original theory.) When the **Normal-Unary-All-Extensions** algorithm is queried with a positive literal, the algorithm directly calls the subroutine **Exists-Ext-Without-Pos-Lits**; a query with a negative literal x is converted into one for a positive literal for the new letter p. The correctness of the algorithm follows from the observation that the default rules added to D' to obtain D'' are such that all extensions of $(D', \emptyset)$ contain the negative literal x *if and only if* all extensions of $(D'', \emptyset)$ contain the positive literal p. This can be seen as follows. If $(D', \emptyset)$ has some extension containing $\sim x$, then there will be some applicable default to add $\sim x$ to the extension, and thus there is some default in the second set of defaults added to D' that can add $\neg p$ to the corresponding extension of $(D'', \emptyset)$. If $(D', \emptyset)$ has some extension that contains neither x nor $\sim x$, then none of the defaults that could add x or $\sim x$ will be applicable, and thus neither p nor $\neg p$ can be added to the corresponding extension of $(D'', \emptyset)$. So, if $(D', \emptyset)$ has an extension that does not contain x, then $(D'', \emptyset)$ has some extension that contains $\neg p$ or one that contains neither p nor $\neg p$. Finally, assume that all extensions of $(D', \emptyset)$ contain x. For each extension containing x, there will be a corresponding extension of $(D'', \emptyset)$ containing p because of the first set of defaults added to D'. Moreover, there are no other extensions of $(D'', \emptyset)$, since if $(D'', \emptyset)$ had an extension containing $\neg p$, then there would exist an extension of $(D', \emptyset)$ containing $\sim x$: contradiction; and, by a similar argument, $(D'', \emptyset)$ cannot have an extension containing neither p nor $\neg p$. Thus, all extensions of $(D', \emptyset)$ contain the negative literal x if and only if all extensions of $(D'', \emptyset)$ contain the positive literal p.

As can be seen from the algorithm, its time complexity is dominated by that of the procedure **Normal-Unary-All-Extensions-Pos**. Therefore, the time complexity of the algorithm is $O(n^2)$, where n is the length of the theory. *Q.E.D.*

Acknowledgement

We thank Hector Levesque for useful discussions and comments.

References

[1] A.V. Aho, J.E. Hopcroft and J.D. Ullman, *The Design and Analysis of Computer Algorithms* (Addison-Wesley, Reading, MA, 1976).

[2] N. Bidoit and C. Froidevaux, Minimalism subsumes default logic and circumscription in stratified logic programming, Preprint (1986).

[3] W.F. Dowling and J.H. Gallier, Linear time algorithms for testing the satisfiability of propositional Horn formula, *J. Logic Program.* **3** (1984) 267–284.

[4] D.W. Etherington and R. Reiter, On inheritance hierarchies with exceptions, in: *Proceedings AAAI-83*, Washington, DC (1983).

[5] D. Etherington, K. Forbus, M. Ginsberg, D. Israel and V. Lifschitz, Critical issues in non-monotonic reasoning, in: *Proceedings First International Conference on Principles of Knowledge Representation and Reasoning*, Toronto, Ont. (1989).

[6] D. Etherington, *Reasoning with Incomplete Information* (Morgan Kaufmann, Los Altos, CA, 1988).

[7] S.E. Fahlman, *Netl*: *A System for Representing and Using Real-World Knowledge* (MIT Press, Cambridge, MA, 1979).

[8] M.R. Garey and D.S. Johnson, *Computers and Intractability* (Freeman, New York, 1979).

[9] J.F. Horty, R. Thomason and D.S. Touretzky, A skeptical theory of inheritance in non-monotonic semantic nets, in: *Proceedings AAAI-87*, Seattle, WA (1987).

[10] H. Levesque, Making believers out of computers, *Artif. Intell.* **30** (1986) 81–108.

[11] J. McCarthy, Applications of circumscription to formalizing common-sense knowledge, *Artif. Intell.* **28** (1986) 89–116.

[12] T.C. Przymusinski, On the relationship between logic programming and non-monotonic reasoning, in: *Proceedings AAAI-88*, St. Paul, MN (1988) 444.

[13] R. Reiter, A logic for default reasoning, *Artif. Intell.* **13** (1980) 81–132.

[14] B. Selman and H. Kautz, The complexity of model-preference default theories, in: *Proceedings CSCSI-88* (1988) 102–109; also in: *Non-monotonic Reasoning*, Lecture Notes in Artificial Intelligence **346** (Springer, Berlin, 1989).

[15] B. Selman and H. Kautz, Model-preference default theories, *Artif. Intell.* **45** (1990) 287–322.

[16] L.A. Stein, Skeptical inheritance: computing the intersection of credulous extensions, in: *Proceedings IJCAI-89*, Detroit, MI (1989) 1153.

[17] J. Stillman, It's not my default: the complexity of membership problems in restricted propositional default logic, in: *Proceedings AAAI-90*, Boston, MA (1990).

[18] D.S. Touretzky, R. Thomason and J.F. Horty, A clash of intuitions: the current state of nonmonotonic multiple inheritance systems, in: *Proceedings IJCAI-87*, Milan, Italy (1987) 476–482.

[19] D.S. Touretzky, *The Mathematics of Inheritance Systems* (Morgan Kaufmann, Los Altos, CA, 1986).

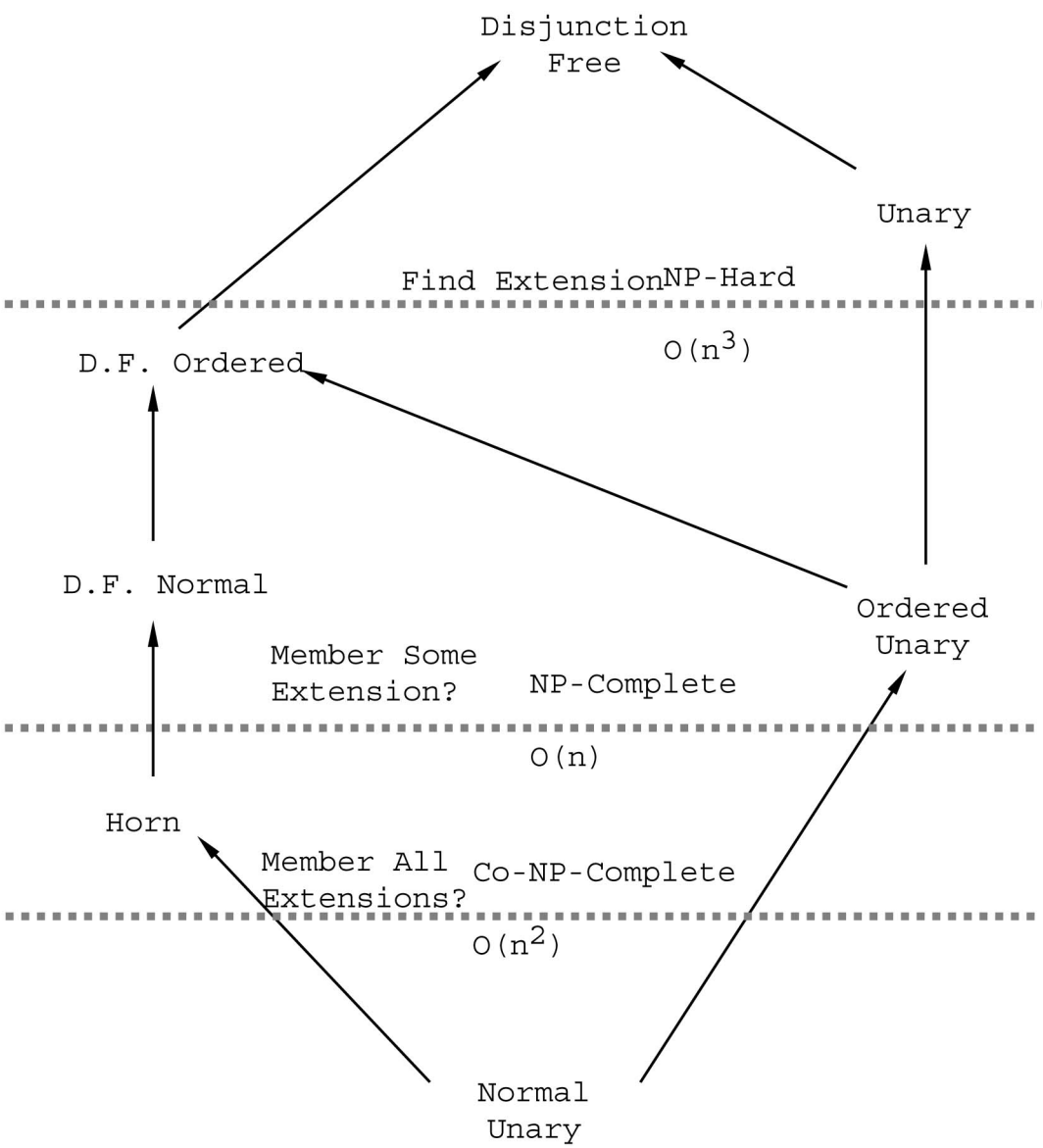

Figure 1: The hierarchy of default theories.

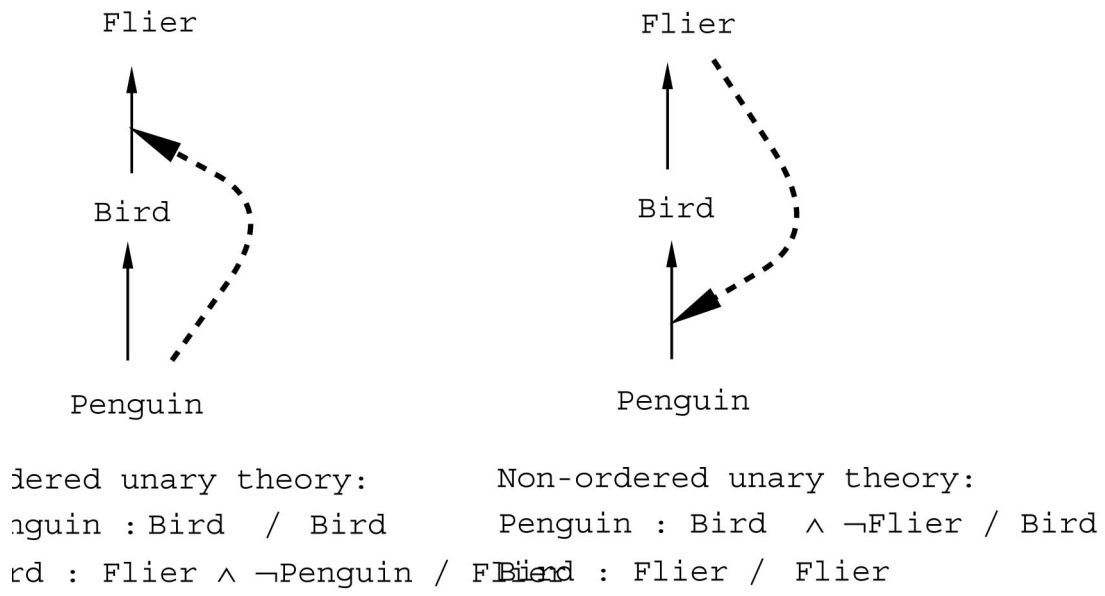

Figure 2: Ordered and unordered unary default theories.

ND-Exists-Extension-Containing(In, Out, D, W)
input: A disjunction-free theory (D,W) and sets of literals In and Out.
output: "Yes" iff there exists an extension containing all of In but none of Out.

Guess E, an arbitrary consistent superset of In disjoint from Out
$E' := W$
while [$\exists \delta \in D$. applicable(δ, E', E)] **do**
$E' := E' \cup \text{concl}(\delta)$
if [$E' \neq E$]
then "no"
else "yes"
end.

ND-Find-Extension(D, W)
input: A disjunction-free theory (D,W).
output: An extension of the theory, or "no" if there is none.

if [$\neg$ ND-Exists-Extension-Containing($\emptyset, \emptyset, D, W$)]
then return "no"
$E := \emptyset$
for $\delta \in D$ **do**
if [ND-Exists-Extension-Containing($E \cup \text{concl}(\delta), \emptyset, D, W$)]
then $E := E \cup \text{concl}(\delta)$
return E
end.

Definitions:
applicable(δ, E', E) iff
(a) $\text{pre}(\delta) \subseteq E'$,
(b) $\text{concl}(\delta) \not\subseteq E'$, and
(c) $\neg \exists p \in \text{just}(\delta). \sim p \in E$

Figure 3: Nondeterministic algorithm to find an extension of a disjunction-free theory.

Ordered-Find-Extension(D, W)
input: A disjunction-free ordered theory (D, W)
output: An extension of the theory.

Topologically sort D by $\prec$, so that $D[i]$ is the ith rule in the ordering
$E := W$
$i := 1$
while $[i \leq |D|\ ]$ **do**
if [applicable$(D[i], E, E)$]
then
begin
$E := E \cup \text{concl}(D[i])$
$i := 1$
end
else $i := i + 1$
return E
end.

Figure 4: Deterministic polynomial-time algorithm to find an extension of a disjunction-free ordered theory.

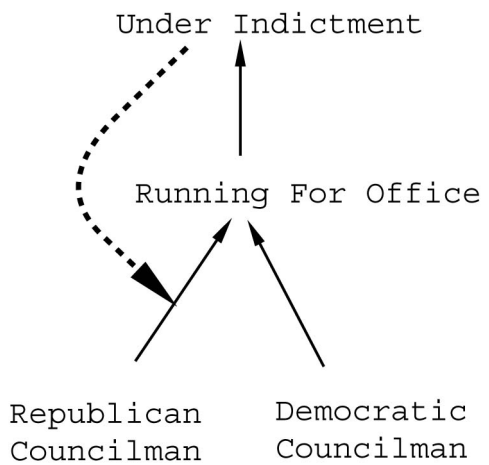

Figure 5: An unordered default theory.

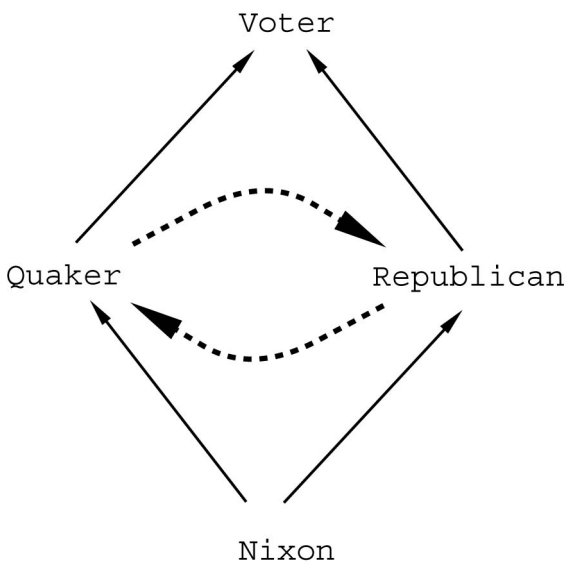

Figure 6: The extended Nixon diamond.

Normal-Unary-All-Extensions-Pos(p_k, D)

input: Positive literal p_k and a set D of normal unary defaults containing letters $p_1, \ldots, p_n$.

output: "Yes" iff every extension of $(D, \emptyset)$ contains p_k.

$L := \{p_1, p_2, \ldots, \neg p_k, \ldots, p_n\}$
while [pos-consistent(L, D)] **do**
  if [exists $p, \neg q$ in L such that
    ((NOT grounded(p, L, D))
      OR neg-inconsistent$(p, \neg q, L, D)$)]
    then $L := (L - \{p\}) \cup \{\neg p\}$
    else return "no"
return "yes"
end.

Definitions:

fixed-pos(p, L, D) iff
  (a) : $p/p \in D$,
  (b) : $\neg p/\neg p \notin D$, and
  (c) $\neg q \in L$, for each rule $q : \neg p/\neg p \in D$

pos-consistent(L, D) iff
  for all p, if fixed-pos(p, L, D), then $p \in L$

grounded(p, L, D) iff
  exists a sequence $q_0, q_1, \ldots, q_k = p$ such that
    (a) $q_j \in L$, $0 \leq j \leq k$,
    (b) : $q_0/q_0 \in D$, and
    (c) $q_{j-1} : q_j/q_j \in D$, $1 \leq j \leq k$

neg-inconsistent$(p, \neg q, L, D)$ iff
  (a) p and $\neg q \in L$
  (b) $p : q/q \in D$,
  (c) : $\neg q/\neg q \notin D$, and
  (d) $\neg r \in L$, for each rule $r : \neg q/\neg q \in D$

Figure 7: Skeptical reasoning algorithm with a positive literal and a set of normal unary defaults as input.

Normal-Unary-All-Extensions(x, D, W)
input: A literal x and a normal unary theory (D, W).
output: "Yes" iff every extension of the theory contains x.

$D' := \{\delta \in D \mid \sim \text{concl}(\delta) \notin W\} \cup \{: y/y \mid y \in W\}$
if [x is a positive literal]
then Normal-Unary-All-Extensions-Pos(x,D')
else
begin
Let p be a new letter
$D'' := D' \cup \{\text{pre}(\delta) : p/p \mid \delta \in D' \text{ and } \text{concl}(\delta) = \{x\}\}$
$\cup \{\text{pre}(\delta) : \neg p/\neg p \mid \delta \in D' \text{ and } \text{concl}(\delta) = \{\sim x\}\}$
Normal-Unary-All-Extensions-Pos(p,D'')
end
end.

Figure 8: Algorithm to determine if a literal holds in all extensions of a normal unary theory.

Reduction-Find-To-Some(D, W)
input: A disjunction-free theory (D, W).
output: An extension of the theory, or "no" if there is none.

Let p be a propositional letter not appearing in (D, W)
if [¬Some-Extension$(p, D \cup \{: p/p\}, W)$]
then return "no"
$E := \emptyset$
for each literal x **do**
if [Some-Extension$(p, D \cup \{\bigwedge E \wedge x : p/p\}, W)$]
then $E := E \cup \{x\}$
return E
end.

Reduction-Some-To-All(x, D, W)
input: A disjunction-free theory (D, W).
output: "Yes" if (D, W) has an extension containing x, otherwise "no".

if [All-Extensions$(\sim x, (D \cup \{: \neg x/\neg x\}, W)$]
then "no"
else "yes"
end.

Figure 9: Turing reductions of reasoning tasks.