ELSEVIER Artificial Intelligence 84 (1996) 113-150

Artificial Intelligence

Default reasoning using classical logic [★]

Rachel Ben-Eliyahu [a,*], Rina Dechter [b,1]

[a] *Mathematics and Computer Science Department, Ben-Gurion University of the Negev, Beer-Sheva 84105, Israel*

[b] *Information & Computer Science, University of California, Irvine, CA 92717, USA*

Received December 1993; revised May 1995

Abstract

In this paper we show how propositional default theories can be characterized by classical propositional theories: for each finite default theory, we show a classical propositional theory such that there is a one-to-one correspondence between models for the latter and extensions of the former. This means that computing extensions and answering queries about coherence, set-membership and set-entailment are reducible to propositional satisfiability. The general transformation is exponential but tractable for a subset which we call 2-DT—a superset of *network default theories* and *disjunction-free default theories*. Consequently, coherence and set-membership for the class 2-DT is NP-complete and set-entailment is co-NP-complete.

This work paves the way for the application of decades of research on efficient algorithms for the satisfiability problem to default reasoning. For example, since propositional satisfiability can be regarded as a constraint satisfaction problem (CSP), this work enables us to use CSP techniques for default reasoning. To illustrate this point we use the taxonomy of tractable CSPs to identify new tractable subsets for Reiter's default logic. Our procedures allow also for computing stable models of extended logic programs.

1. Introduction

Researchers in artificial intelligence have found Reiter's default logic [29] [2] attractive and have used it widely for declarative representations of problems in a variety of areas, including diagnostic reasoning [30], theory of speech acts [28], natural language

[★] Most of this work was done while the first author was a graduate student at the Cognitive Systems Laboratory, Computer Science Department, University of California, Los Angeles, CA, USA.

[*] Corresponding author. E-mail: rachel@cs.bgu.ac.il.

[1] E-mail: dechter@ics.uci.edu.

[2] In this paper, when we mention "default logic" we mean "Reiter's default logic".

0004-3702/96/$15.00 © 1996 Elsevier Science B.V. All rights reserved
SSDI 0004-3702(95)00095-X

processing [25], and inheritance hierarchies with exceptions [11]. Most importantly, it has been shown that logic programs with classical negation and with "negation by default" can be embedded very naturally in default logic, and thus default logic provides semantics for logic programs [4, 16].

However, while knowledge can be specified in a natural way in default logic, the concept of extension as presented by Reiter is quite tricky. Moreover, as Reiter has shown, there is no procedure that computes extensions of an arbitrary default theory. Recent research indicates that the complexity of answering basic queries on propositional default logic is very high (Σ_2^P- or Π_2^P-complete [17,34]), and that even for very simple propositional default theories, the problem is NP-hard [19,33].

In this paper we show how we can confront these difficulties by translating default theories into classical propositional theories. Our approach leads to the identification of a class of theories for which we have effective ways of computing extensions and testing set-membership and set-entailment, and to the identification of new tractable subsets for default logic.

We introduce the concept of *meta-interpretations*—truth functions that assign truth values to *clauses* rather than to logical symbols—and define when such a truth function is a model for a given default theory. Studying the properties of these models enables us to show that any finite propositional default theory can be compiled into a classical propositional theory such that there is a one-to-one correspondence between models of the classical theory and extensions of the default theory. Queries about coherence and entailment in default logic are thus reducible to queries about satisfiability in propositional logic.

The main advantage of this mapping is that it reduces computation in default logic to propositional satisfiability, a task that has already been explored extensively. Moreover, our method introduces a deterministic algorithm for computing extensions of any finite propositional default theory, while previous algorithms[3] (e.g. [11, 18, 19, 33]) produce an extension only for certain subsets of all default theories. Our translation is exponential in general. However, there is a significant sublanguage which we call *2-default theories* (2-DT), for which our translation is tractable. The class 2-DT includes the so-called *network default theories*—the default logic version of inheritance networks [11] and the class of *disjunction-free default theories*, in which formulas with disjunction are forbidden. It has been shown [16] that the class of disjunction-free default theories can embed extended logic programs; answer sets of the latter coincide with extensions of the former. Therefore, techniques developed for finding extensions for 2-DT are applicable for computing logic programs as well.

As a by-product of our translation, we learn that the coherence problem and the set-membership problem for the class 2-DT is NP-complete and that the set-entailment problem for the class 2-DT is co-NP-complete.[4] The translation also provides a general framework for identifying additional NP-complete subclasses. Note that in general these problems are Σ_2^P- or Π_2^P-hard.

[3] Of course there also exists the brute-force algorithm, according to which you check for every subset of clauses, whether or not it is an extension of the theory. Though it is clear that it is sufficient to consider a finite number of such subsets, this brute-force algorithm is extremely expensive.

[4] See Section 5.1 for details.

Once a default theory is expressed as a propositional theory, we may apply many existing heuristics and algorithms on propositional satisfiability. In particular, we show how topological considerations can be used to identify new tractable subsets, and how constraint satisfaction techniques can be effectively applied to tasks of default reasoning.

The rest of the introduction is organized as follows: in the following section we discuss the connections between default logic, logic programming, and inheritance networks, to demonstrate that the work presented here has a direct influence on computational issues in these fields as well. In Section 1.2 we will then give an introductory discussion about the basic ideas and contributions of this paper and explain its organization.

1.1. Default logic, inheritance networks, and logic programs

1.1.1. Reiter's default logic

We begin with a brief introduction to Reiter's default logic [29]. Let $\mathcal{L}$ be a first-order language over a countable alphabet. A *default theory* is a pair $\Delta = (D, W)$, where D is a set of defaults and W is a set of closed well-formed formulas (wffs) in $\mathcal{L}$. A *default* is a rule of the form

$$\frac{\alpha : \beta_1, \ldots, \beta_n}{\gamma}, \tag{1}$$

where $\alpha, \beta_1, \ldots, \beta_n$, and γ are formulas in $\mathcal{L}$.[5]

A default δ can also be written using the syntax $\alpha : \beta_1, \ldots, \beta_n / \gamma$. α is called the prerequisite (notation: $pre(\delta)$); $\beta_1, \ldots, \beta_n$ are the justifications (notation: $just(\delta)$); and γ is the conclusion (notation: $concl(\delta)$). The intuition behind a default can be stated as "if I believe α and I have no reason to believe that one of the β_i is false, then I can believe γ". A default $\alpha : \beta/\gamma$ is *normal* if $\gamma = \beta$. A default is *semi-normal* if it is in the form $\alpha : \beta \wedge \gamma/\gamma$. A default theory is *closed* if all the first-order formulas in D and W are closed.

The set of defaults D induces an *extension* on W. Intuitively, an extension is a maximal set of formulas that is deducible from W using the defaults in D. Let E^* denote the logical closure of E in $\mathcal{L}$. We use the following definition of an extension:

Definition 1.1 (*Extension* [29, Theorem 2.1]). Let $E \subseteq \mathcal{L}$ be a set of closed wffs, and let (D, W) be a closed default theory. Define

(1) $E_0 = W$, and

(2) for $i \geqslant 0$, $E_{i+1} = E_i^* \cup \{\gamma \mid \alpha : \beta_1, \ldots, \beta_n/\gamma \in D$ where $\alpha \in E_i$ and $\neg\beta_1, \ldots, \neg\beta_n \notin E\}$.

E is an extension for Δ iff for some ordering $E = \bigcup_{i=0}^{\infty} E_i$. (Note the appearance of E in the formula for E_{i+1}.)

Many tasks on a default theory Δ may be formulated using one of the following queries:

[5] Empty justifications are equivalent to the identically true proposition **true** [31].

- *Coherence*: Does Δ have an extension? If so, find one.
- *Set-membership*: Given a set of clauses T, is T contained in some extension of Δ?
- *Set-entailment*: Given a set of clauses T, is T contained in every extension of Δ?

In Section 6 we will also consider a special case of set-membership which we call *clause-membership*, where the set T is a single clause.

In this paper we focus on *propositional* default logic. It has been shown that the coherence problem is Σ_2^P-complete for this class and remains so even if restricted to semi-normal default theories [17,34]. Membership and entailment for the class of normal propositional default theories were shown to be Σ_2^P-complete and Π_2^P-complete, respectively, even if T is restricted to contain a single literal [17,34]. In this paper we will show subclasses for which these tasks are easier.[6]

It has been shown that the subclass 2-DT of all default theories is powerful enough to embed both inheritance networks and logic programs. The following two subsections elaborate on this.

1.1.2. Inheritance networks and network default theories

An inheritance network is a knowledge representation scheme in which the knowledge is organized in a taxonomic hierarchy, thus allowing representational compactness. If many individuals share a group of common properties, an abstraction of those properties is created, and all those individuals can then "inherit" from that abstraction. Inheritance from multiple classes is also allowed. For more information on this subject, see [11,37].

Etherington [11] proposed a subclass of default theories, called *network default theories*, as suitable for providing formal semantics and a notion of sound inference for inheritance networks.

Definition 1.2 (*Network default theory* [11]). A default theory Δ is a *network theory* iff it satisfies the following conditions:

(1) W contains only
  (a) literals (i.e., atomic formulas or their negations) and
  (b) disjuncts of the form $(\alpha \vee \beta)$ where α and β are literals.
(2) D contains only normal and semi-normal defaults of the form $\alpha : \beta/\beta$ or $\alpha : \beta \wedge \gamma_1 \wedge \cdots \wedge \gamma_n/\beta$ where α, β, and γ_i are literals.

Etherington suggests formalizing inheritance relations in network default theories in such a way that an extension of a network default theory would correspond to a set of coherent conclusions that one could draw from the inheritance network it represents. Thus all the queries defined above (coherence, set-membership, set-entailment) are still relevant when dealing with network default theories.

1.1.3. Default theories and logic programs

Logic programming is a paradigmatic way of representing programs and data in a declarative manner using symbolic logic. Originally, the language used by logic programs was restricted to Horn clauses. Its expressive power was greatly improved after the

[6] Assuming the polynomial hierarchy does not collapse at this level.

introduction of negation in the body of the rules. This negation was generally interpreted as "negation by default", not classical negation, resulting in a grounded predicate being considered false iff it cannot be proved from the program. For an overview of this field, see [21].

One of the most prominent semantics for logic programs is *stable model semantics* [4,14,16]. Gelfond and Lifschitz [16] have shown how stable model semantics can be naturally generalized to the class of *extended* logic programs, in which two types of negation—classical negation and negation by default—are used.

An extended logic program is a set of rules of the form

$$r_0 \longleftarrow p_1, \ldots, p_m, \text{not } q_1, \ldots, \text{not } q_n, \tag{2}$$

where each of the r, p, and q are literals, and *not* is a negation by default operator. Stable model semantics associates a set of models, or *answer sets*, with such an extended logic program.

Gelfond and Lifschitz established a one-to-one correspondence between extended logic programs and disjunction-free default theories by identifying a rule of the form (2) with the default

$$\frac{p_1 \wedge \cdots \wedge p_m : \sim q_1, \ldots, \sim q_n}{r_0},$$

where $\sim q$ is the literal opposite to q ($\sim P = \neg P$, $\sim \neg P = P$). They have shown that each extension of such a default theory corresponds to an answer set of its twin logic program. A similar idea was introduced by Bidoit and Froidevaux [4].

The above discussion suggests concluding that any algorithm that computes extensions of a default theory will also compute answer sets of logic programs under stable model semantics. Moreover, any semantics attached to a default theory provides meaning to a logic program as well.

1.2. The main contribution of this paper

The exposition in some sections of this paper involves many technical issues, so we will first familiarize the reader with the basic ideas.

In this paper we provide a way to translate *any* finite propositional default theory into a classical propositional theory so that the queries on the default theory are specifiable as queries about satisfiability or entailment in classical propositional logic. In order to give the reader a feel for this translation, we will present three default theories considered in Reiter's original paper on default logic [29], and for each theory we will provide the corresponding propositional theory. We will explain, without delving into technical details, the principle behind our mapping.

Example 1.3. Consider the following default theory [29, Example 2.3]

$$D = \left\{ \frac{:C}{\neg D}, \frac{:D}{\neg C} \right\}, \qquad W = \emptyset.$$

This theory has two extensions: $\{\neg C\}^*$ and $\{\neg D\}^*$. We will now show how this result is realized using our translation. For each literal X in $\{\neg C, \neg D\}$, let I_X be an atom with the intuitive meaning "X is in the extension". So, for example, $I_{\neg D}$ has the meaning "$\neg D$ is in the extension". Applying this vocabulary, we will set constraints on the extension of (D, W). The default rule $: C/\neg D$ imposes the constraint "if C is consistent with the extension, then $\neg D$ is in the extension", in other words: "if $\neg C$ is not in the extension, then $\neg D$ is in the extension". We can write it in propositional logic as follows:[7]

$$\neg I_{\neg C} \supset I_{\neg D}. \tag{3}$$

Accordingly, the default rule $: D/\neg C$ imposes the constraint "if $\neg D$ is not in the extension, then $\neg C$ is in the extension". We can write it in propositional logic as:

$$\neg I_{\neg D} \supset I_{\neg C}. \tag{4}$$

If $\neg D$ is in the extension, it must be the case that $\neg C$ is not in the extension, because the default $C/\neg D$ is the only rule that can be used to derive $\neg D$, and it will be activated only if C is consistent. The same applies for $\neg C$. Therefore, we add the constraints:

$$I_{\neg D} \supset \neg I_{\neg C}, \tag{5}$$

$$I_{\neg C} \supset \neg I_{\neg D}. \tag{6}$$

If we combine the formulas (3)-(6), we arrive at a theory which has two models: M_1 and M_2. In M_1, $I_{\neg C}$ is true and $I_{\neg D}$ is false. In M_2, $I_{\neg D}$ is true and $I_{\neg C}$ is false. M_1 corresponds to the extension $\{\neg C\}^*$ and M_2 corresponds to the extension $\{\neg D\}^*$.

Example 1.4. Consider the following default theory [29, Example 2.2]

$$D = \left\{ \frac{:C}{\neg D}, \frac{:D}{\neg E}, \frac{:E}{\neg F} \right\}, \qquad W = \emptyset.$$

This theory has one extension: $\{\neg D, \neg F\}^*$. We will now show how this result is realized using our translation. This time we use the vocabulary $\{I_{\neg C}, I_{\neg D}, I_{\neg E}, I_{\neg F}\}$. The default rule $: C/\neg D$ imposes the constraint

$$\neg I_{\neg C} \supset I_{\neg D}, \tag{7}$$

the default rule $: D/\neg E$ imposes the constraint

$$\neg I_{\neg D} \supset I_{\neg E}, \tag{8}$$

and the default rule $: E/\neg F$ imposes the constraint

$$\neg I_{\neg E} \supset I_{\neg F}. \tag{9}$$

Since extensions are supposed to be minimal, we assert that if $\neg D$ is in the extension, it must be the case that $\neg C$ is not in the extension, because the default $C/\neg D$ is the only rule that can be used to derive $\neg D$, and it will be activated only if C is consistent. Same for $\neg E$ and $\neg F$. Therefore, we add the constraints:

[7] $\supset$ is the usual material implication in classical logic.

$$I_{\neg D} \supset \neg I_{\neg C}, \tag{10}$$

$$I_{\neg E} \supset \neg I_{\neg D}, \tag{11}$$

$$I_{\neg F} \supset \neg I_{\neg E}. \tag{12}$$

Since there is no default which derives $\neg C$, we also add the requirement

$$\neg I_{\neg C}. \tag{13}$$

If we combine the formulas (7)-(13), we arrive at a theory which has one model, where the only true atoms are $I_{\neg D}$ and $I_{\neg F}$. This model corresponds to the extension $\{\neg D, \neg F\}^*$.

Example 1.5. Consider the following default theory [29, Example 2.6, p.91]

$$D = \left\{ \frac{:A}{\neg A} \right\}, \qquad W = \emptyset.$$

We will translate this theory as follows:

$$\neg I_{\neg A} \supset I_{\neg A},$$

$$I_{\neg A} \supset \neg I_{\neg A}.$$

The first formula constrains that the default rule should be satisfied. The second conveys the claim that since the extension is minimal, if it contains $\neg A$ it must be the case that $\neg A$ was derived using the only default in D, and therefore that $\neg A$ is not in the extension. The propositional theory above is inconsistent, and indeed the default theory we consider has no extension.

In the sequel to this section we will formally justify the translations illustrated above, present the general algorithms, and give more examples. The rest of the paper is organized as follows: After introducing some preliminary definitions in Section 2, we provide in Section 3 the concept of a model for a default theory and explain the theory behind our translation. In Sections 4 and 5 we discuss how the models presented in Section 3 can be treated as classical models of propositional logic. We present algorithms that associate for each finite default theory a classical propositional theory that characterizes its extensions. Then, in Section 6 we use constraint satisfaction techniques to show how our approach leads to the discovery of new tractable subsets for default logic. Section 7 contains concluding remarks, and missing proofs appear in Appendix A.

Before moving on, we would like to clarify a subtle but important point. Some of the decision problems we discuss here have been proven to be NP-complete or co-NP-complete for some *subsets* of all propositional default theories [19,33]. This means, almost by definition, that there actually exists a polynomial translation from these *subsets* to propositional theories such that queries on the translated default theories are answerable by solving satisfiability of the corresponding classical theories. The consequences of the work presented here goes beyond this initial observation. First, we can show a direct and simple translation: our translation does not require the encoding of Turing machines in propositional theory. In other words, even for subclasses of default logic for which the above problems were shown to be NP- or co-NP-complete, the complexity of the translation we provide is much lower than the complexity implied by these decision problems being NP- or co-NP-complete. Second, our translation is *perfect*[8]—which means that each model of the classical theory derived from the translation corresponds to an extension of the original default theory. Third, our translation applies to the class of *all* finite propositional default theories—not only to restricted subclasses—and can therefore also be used as a tool for identifying additional subclasses of default theories for which the problem of coherence, set-entailment and set-membership are in NP or in co-NP. In general the complexity of our translation is exponential, but if it is polynomial for some subclass, it means that for this subclass the problems of coherence, entailment and membership are in NP or in co-NP.

2. Definitions and preliminaries

We denote propositional symbols by uppercase letters $P, Q, R, \ldots$, propositional literals (e.g. P, $\neg P$) by lowercase letters $p, q, r, \ldots$, formulas by $\alpha, \beta, \ldots$, conjunctions of literals by $d, d_1, \ldots$, and disjunctions of literals (clauses) by $c, c_1, c_2, \ldots$. The empty clause is denoted by Λ. The set of all resolvents of two clauses c_1 and c_2 will be denoted by $res(c_1, c_2)$. The resolution closure of a set of clauses T is the set obtained by repeatedly resolving pairs of clauses of T and adding the resolvents to T until a fixed point is reached.

A formula is in a conjunctive normal form (CNF) iff it is a conjunction of clauses. A formula is in disjunctive normal form (DNF) iff it is a disjunction of conjunctions of literals. Each formula has equivalent formulas[9] in CNF and DNF. The function $CNF(\alpha)$ (respectively $DNF(\alpha)$) returns a formula in CNF (respectively DNF) that is equivalent to α. Although a formula may have several equivalent CNF or DNF formulas, we assume that the functions $CNF()$ and $DNF()$ return a unique output formula for each input formula. When convenient, we will refer to a clause as a set of literals, to a formula in CNF as a set of clauses, and to a formula in DNF as a set of sets of literals.

A propositional theory (in brief, a theory) is a set of propositional formulas. An *interpretation* for a theory T is a pair (S, f) where S is the set of atoms used in T and f is a truth assignment for the symbols in S. A *model* for T is an interpretation that satisfies all the formulas in T. $T \vdash \alpha$ means that α is propositionally provable from premises T, and $T \models \alpha$ means that T entails α, that is, every model of T is a model for α as well. In propositional logic, $T \vdash \alpha$ iff $T \models \alpha$. Hence we will use these notations interchangeably.

The relation $\leqslant$ between interpretations is defined as follows: $\theta_1 \leqslant \theta_2$ iff the set of symbols to which θ_1 assigns **true** is a subset of the set of symbols to which θ_2 assigns **true**. An interpretation θ is minimal among a set of interpretations I iff there is no $\theta' \neq \theta$ in I such that $\theta' \leqslant \theta$.

[8] We thank Mirek Truszczyński for suggesting this rather appropriate term.

[9] Two formulas α and β are equivalent iff $\alpha \models \beta$ and $\beta \models \alpha$.

The *logical closure* of a theory T, denoted T^*, is the set $\{\omega \mid T \vdash \omega\}$. How do we compute the logical closure of a theory T? Since the logical closure is an infinite set, it is obvious that we cannot compute it explicitly. However, when the theory is finite, we can compute a set that will represent the logical closure by using the notion of *prime implicates* as presented by Reiter and de Kleer [32].

Definition 2.1. A prime implicate of a set T of clauses is a clause c such that

(1) $T \models c$ and

(2) there is no proper subset c' of c such that $T \models c'$.

The prime implicates of a theory T will be denoted by $PI(T)$. As Reiter and de Kleer note, a brute-force method of computing $PI(T)$ is to repeatedly resolve pairs of clauses of T, add the resolvents to T, and delete subsumed clauses,[10] until a fixed point is reached.[11] There are some improvements to that method (see for example [26]), but it is clear that the general problem is NP-hard since it also solves satisfiability. Nevertheless, for special cases such as size-2 clauses, the prime implicates can be computed in $O(n^3)$ time.

Throughout the paper, and unless stated otherwise, we will assume without loss of generality that all formulas we use in default theories are in CNF, W is a set of clauses, the conclusion of each default is a single clause, and each formula in the justification part of a default is consistent.[12]

3. Propositional semantics for default logic

An extension is a belief set, that is, it is a set of formulas that are believed to be true. A single classical interpretation cannot capture the idea of a belief set. In other words, we cannot in general represent a belief set by a single model by identifying the set of all formulas that the model satisfies with the belief set. The reason is that a classical interpretation assigns a truth value to any formula, while it might be the case that neither a formula nor its negation belongs to the agent's set of beliefs.

We propose to use *meta-interpretations* to represent belief sets. In meta-interpretations we assign truth values to clauses rather than to propositional atoms, with the intuition that a clause is assigned the truth value **true** iff it belongs to the belief set. If both P and $\neg P$ are not in my belief set, they will both be assigned **false** by the meta-interpretation that represents my belief set. This motivates the following definition:

Definition 3.1 (*Meta-interpretation*). Let $\mathcal{L}$ be a set of propositional symbols. A *meta-interpretation* θ over $\mathcal{L}$ is a pair (S, f), where S is a set of clauses over $\mathcal{L}$ and f is a classical propositional interpretation for the set of symbols $\mathcal{L}_S = \{I_c \mid c \in S\}$.[13] That

[10] A clause c_1 *subsumes* a clause c_2 iff $c_1 \subseteq c_2$. c_2 is called a *subsumed* clause [5, Chapter 5].

[11] It is clear that this method will not generate all the tautologies, but these exceptions are easy to detect and handle. Hence, when computing prime implicates in the examples in this paper we omit tautologies.

[12] Note that if a default has an inconsistent justification we can simply ignore it.

[13] We chose this notation because intuitively, I_c = **true** means that c is In the belief set.

is, f is a function from $\mathcal{L}_S$ into $\{\textbf{true}, \textbf{false}\}$. A clause belonging to S will be called an *atomic clause.*

We are usually interested in a belief set of an agent that is capable of making classical logical inferences. Hence, in order to keep the size of the meta-interpretations as manageable as possible, we can assume that if a clause is assigned the value **true** in the meta-interpretation, then it is as if all its supersets were assigned **true**. In the same spirit, an arbitrary formula α will be considered **true** iff all the clauses in $CNF(\alpha)$ are true. These ideas are summarized in the following definition, in which we state when a meta-interpretation satisfies a formula.

Definition 3.2 (*Satisfiability*). A meta-interpretation $\theta = (S, f)$ *satisfies* a clause c ($\theta \mid\!\approx c$) iff either c is a tautology in classical propositional logic or there is an atomic clause $c' \subseteq c$ such that $f(I_{c'}) = \textbf{true}$. A meta-interpretation $\theta = (S, f)$ *satisfies* the formula $c_1 \wedge c_2 \wedge \cdots \wedge c_n$ ($\theta \mid\!\approx c_1 \wedge c_2 \wedge \cdots \wedge c_n$) iff for all $1 \leqslant i \leqslant n$, $\theta \mid\!\approx c_i$. A meta-interpretation *satisfies* a formula α in propositional logic iff it satisfies $CNF(\alpha)$.

Note that this definition of satisfiability has the desirable property that it is not the case that, for a given formula α, $\theta \mid\!\approx \alpha$ iff $\theta \not\mid\!\approx \neg\alpha$.

Example 3.3. Consider the meta-interpretation $M2$ in Table 1. $M2 \not\mid\!\approx P$, $M2 \not\mid\!\approx \neg P$.

In classical propositional logic, an interpretation for a theory is an assignment of truth values to the set of symbols that are used by the theory. In analogy to the classical case, we now define which meta-interpretation will be considered an interpretation for a default theory. Meta-interpretations assign truth values to clauses, not to atomic symbols. So the question is which set of clauses should be represented as atomic symbols in meta-interpretations of a given default theory. We suggest that it will be a set of clauses that contains all the prime implicates of every possible extension, because this way we can make sure that each clause in an extension will be representable by the meta-interpretation. Hence the following definitions:

Definition 3.4 (*Closure*). Let $\Delta = (D, W)$ be a default theory. We will say that a set of clauses S is a *closure* of Δ iff S is a superset of all prime implicates of every possible extension of Δ.

Definition 3.5 (*Interpretation*). Let Δ be a default theory. An *interpretation* for Δ is a meta-interpretation (S, f), where S is a closure of Δ.

Table 1
Three meta-interpretations

	I_A	$I_{\neg A}$	I_P	I_B	$I_{\neg P}$	$I_{\neg P \vee B}$
$M1$	T	F	T	T	F	T
$M2$	F	T	F	F	F	T
$M3$	F	T	T	T	F	T

It is easy to find a closure S of a given a default theory $\Delta = (D, W)$. For example, we can choose S to be the set of all clauses in the language of Δ, or the resolution closure of W union the set of all conclusions of defaults from D. However, in general, we would like the size of S to be small. We can show that the set *prime*(Δ), defined below, is a closure of Δ.

Definition 3.6 (*prime*(Δ)). Given a default theory $\Delta = (D, W)$, we first define the following sets:

- C_D is the set of all conclusions[14] of defaults in D, that is,

$$C_D = \left\{ c \mid \frac{\alpha : \beta_1, \ldots, \beta_n}{c} \in D \right\}.$$

- $\rho(\Delta)$ is the resolution closure of C_D and $PI(W)$.

We can now define *prime*(Δ): Let $\Delta = (D, W)$ be a default theory. The set *prime*(Δ) is the union of $\rho(\Delta) - \{\Lambda\}$ and $PI(W)$.[15]

Proposition 3.7 (*prime*(Δ) is a closure). *Let Δ be a default theory. prime(Δ) is a closure of Δ.*

Example 3.8. Consider the following default theory Δ:

$$D = \left\{ \frac{A : P}{P}, \frac{: A}{A}, \frac{: \neg A}{\neg A} \right\}, \qquad W = \{\neg P \vee B\}.$$

$PI(W) = \{\neg P \vee B\}$, $C_D = \{P, A, \neg A\}$, and $\rho(\Delta) = PI(W) \cup C_D \cup \{B, \Lambda\}$. Therefore, *prime*$(\Delta) = \{\neg P \vee B, P, A, \neg A, B\}$.

As we will see later, this theory has two extensions:

- *extension* 1 $(E1)$: $\{A, P, B\}^*$,
- *extension* 2 $(E2)$: $\{\neg A, \neg P \vee B\}^*$,

and indeed *prime*(Δ) is a superset of all prime implicates of $E1$ and $E2$.

We now want to build an interpretation (S, f) for Δ. For reasons to be explained later, we will choose S to be *prime*$(\Delta) \cup \{\neg P\}$. So we get $\mathcal{L}_S = \{I_{\neg P \vee B}, I_P, I_{\neg P}, I_A, I_{\neg A}, I_B\}$. Since $|\mathcal{L}_S| = 6$, we have 2^6 different interpretations over this fixed S. Table 1 lists three of them.

In classical propositional logic, a model for a theory is an interpretation that satisfies the theory. The set of formulas satisfied by the model is a set that is consistent with the theory, and a formula is entailed by the theory if it is true in all of its models. In the same spirit, we want to define when an interpretation for a default theory is a model. Ultimately, we want the set of all the formulas that a model for the default theory satisfies to be an extension of that default theory. If we practice skeptical reasoning, a formula will be entailed by the default theory if it belongs to all of its models.

[14] Note that we have assumed that the conclusion of each default is a single clause.

[15] Note that this definition means that Λ belongs to *prime*(Δ) iff it belongs to $PI(W)$.

Since each model is supposed to represent an extension that is a deductively closed theory, each model for a default theory is required to have the property that if a clause c follows from a set of clauses C and for each $c' \in C$, $I_{c'}$ is true, then I_c is true too. Formally,

Definition 3.9 (*Deductive closure*). A meta-interpretation $\theta = (S, f)$ is *deductively closed* iff it satisfies:

(1) For each two atomic clauses c and c' such that $c \subset c'$, if $f(I_c) = \textbf{true}$ then $f(I_{c'}) = \textbf{true}$.

(2) For each two atomic clauses c and c', if $f(I_c) = \textbf{true}$ and $f(I_{c'}) = \textbf{true}$ then $\theta \mathrel{|\!\!\approx} res(c, c')$.

A model of a default theory will also have to *satisfy* each clause from W and each default from D, in the following sense:

Definition 3.10 (*Satisfying a default theory*). A meta-interpretation θ *satisfies* a default theory Δ iff:

(1) For each $c \in W$, $\theta \mathrel{|\!\!\approx} c$.

(2) For each default from D, if θ satisfies its preconditions and does not satisfy the negation of each of its justifications, then it satisfies its conclusion.

We would also like every clause that a model for a default theory satisfies to have a "reason" to be true:

Definition 3.11 (*Being based on a default theory*). A meta-interpretation θ *is based on* a default theory Δ iff, for each atomic clause c such that $\theta \mathrel{|\!\!\approx} c$, at least one of the following conditions holds:

(1) c is a tautology.

(2) There is a clause c_1 such that $c_1 \subset c$ and $\theta \mathrel{|\!\!\approx} c_1$.

(3) There are clauses c_1, c_2 such that $\theta \mathrel{|\!\!\approx} c_1, c_2$ and $c \in res(c_1, c_2)$.

(4) $c \in W$.

(5) There is a default $\alpha : \beta_1, \ldots, \beta_n / c$ in D such that $\theta \mathrel{|\!\!\approx} \alpha$, and for each $1 \leqslant i \leqslant n$, $\theta \not\mathrel{|\!\!\approx} \neg\beta_i$.

Example 3.12. Consider the following default theory Δ:

$$W = \{\}, \qquad D = \left\{ \frac{P : R}{Q} \right\}.$$

Clearly, $\{Q\}$ is a closure of Δ, and the meta-interpretation θ that assigns **true** only to I_Q is an interpretation for Δ. Note that θ satisfies Δ but it is not based on Δ. Indeed, the set $\{Q\}^*$ is NOT an extension of Δ.

We first define when a meta-interpretation is a weak model for a default theory Δ. As we will see later, for what we call *acyclic* default theories, every weak model is a model.

Definition 3.13 (*Weak model*). Let Δ be a default theory. A *weak model* for Δ is an interpretation θ for Δ such that

(1) θ is deductively closed,
(2) θ satisfies Δ, and
(3) θ is based on Δ.

In general, however, weak models are not models of a default theory, unless each clause that they satisfy has a *proof*, where a proof is a sequence of defaults that derive the clause from W.

Definition 3.14 (*Proof*). Let $\Delta = (D, W)$ be a default theory, and let θ be an interpretation of Δ. A *proof of a clause c* with respect to θ and Δ is a sequence of defaults $\delta_1, \ldots, \delta_n$, such that the following three conditions hold:

(1) $c \in (W \cup \{concl(\delta_1), \ldots, concl(\delta_n)\})^*$.
(2) For all $1 \leqslant i \leqslant n$ and for each $\beta_j \in just(\delta_i)$, the negation of β_j is not satisfied by θ.
(3) For all $1 \leqslant i \leqslant n$, $pre(\delta_i) \subseteq (W \cup \{concl(\delta_1), \ldots, concl(\delta_{i-1})\})^*$.

Example 3.15. Consider the following default theory Δ:

$$W = \{\}, \qquad D = \left\{ \frac{P : Q}{Q}, \frac{Q : P}{P}, \frac{: \neg P}{R} \right\}.$$

Clearly, $\{P, Q, R\}$ is a closure of Δ, and the meta-interpretation θ that assigns **true** only to I_Q and I_P is an interpretation for Δ. Note that θ is a weak model for Δ, but both P and Q do not have proofs with respect to θ and Δ. Indeed, the set $\{Q, P\}^*$ is NOT an extension of Δ.

Definition 3.16 (*Model*). Let Δ be a default theory. A *model* for Δ is a weak model θ for Δ such that each atomic clause that θ satisfies has a proof with respect to θ and Δ.

Our central claim is that if a meta-interpretation is a model for a default theory Δ, then the set of all formulas that it satisfies is an extension of Δ, and vice versa. Formally,

Theorem 3.17 (Model-extension). *Let Δ be a default theory. A theory E is an extension for Δ iff there is a model θ for Δ such that $E = \{s \mid \theta \models\!\approx s\}$.*

This theorem suggests that given a default theory $\Delta = (D, W)$ we can translate queries on this theory to queries on its models as follows: Δ has an extension iff it has a model, a set T of formulas is a member in some extension iff there is a model for Δ that satisfies T, and T is included in every extension iff it is satisfied by every model for Δ.

Example 3.18. Consider again the default theory Δ from Example 3.8, where:

$$D = \left\{ \frac{A : P}{P}, \frac{: A}{A}, \frac{: \neg A}{\neg A} \right\}, \qquad W = \{\neg P \vee B\}.$$

Recall that Δ has two extensions:

- *extension* 1 ($E1$): $\{A, P, B\}^*$,
- *extension* 2 ($E2$): $\{\neg A, \neg P \vee B\}^*$.

$M1$ and $M2$ in Table 1 are models for Δ. The set of formulas that $M1$ satisfies is equal to $E1$. The set of formulas that $M2$ satisfies is equal to $E2$. $M3$ is not a model for Δ, because $M3$ is not based on Δ: $M3$ satisfies the atomic clause P but none of the conditions of Definition 3.11 are satisfied for P.

The idea behind the definition of a proof is that each clause that the model satisfies will be derivable from W using the defaults and propositional inference. An alternative way to ensure this is to assign each atomic clause an index that is a nonnegative integer and require that if this clause is satisfied by the meta-interpretation, the clauses used in its proof have a lower index. Clauses from $PI(W)$ will get index 0, and this way the well-foundedness of the positive integers will induce well-foundedness on the clauses. The following theorem conveys this idea. Elkan [10] used a similar technique in order to ensure that the justifications supporting a node in a TMS are non-circular.

Theorem 3.19 (Indexing and proofs). *A weak model $\theta = (S, f)$ for Δ is a model for Δ iff there is a function $\rho : S \longrightarrow N^+$ such that for each atomic clause c the following conditions hold:*

(1) *$c \in W$ iff $\rho(c) = 0$.*

(2) *If $c \notin W$ then at least one of the following conditions hold:*

   (a) *There is a default $\delta = \alpha : \beta_1, \ldots, \beta_n / c \in D$ such that θ satisfies α and does not satisfy any of $\neg\beta_i$ and, for all $c_1 \in CNF(\alpha)$, there is an atomic clause $c_2 \subseteq c_1$ such that $\rho(c_2) < \rho(c)$.*

   (b) *There are two atomic clauses c_1 and c_2 such that c is a resolvent of c_1 and c_2, θ satisfies c_1 and c_2, and $\rho(c_1), \rho(c_2) < \rho(c)$.*

   (c) *There is an atomic clause $c' \subset c$ such that $\theta \models\!\!\approx c'$ and $\rho(c') < \rho(c)$.*

The above theorem is very useful in proving that for what we call *acyclic* default theories every weak model is a model for Δ.

Acyclicity is defined as follows:

Definition 3.20 (*Dependency graph*). Let Δ be a default theory and S a closure of Δ. The *dependency graph* of Δ with respect to S, $G_{\Delta,S}$, is a directed graph defined as follows:

(1) For each $c \in S$ there is a node in the graph.

(2) There is an edge from node c to node c' iff $c' \notin W$ and at least one of the following conditions holds:

   (a) $c \subset c'$.

   (b) There is a clause $c'' \in S$ such that $c' \in res(c, c'')$.

   (c) There is a default $\alpha : \beta_1, \ldots, \beta_n / c'$ in D and $c \in \alpha$.

A default theory Δ is *acyclic* with respect to a closure S iff $G_{\Delta,S}$ is acyclic.

Hence, if Δ is acyclic with respect to S, the order that $G_{\Delta,S}$ induces on S satisfies the conditions of Theorem 3.19. So we can conclude the following:

$$\begin{tikzcd}
A \arrow[r] & P \arrow[r] & B & \neg P \\
\neg A & \neg P \vee B \arrow[ur] & &
\end{tikzcd}$$

Fig. 1. Dependency graph.

Theorem 3.21 (Models for acyclic theories). *If $\theta = (S, f)$ is a weak model for an acyclic default theory Δ, then θ is a model for Δ.*

Example 3.22 (*Example* 3.8 *continued*). The dependency graph of Δ is shown in Fig. 1. Δ is acyclic with respect to S.

We can also show that every model for a default theory is a *minimal* weak model. For meta-interpretations over a fixed set of atomic clauses, minimality is defined w.r.t. the following partial order: $\theta \leqslant \theta'$ iff the set of atomic clauses that θ satisfies is a subset of the set of atomic clauses that θ' satisfies. We will say that θ is minimal among a set of meta-interpretations I iff there is no $\theta' \neq \theta$ in I such that $\theta' \leqslant \theta$.

Theorem 3.23 (Minimality of models). *Every model of a default theory Δ is a minimal weak model for Δ.*

4. Expressing an acyclic default theory as a propositional theory

An interpretation (S, f) for a default theory Δ may be viewed as a classical logic interpretation over S: Treat each clause in S as a propositional symbol, and the truth value of each such "symbol" will be the value assigned by f to its corresponding clause. Our next task is to identify among those classical interpretations the ones that are *models* of Δ. We will do this by constructing a propositional theory that these models must satisfy (in the classical sense). In this section we will concentrate on acyclic default theories. Given a finite default theory Δ which is acyclic with respect to some closure of Δ, S, we will show a propositional theory $\mathcal{P}_{\Delta,S}$ that characterizes these models: If $(\mathcal{L}_S, f)$ is a classical model for that propositional theory, then (S, f) is a model for Δ; and, vice versa, if (S, f) is a model for Δ, then $(\mathcal{L}_S, f)$ is a classical model for $\mathcal{P}_{\Delta,S}$. In the next section we will generalize this approach for the class of all finite default theories.

We will first demonstrate our method with an example.

Example 4.1 (*Example* 3.8 *continued*). Consider again the default theory Δ from Example 3.8, where:

$$D = \left\{ \frac{A : P}{P}, \frac{: A}{A}, \frac{: \neg A}{\neg A} \right\}, \qquad W = \{\neg P \vee B\}.$$

Recall that Δ has two extensions:

- *extension* 1 ($E1$): $\{A, P, B\}^*$,
- *extension* 2 ($E2$): $\{\neg A, \neg P \vee B\}^*$.

Let $S = \{\neg P \vee B, P, \neg P, A, \neg A, B\}$ be a closure of Δ. Δ is acyclic with respect to S. For this theory, $\mathcal{P}_{\Delta,S}$ is the following set of formulas:

(1) $I_{\neg P} \supset I_{\neg P \vee B}$, $I_B \supset I_{\neg P \vee B}$, $I_P \wedge I_{\neg P \vee B} \supset I_B$,

(2) $I_{\neg P \vee B}$, $I_A \wedge \neg I_{\neg P} \supset I_P$, $\neg I_{\neg A} \supset I_A$, $\neg I_A \supset I_{\neg A}$,

(3) $I_A \supset \neg I_{\neg A}$, $I_{\neg A} \supset \neg I_A$, $I_P \supset I_A \wedge \neg I_{\neg P}$, $I_B \supset I_P \wedge I_{\neg P \vee B}$, $\neg I_{\neg P}$.

The classical theory $\mathcal{P}_{\Delta,S}$ expresses the requirements from a model of Δ. The first group of formulas expresses the requirement that a model for Δ must be deductively closed. It says that if one of B or $\neg P$ is true in the model then $\neg P \vee B$ should be true too, since B and $\neg P$ are subsets of $\neg P \vee B$. Similarly, since B is a resolvent of $\neg P \vee B$ and P, if both of them are true then B must be true too. Note that we do not have, for example, the formula $I_{\neg B} \wedge I_{\neg P \vee B} \supset I_{\neg P}$ since $\neg B$ does not belong to S at all.

The second group of formulas expresses the requirement that the model should satisfy Δ. For example, since $\neg P \vee B$ belongs to W, the first formula in the second group says that $\neg P \vee B$ must be true; since we have the default $A : P/P$ in Δ, we add the second formula in the group, which says that if A is true in the model and $\neg P$ is not, then P should be true in the model.

The third group of formulas says that a model for Δ should be based on Δ. For example, since the only way to add A to an extension is to use the default $: A/A$ in Δ, the first formula in this group says that if A is true in the model, then the model must not satisfy $\neg A$, otherwise the default $: A/A$ could not be activated; since no combination of formulas from W and consequences of defaults in Δ can derive $\neg P$ (except $\neg P$ itself), $\neg P$ will not be in any extension, so $\mathcal{P}_{\Delta,S}$ includes the formula $\neg I_{\neg P}$.

The reader can verify that $M1$ and $M2$ from Table 1 are the only models of $\mathcal{P}_{\Delta,S}$. If we look at $M1$ and $M2$ as meta-interpretations, we see that the set of formulas that $M1$ satisfies is equal to the extension $E1$ and the set of formulas that $M2$ satisfies is equal to the extension $E2$.

Before presenting the algorithm that translates a default theory into a classical propositional theory, some assumptions and definitions are needed. From now on we will assume that a closure S of a default theory Δ contains all the clauses that appear in Δ and all the clauses that appear in one of the CNF of the negation of each justification. We will also need the following notational shortcuts: For a given Δ over $\mathcal{L}$ and a closure of Δ, S, we will define the macros *in*( ) and *cons*( ) which translate formulas over $\mathcal{L}$ into formulas over $\mathcal{L}_S$. Intuitively, $in(\alpha)$ says that α is satisfied by the interpretation, that is, for each clause c in $CNF(\alpha)$, there is an atomic clause c' such that c' is a subset of c and $I_{c'}$ is true. $in(\alpha)$ is defined as follows:

(1) If α is a tautology, then $in(\alpha)$ = **true**.

(2) If α is an atomic clause c that is not a tautology, then $in(\alpha) = I_c$.

(3) If α is a non-atomic clause c and is not a tautology, then $in(\alpha) = \bigvee_{c' \text{ is atomic}, c' \subseteq c} I_{c'}$.

(4) If $\alpha = c_1 \wedge \cdots \wedge c_n$, then $in(\alpha) = \bigwedge_{1 \leqslant i \leqslant n} in(c_i)$.

(5) If α is not in CNF, then $in(\alpha) = in(CNF(\alpha))$.

The function $cons(\beta)$ is defined using the function *in*( ). Intuitively, $cons(\beta)$ means that the negation of β is not satisfied by the interpretation. *cons*( ) is defined as follows:

$cons(\beta) = \neg[in(\neg\beta)]$.

The algorithm shown in Fig. 2 compiles a given finite propositional default theory Δ and a closure of Δ, S, into a propositional theory, $\mathcal{P}_{\Delta,S}$, that characterizes the models of Δ. The appealing features of $\mathcal{P}_{\Delta,S}$ are summarized in the following theorems.

Theorem 4.2. *Let Δ be a finite acyclic default theory and S a closure of Δ. θ is a classical model for $\mathcal{P}_{\Delta,S}$ iff θ is a model for Δ.*

```
Algorithm TRANSLATE-1
begin:
  Step 1. $\mathcal{P}_{\Delta,S} = \emptyset$
  Step 2. $\mathcal{P}_{\Delta,S} = \mathcal{P}_{\Delta,S} + \{I_c \mid c \in W\}$
  Step 3. $\mathcal{P}_{\Delta,S} = \mathcal{P}_{\Delta,S} + \{in(\alpha) \wedge cons(\beta_1) \wedge \cdots \wedge cons(\beta_n) \supset I_c \mid \alpha : \beta_1, \ldots, \beta_n/c \in D\}$
  Step 4. $\mathcal{P}_{\Delta,S} = \mathcal{P}_{\Delta,S} + \{I_{c_1} \supset I_{c_2} \mid c_1, c_2 \in S, c_1 \subset c_2\}$
  Step 5. $\mathcal{P}_{\Delta,S} = \mathcal{P}_{\Delta,S} + \{I_{c_1} \wedge I_{c_2} \supset I_{c_3} \mid c_1, c_2, c_3 \in S$, and $c_3 \in res(c_1, c_2)\}$
  Step 6. For each atomic clause $c$, define:

          $S_c = \{c_1 \mid c_1 \in S$ and $c_1 \subset c\}$
          $R_c = \{(c_1, c_2) \mid c_1, c_2 \in S, c \in res(c_1, c_2)\}$
          $D_c = \{(\alpha, \beta_1, \ldots, \beta_n) \mid \alpha : \beta_1, \ldots, \beta_n/c \in D\}$
          SUBSET-reasons$(c) = \left[\bigvee_{c_1 \in S_c} in(c_1)\right]$
          RESOLUTION-reasons$(c) = \left[\bigvee_{(c_1,c_2) \in R_c} [in(c_1) \wedge in(c_2)]\right]$
          DEFAULT-reasons$(c)$
            $= \left[\bigvee_{(\alpha,\beta_1,\ldots,\beta_n) \in D_c} [in(\alpha) \wedge cons(\beta_1) \wedge \cdots \wedge cons(\beta_n)]\right]$

  Step 7. For each atomic clause $c \notin W$, if $S_c \cup R_c \cup D_c = \emptyset$,
          then $\mathcal{P}_{\Delta,S} = \mathcal{P}_{\Delta,S} + \{I_c \supset$ false$\}$;
          else $\mathcal{P}_{\Delta,S} = \mathcal{P}_{\Delta,S} + \{I_c \supset [$SUBSET-reasons$(c)$
                                      $\vee$ RESOLUTION-reasons$(c)$
                                      $\vee$ DEFAULT-reasons$(c)]\}$
end.
```

Fig. 2. An algorithm that translates an acyclic default theory into a propositional theory.

Proof. $\mathcal{P}_{\Delta,S}$ states the conditions of Definition 3.13 in propositional logic, and since a weak model of an acyclic default theory is a model of the default theory (Theorem 3.21), the assertion holds. □

Corollary 4.3. *Let Δ be a finite default theory which is acyclic with respect to some closure of Δ, S. Suppose $\mathcal{P}_{\Delta,S}$ is satisfiable and $\theta = (S, f)$ is a classical model for $\mathcal{P}_{\Delta,S}$, and let $E = \{c \mid c \in S,\ \theta \models I_c\}$. Then*

(1) *E^* is an extension of Δ.*

(2) *E contains all its prime implicates (that is, $PI(E) \subseteq E$).*

Proof. The first claim follows from Theorems 4.2 and 3.17. To prove the second claim, suppose c is a prime implicate of E and it is not a tautology. By definition of S, and since Δ has a consistent extension, $c \in S$. Then, by the definition of $\mathcal{P}_{\Delta,S}$ and since θ is a model for $\mathcal{P}_{\Delta,S}$, it must be the case that $\theta \models I_c$. So $c \in E$. □

5. Translating cyclic default theories

So far we have shown that for any finite *acyclic* default theory Δ and a closure of Δ, S, we can find a propositional theory, $\mathcal{P}_{\Delta,S}$, such that if $\theta = (S, f)$ is a classical model for $\mathcal{P}_{\Delta,S}$, then θ is a model for Δ. In this section we will generalize this result for default theories that might have cycles. This will imply that for any finite default theory, the questions of coherence, membership and entailment reduce to solving propositional satisfiability. We will use Theorem 3.19, which suggests the use of indices to verify that the interpretations are grounded in the default theory.

When finite default theories are under consideration, the fact that each atomic clause is assigned an index and the requirement that an index of one atomic clause will be lower than the other's can be expressed in propositional logic. Let $\#c$ stand for "the index associated with c", and let $[\#c_1 < \#c_2]$ stand for "the number associated with c_1 is less than the number associated with c_2". We use these notations as shortcuts for formulas in propositional logic that express these assertions (see Appendix B). Using these new index variables and formulas, we can express the conditions of Theorem 3.19 in propositional logic.

The size of the formulas $\#c$ and $[\#c_1 < \#c_2]$ is polynomial in the range of the indices we need. Note that we do not have to index all the clauses in S. We examine $G_{\Delta,S}$ (the dependency graph of Δ with respect to S): If a clause appearing in a prerequisite of a default is not on a cycle with the default consequent, we do not need to enforce the partial order among these two clauses. Indices are needed only for clauses that reside on cycles in the dependency graph. Furthermore, since we will never have to solve cyclicity between two clauses that do not share a cycle, the range of the index variables is bounded by the maximum number of clauses that share a common cycle. In fact, we can show that the index variable's range can be bounded further by the maximal length of an acyclic path in any *strongly connected component* in $G_{\Delta,S}$ [1].

The strongly connected components of a directed graph are a partition of its set of nodes such that for each subset C in the partition and for each $x, y \in C$, there are directed paths from x to y and from y to x in G. The strongly connected components can be identified in linear time [35]. Note that, as also implied by Theorem 3.21, if the default theory is acyclic, we do not need any indexing.

We summarize all the above discussions with an algorithm for computing $\mathcal{P}_{\Delta,S}$ for a finite default theory Δ and a closure of Δ, S. In addition to the one-place macro *in*( ), the algorithm uses a two-place macro $in(\alpha, c)$ which means "α is true independently of c", or, in other words, "α is true, and, for each clause $c' \in \alpha$, if c and c' are in the same component in the dependency graph, then the index of c' is strictly lower than the index of c".

The function $in(\alpha, c)$ is defined as follows.[16]

(1) If α is a tautology, then $in(\alpha, c) = \mathbf{true}$.

(2) If $\alpha = c'$ where c' is a clause not in the same component in the dependency graph as c, then $in(\alpha, c) = I_{c'}$.

(3) If $\alpha = c'$ where c' is a clause in the same component in the dependency graph as c, then $in(\alpha, c) = [I_{c'} \wedge [\#c' < \#c]]$.

(4) If $\alpha = c_1 \wedge \cdots \wedge c_n$, then $in(\alpha, c) = \bigwedge_{1 \leqslant i \leqslant n} in(c_i, c)$.

(5) If α is not in CNF, then $in(\alpha, c) = in(CNF(\alpha), c)$.

Except for Step 6, which is shown in Fig. 3, algorithm TRANSLATE-2 is identical to algorithm TRANSLATE-1.

The following theorems summarize the properties of our transformation. In all of these theorems, $\mathcal{P}_{\Delta,S}$ is the set of formulas resulting from translating a finite propositional theory Δ and a closure of Δ, S, using algorithm TRANSLATE-2 (see Fig. 3).

Theorem 5.1. *Let Δ be a default theory. Suppose $\mathcal{P}_{\Delta,S}$ is satisfiable and θ is a classical model for $\mathcal{P}_{\Delta,S}$, and let $E = \{c \mid c \text{ is atomic}, \theta \models I_c\}$. Then:*

(1) E^* is an extension of Δ.

(2) E contains all its prime implicates.

Proof. Part (1) follows from Theorem 3.17 and the observation that $\mathcal{P}_{\Delta,S}$ expresses the conditions of Definition 3.13 and Theorem 3.19 in propositional logic. The proof of part (2) is very similar to the proof of part (2) of Corollary 4.3. □

Theorem 5.2. *For each extension E^* for a default theory Δ, there is a model θ for $\mathcal{P}_{\Delta,S}$ such that a clause c is in E^* iff $\theta \models c$.*

Proof. Follows from Theorem 3.17 and arguments similar to those used in proving Theorem 5.1 above. □

These two theorems suggest a necessary and sufficient condition for the coherence of a finite propositional theory:

[16] Note that $in(\alpha, c)$ may be undefined when c or α contains a non-atomic clause, but that is not problematic since we will use it only when this situation does not occur.

Algorithm TRANSLATE-2, Step 6

Step 6. For each atomic clause c, define:

$$S_c = \{c_1 \mid c, c_1 \in S \text{ and } c_1 \subset c\}$$

$$R_c = \{(c_1, c_2) \mid c_1, c_2 \in S,\ c \in res(c_1, c_2)\}$$

$$D_c = \{(\alpha, \beta_1, \ldots, \beta_n) \mid \alpha : \beta_1, \ldots, \beta_n / c \in D\}$$

$$\text{SUBSET-reasons}(c) = \left[\bigvee_{c_1 \in S_c} in(c_1, c)\right]$$

$$\text{RESOLUTION-reasons}(c) = \left[\bigvee_{(c_1,c_2) \in R_c} [in(c_1, c) \wedge in(c_2, c)]\right]$$

$$\text{DEFAULT-reasons}(c) = \left[\bigvee_{(\alpha,\beta_1,\ldots,\beta_n) \in D_c} [in(\alpha, c) \wedge cons(\beta_1) \wedge \cdots \wedge cons(\beta_n)]\right]$$

Fig. 3. Step 6 of algorithm TRANSLATE-2.

Corollary 5.3. *A default theory Δ has an extension iff $\mathcal{P}_{\Delta,S}$ is satisfiable.*

Corollary 5.4. *A set of clauses T is contained in an extension of a default theory Δ iff there is a model θ for $\mathcal{P}_{\Delta,S}$ such that for each $c \in T$, $\theta \models in(c)$.*

Corollary 5.5. *A clause c is in every extension of a default theory Δ iff each model θ for $\mathcal{P}_{\Delta,S}$ satisfies $in(c)$, in other words, iff $\mathcal{P}_{\Delta,S} \models in(c)$.*

These theorems suggest that we can first translate a given finite propositional theory Δ to $\mathcal{P}_{\Delta,S}$ and then answer queries as follows: To test whether Δ has an extension, we test satisfiability of $\mathcal{P}_{\Delta,S}$; to see whether a set T of clauses is a member in some extension, we test satisfiability of $\mathcal{P}_{\Delta,S} + \{in(c) \mid c \in T\}$; and to determine whether T is included in every extension, we test whether $\mathcal{P}_{\Delta,S}$ entails the formula $[\bigwedge_{c \in T} in(c)]$.

5.1. Complexity considerations

Clearly, the transformation presented above is exponential in general. However, there are tractable subsets. For example, if the default theory is what we call a 2-default theory (2-DT), then the transformation can be done in polynomial time and the size of the propositional theory produced is polynomial in the size of the default theory. The class 2-DT is defined below. Note that this class is a superset of network default theories and normal logic programs, discussed in Sections 1.1.2 and 1.1.3.

Definition 5.6. A *2-default theory* (2-DT) is a propositional default theory Δ where all the formulas in W are in 2-CNF and, for each default $\alpha : \beta_1, \dots, \beta_n/\gamma$ in D, α is in 2-CNF, each β_i is in 2-DNF, and γ is a clause of size 2 or less.

A step-by-step analysis of the complexity of algorithm TRANSLATE-2 for a default theory $\Delta = (D, W)$ that belongs to the class 2-DT is shown below.

Let n be the number of letters in $\mathcal{L}$, the language upon which Δ is built, and let d be the maximum size of a default (the total number of characters used to write it). We assume that S, which is the closure of Δ, is the union of $prime(\Delta)$, the set of all clauses appearing in Δ, and the set of all clauses that appear in the CNF of all negations of justifications.[17] Note that S can be computed in $O(n^3 + |D|d)$ steps.[18] We denote by l the length of the longest acyclic path in any component of $G_{\Delta,S}$, by d_c the maximal number of defaults having the conclusion c, and by r the maximal number of pairs of clauses in S that yield the same clause when resolved. Note that $r \leqslant n$. Let p denote the maximum number of clauses that appear in any prerequisite and reside on the same cycle in the dependency graph (note that p is smaller than d and smaller or equal to the size of any component in the dependency graph, so $p \leqslant \min(d, n)$).

Step 2: Takes $O(n^2)$ time. Produces no more than $O(n^2)$ formulas of size 1.

Step 3: The reason we require the justification to be in 2-DNF is that we can transfer the negation of it into a 2-CNF representation in linear time. So Step 3 can be done in time $O(|D|d)$ and $|D|$ formulas of size $O(d)$ are generated.

Steps 4–5: There are at most $O(n^2)$ clauses of size $\leqslant 2$. It takes $O(n^2)$ time to find all pairs c_1, c_2 such that $c_1 \subset c_2$ (one way to do this, is to allocate an array of size $2n$ and store all clause with a common literal in the same bucket, and then produce all such pairs). Therefore, Step 4 takes $O(n^2)$ time and produces $O(n^2)$ formulas of size 2. Similarly, Step 5 takes $O(n^3)$ time (use the same array as in Step 4, but this time you have to go over two different buckets: the one for an atom, and the one for its negation), and produces $O(n^3)$ formulas of size 3.

Steps 6–7: For this step, we first have to build the dependency graph of Δ with respect to S. This takes $O(n^2 + |D|d)$ time. We assume that at the end of the graph-building phase, there is a pointer from each clause to its component and to all the defaults for which the clause is a conclusion.

For each clause c in S, the size of S_c is $\leqslant 2$, the size of R_c is $O(r)$, and the size of D_c is $O(d_c)$. For any clause c', computing $in(c', c)$ takes $O(l^2)$ time and produces a formula of size $O(l^2)$; for any prerequisite α, computing $in(\alpha, c)$ takes $O(l^2 p)$ time and produces a formula of size $O(l^2 p)$. Therefore, for each clause c, computing SUBSET-reasons takes $O(l^2)$ time and produces a formula of size $O(l^2)$. Computing DEFAULT-reasons takes $O(d_c(d + pl^2))$ time and produces a formula of this size. Computing RESOLUTION-reasons takes $O(n^2)$ time and produces a formula of size $O(r)$. Since we have $O(n^2)$ clauses, the whole step takes $O(n^2(l^2 + n^2) + |D|(d +$

[17] Note that the justifications are in 2-DNF, and hence their negation translates very easily into a 2-CNF.

[18] The reader can verify that the set of prime implicates of a set of clauses of size 2 can be computed in time $O(n^3)$ where n is the number of letters in the language.

pl^2)) time and produces $O(n^2)$ formulas of size $O(\max(d_c(d+pl^2), r))$. Note that $\max(d_c(d+pl^2), r) \leqslant d_c(d+nl^2)$.

Proposition 5.7. *For* 2-DT, *the above transformation takes* $O(n^2(l^2+n^2)+|D|(d+pl^2))$ *time and produces* $O(\max(n^3, |D|))$ *formulas of size* $O(d_c(d+nl^2))$.

The above proposition shows that there is a direct connection between the complexity of the translation and the cyclicity of the default theory translated, since for acyclic theories $p = l = 1$.

The complexity results obtained by Kautz and Selman [19] and Stillman [33] for default logic show that the satisfiability problem is polynomially reducible to deciding extension existence and membership in a subset of the class 2-DT, and that entailment in propositional logic is polynomially reducible to entailment for a subset of the class 2-DT. Their results establish the NP-hardness of the existence and membership problems and the co-NP-hardness of the entailment problem for the class 2-DT. The polynomial transformation to satisfiability that we have presented in the last section implies that existence, membership, and entailment are in NP or in co-NP for the class 2-DT. Hence we conclude the following:

Corollary 5.8. *The coherence problem (i.e. extension existence) for the class* 2-DT *is* NP-*complete.*

Corollary 5.9. *Set-membership for the class* 2-DT *is* NP-*complete.*

Proof. By Corollary 5.4, in order to check if a theory T is contained in some extension of a 2-DT Δ, we should check whether $\mathcal{P}_{\Delta,S} \cup in(T)$, where $in(T) = \{in(c) \mid c \in T\}$, is satisfiable. Since Δ is 2-DT both $\mathcal{P}_{\Delta,S}$ and $in(T)$ can be computed in time polynomial in the size of Δ and T. □

Corollary 5.10. *Set-entailment for the class* 2-DT *is co*-NP-*complete.*

Proof. Follows from Corollary 5.5 above. □

6. Tractable subsets for default logic

Once queries on a default theory are reduced to propositional satisfiability, we can use any of a number of techniques and heuristics to answer them. For instance, entailment in default logic can be solved using any complete resolution technique, since we have shown that it is reducible to entailment in propositional logic.

Our approach is useful especially for the class 2-DT, since our algorithm compiles a 2-DT in polynomial time. So if a 2-DT translates into an easy satisfiability problem, queries on the knowledge it represents can be answered efficiently. In other words, each subclass of 2-DT that translates into a tractable subclass of propositional satisfiability is a tractable subset for default logic. Consequently, we can identify easy default theories by analyzing the characteristics of 2-DT that would translate into tractable propositional theories. We will give an example of such a process by showing how some techniques developed by the *constraints-based reasoning* community can be used to identify new tractable subsets for default logic.

Constraint-based reasoning is a paradigm for formulating knowledge in terms of a set of constraints on some entities, without specifying methods for satisfying such constraints. Some techniques for testing the satisfiability of such constraints, and for finding a setting that will satisfy all the constraints specified, exploit the structure of the problem through the notion of a *constraint graph.*

The problem of the satisfiability of a propositional theory can be also formulated as a constraint satisfaction problem (CSP). For a propositional CNF theory, the constraint graph associates a node with each propositional letter and connects any two nodes whose associated letters appear in the same clause. Various parameters of the constraints graph were shown as crucially related to the complexity of solving CSP and hence to solving the satisfiability problem. These include the *induced width*, w^* (also called *tree width*), the *size of the cycle-cutset*, the *depth of a depth-first search spanning tree* of this graph, and the *size of the non-separable components* [7,8,15]. It can be shown that the worst-case complexity of deciding consistency is polynomially bounded by any one of these parameters. Since these parameters can be bounded easily by a simple processing of the graph, they can be used for bounding the complexity ahead of time. For instance, when the constraint graph is a tree, satisfiability can be answered in linear time.

In the sequel we will focus on two specific CSP techniques: *tree-clustering* [9] and *cycle-cutset decomposition* [7].

The tree-clustering scheme has a *tree-building* phase and a *query-processing* phase. The complexity of the former is exponentially dependent on the sparseness of the constraint graph, while the complexity of the latter is always linear in the size of the database generated by the tree-building preprocessing phase. Consequently, even when building the tree is computationally expensive, it may pay off when the size of the resulting tree is manageable and many queries on the same theory are expected. More details about tree-clustering and its application to reasoning in default logic can be found in Appendix C.

One of the advantages of applying tree-clustering to default reasoning is that it is possible to assess the cost of the whole process by examining the default theory prior to the translation step. We will characterize the tractability of default theories as a function of the topology of their *interaction graph.*

The interaction graph of a default theory Δ and a closure of Δ, S, is an undirected graph where each clause in S is associated with a node. Arcs are added such that for every default

$$\frac{c_1 \wedge \cdots \wedge c_n : d_{n+1}, \ldots, d_{n+m}}{c_0},$$

there are arcs connecting $c_0, c_1, \ldots, c_n$, $CNF(\neg d_{n+1}), \ldots, CNF(\neg d_{n+m})$ in a clique; every two clauses c and c' are connected iff they can be resolved, or $c \subset c'$, or there exist c'' such that $c = res(c', c'')$.

A *chord* of a cycle is an arc connecting two non-adjacent nodes in the cycle. A graph is *chordal* iff every cycle of length at least 4 has a chord. The *induced width* (w^*) of a graph G is the minimum size of a maximal clique in any chordal graph that embeds G.[19] The next theorem summarizes the complexity of our algorithm in terms of the *induced width* (w^*) of the interaction graph.

Theorem 6.1. *For a* 2-DT *whose interaction graph has an induced width* w^*, *existence, clause-membership, and set-entailment*[20] *can be decided in* $O(\alpha * 2^{w^*+1})$ *steps, where* α *is polynomial in the size of the input.*[21]

Note that w^* is always at least as large as the size of the largest default in the theory, and since there are at most $2n^2$ clauses of size $\leqslant 2$ in the language, $w^* \leqslant 2n^2$. We believe that this algorithm is especially useful for temporal reasoning in default logic, where the temporal persistence principle causes the knowledge base to have a repetitive structure, as the following example demonstrates:

Example 6.2. Suppose I leave my son at the child-care services at time t_1. If he was not picked up by my husband, between time t_2 and t_{n-1}, I expect my son to be there during any time t_i between t_2 and t_n. This can be formalized in the following default theory (D, W), where in D we have defaults of the form

$$\frac{\text{at-school}(t_i) : \text{at-school}(t_{i+1})}{\text{at-school}(t_{i+1})}$$

for $i = 1, \ldots, n-1$, and in W we have formulas of the form:

$$\text{picked-at}(t_i) \supset \neg\text{at-school}(t_{i+1})$$

for $i = 2, \ldots, n-1$.

For notational convenience, we abbreviate the above rules as follows:

$$\frac{s_i : s_{i+1}}{s_{i+1}}, \qquad p_i \supset \neg s_{i+1}.$$

The interaction graph of this theory for the closure $\{s_i, \neg s_i, \neg p_i, \neg p_i \vee \neg s_{i+1}\}$ (for the S_i, $i = 1, \ldots, n$, for the P_i, $i = 2, \ldots, n-1$) is shown in Fig. 4.

The reader can verify that $w^* \leqslant 2$ for this particular set of problems. Thus, as the number of time slots (n) grows, the time complexity for answering queries about coherence, set-membership, and set-entailment using the *tree-clustering* method grows linearly. Note that according to Selman and Kautz's classification [19], this family of theories belongs to a class for which the complexity of answering such queries is NP-hard.

[19] A graph G' embeds graph G iff $G \subseteq G'$ when we view graphs as sets of nodes and arcs.

[20] Recall the definition of these decision problems from Section 1.1.1.

[21] The input is the default theory and the set of clauses for which we test membership or entailment.

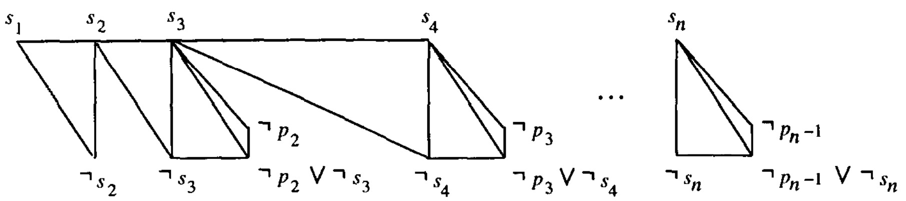

Fig. 4. Interaction graph for Example 6.2.

The *cycle-cutset* algorithm is another method that exploits the structure of the constraint graph. The cycle-cutset method is based on two facts: that tree-structured CSPs can be solved in linear time, and that variable instantiation changes the effective connectivity of the constraint graph. The basic idea is to instantiate a set of variables that constitute a cycle-cutset of the constraint graph, where a cycle-cutset is a set of nodes that, once removed, render the graph cycle-free. After the cycle-cutset is instantiated, the remaining graph is a tree, and we can apply the linear-time tree algorithm for solving the rest of the problem. If no solution is found, we have to try another instantiation of the cycle-cutset variables, and so on. Clearly, the complexity of this approach is exponentially bounded by the size of the cycle-cutset that is used. For more details on this method, see [7].

We have the following complexity bound on reasoning tasks in 2-DT:

Theorem 6.3. *For a* 2-DT *whose interaction graph has a cycle-cutset of cardinality k, existence, clause-membership, and set-entailment can be decided in* $O(\alpha * 2^k)$ *steps, where α is polynomial in the size of the input.*

7. Relation to Clark's predicate completion

In this section we discuss the relationship between the work presented here and Clark's work on program completion. Clark [6] made one of the first attempts to give meaning to logic programs with negated atoms in a rule's body ("normal programs"). He shows how each normal program Π can be associated with a first-order theory $COMP(\Pi)$, called its *completion*. His idea is that when a programmer writes a program Π, the programmer actually has in mind $COMP(\Pi)$, and thus all queries about the program should be evaluated with respect to $COMP(\Pi)$. So a formula Q is implied by the program iff $COMP(\Pi) \models Q$.

For the comparison between Clark's work and ours, we consider only normal propositional programs, that is, a set of rules of the form

$$Q \longleftarrow P_1, \ldots, P_n, \text{not}\, R_1, \ldots, \text{not}\, R_m \qquad (14)$$

where Q, $P_1, \ldots, P_n$, and $R_1, \ldots, R_m$ are atoms.

As discussed in Section 1.1.3, normal logic programs can be viewed as disjunction-free default theories by taking $W = \emptyset$ and by identifying a rule of the form (14) with the default

$$\frac{P_1 \wedge \cdots \wedge P_n : \neg R_1, \ldots, \neg R_m}{Q}.$$

Hence we can treat normal logic programs as a subclass of all default theories, and talk about extensions of normal logic programs: those are the extensions of their corresponding default theories.

Given a propositional logic program Π, $COMP(\Pi)$ is obtained in two steps:

Step 1: Replace each rule of the form (14) with the rule

$$Q \longleftarrow P_1 \wedge \cdots \wedge P_n \wedge \neg R_1 \wedge \cdots \wedge \neg R_m.$$

Step 2: For each symbol Q, let $Support(Q)$ denote the set of all clauses with Q in the head. Suppose $Support(Q)$ is the set

$$\begin{gathered} Q \longleftarrow Body_1, \\ \vdots \\ Q \longleftarrow Body_k. \end{gathered}$$

Replace it with a single sentence,

$$Q \longleftrightarrow Body_1 \vee \cdots \vee Body_k.$$

Note two special cases: If "$Q \longleftarrow$" in $Support(Q)$, simply replace $Support(Q)$ by Q. If $Support(Q)$ is empty, replace it with $\neg Q$.

Example 7.1. Consider the following program Π:

$$\begin{aligned} &P \longleftarrow Q, \neg R, \\ &P \longleftarrow V, \\ &R \longleftarrow S, \\ &V \longleftarrow . \end{aligned}$$

The completion of Π is the following propositional theory:

$$P \longleftrightarrow [Q \wedge \neg R] \vee V, \tag{15}$$

$$R \longleftrightarrow S, \tag{16}$$

$$V, \tag{17}$$

$$\neg S, \tag{18}$$

$$\neg Q. \tag{19}$$

There are interesting similarities between $COMP(\Pi)$ and the translation we provide for the same logic program. If we take the program in the previous example and translate it using algorithm TRANSLATE-1, we get that $\mathcal{P}_\Pi$ is the following theory (note that Π is acyclic according to our definitions):

$$I_V, \tag{20}$$

$$I_Q \wedge \neg I_R \supset I_P, \tag{21}$$

$$I_S \supset I_R, \tag{22}$$

$$I_V \supset I_P, \tag{23}$$

$$I_P \supset I_Q \wedge \neg I_R \vee I_V, \tag{24}$$

$$I_R \supset I_S, \tag{25}$$

$$\neg I_S, \neg I_Q, \{\neg I_{\neg L} \mid L \in \{P, Q, R, S, V\}\}, \tag{26}$$

$$\{I_L \wedge I_{\neg L} \supset \mathbf{false} \mid L \in \{P, Q, R, S, V\}\}. \tag{27}$$

Combining sentences (21), (23), and (24) and sentences (22) and (25) and replacing each symbol of the form I_L, where L is positive, with L, we get the following equivalent theory (compare to (15)-(19)):

$$P \longleftrightarrow [Q \wedge \neg R] \vee V,$$

$$R \longleftrightarrow S,$$

$$V,$$

$$\neg S,$$

$$\neg Q,$$

$$\{\neg I_{\neg L} \mid L \in \{P, Q, R, S, V\}\},$$

$$\{L \wedge I_{\neg L} \supset \mathbf{false} \mid L \in \{P, Q, R, S, V\}\}.$$

It is easy to see that each model for the above theory is a model of the completion of the program and that each model of the completion of the program can be extended to be a model for this theory. The above example can easily be generalized to a proof of the following theorem, which was proved independently by Fages [13] and in our previous work [3]:

Theorem 7.2. *Let Π be a normal acyclic propositional logic program. Then M is a model for $COMP(\Pi)$ iff $\{I_P \mid P \in M\}$ is a model for $\mathcal{P}_\Pi$.*

Proof (*Sketch*). Let Π be an acyclic normal logic program, $\mathcal{L}$ the language of Π, and $\mathcal{P}'_\Pi$ the theory obtained from $\mathcal{P}_\Pi$ by replacing each occurrence of the atom I_P, where P is an atom in $\mathcal{L}$ with the symbol P. It is easy to see that the set of models of $\mathcal{P}'_\Pi$ projected on $\mathcal{L}$ is equivalent to the set of models of $COMP(\Pi)$. □

Corollary 7.3. *Let Π be an acyclic normal propositional logic program. Π has an extension iff $COMP(\Pi)$ is consistent. Furthermore, M is a model for $COMP(\Pi)$ iff $\{P \mid M(P) = \mathbf{true}\}^*$ is an extension of Π.*

Proof. Follows from the above theorem and Theorem 3.17. $\square$

Corollary 7.4. *Let Π be an acyclic normal propositional logic program. An atom P is in the intersection of all the extensions of Π iff $COMP(\Pi) \models P$.*

Corollary 7.5. *Let Π be an acyclic normal propositional logic program. An atom P does not belong to any of the extensions of Π iff $COMP(\Pi) \models \neg P$.*

The above observations identify the class of acyclic normal propositional logic programs as a class for which default logic semantics (under "skeptical reasoning"[22]) is *equivalent* to Clark's predicate completion.

Note that if Π is a cyclic program, our translation is different from Clark's completion:

Example 7.6. Consider the following program Π_1:

$$P \longleftarrow P,$$
$$Q \longleftarrow \neg P.$$

$COMP(\Pi_1)$ is the theory $\{P \longleftrightarrow P, Q \longleftrightarrow \neg P\}$. $\mathcal{P}_{\Pi_1}$ is the theory $\{I_P \supset I_P, I_P \supset I_P \wedge [\#I_P < \#I_P], I_Q \longleftrightarrow \neg I_P\}$. Substituting I_P with P and I_Q with Q, we get that $\mathcal{P}_{\Pi_1}$ is the theory $\{P \supset P, P \supset P \wedge [\#P < \#P], Q \longleftrightarrow \neg P\}$. $COMP(\Pi_1)$ has two models, in one of them P is true and Q is false, in the other P is false and Q is true. $\mathcal{P}_{\Pi_1}$ has only one model, the model in which P is false and Q is true. Hence $\mathcal{P}_{\Pi_1}$ entails Q, while $COMP(\Pi_1)$ does not entail Q. Indeed Π_1 has one extension which is the logical closure of $\{Q\}$.

Another major difference between Clark's completion and our work is that we handle all propositional default logic and not only the subset that corresponds to normal logic programs.

8. Conclusions and related work

Reiter's default logic is a useful formalism for nonmonotonic reasoning. The applicability of default logic, however, is limited by the lack of intuitive semantics for the set of conclusions that the logic ratifies, and by the high computational complexity required for drawing such conclusions.

In this paper we have addressed some of these problems. We have shown how default theories can be characterized by theories of the already well-studied propositional logic, we have presented a procedure that computes an extension for any finite propositional default theory, and we have identified new tractable default theories.

[22] "Skeptical reasoning" means that a program entails an atom iff the atom belongs to all of the program's answer sets.

The work presented here can also be viewed as an attempt to provide default logic with semantics that are in the spirit of the semantics of Moore's auto-epistemic logic [27]. The concepts of meta-interpretation and model for a default theory are in some sense parallel to the notions of *propositional interpretation* of an auto-epistemic theory (AET) and *auto-epistemic model* of an AET [27, Section 3]. Moore defines a propositional interpretation of an AET as an assignment of truth values to the formulas in the theory provided it is consistent with the usual interpretations for classical logic (treating a formula of the form LP, where L is the "belief" operator, as a propositional symbol). Similarly, we define a meta-interpretation of a theory to be an assignment of truth values to clauses in the language of the theory. Moore defines an auto-epistemic model of an AET T as an auto-epistemic interpretation in which: (a) all the formulas of T are true and, (b) for every formula P, LP is true iff P is in T. Expansions in auto-epistemic logic correspond to extensions in default logic, and are supposed to be *stable*. Moore shows that an AET T is stable iff T contains every formula that is true in every auto-epistemic model of T. We define a model for a default theory in such a way that all the formulas satisfied by a certain model of the default theory are an extension of the theory.

Using the theory of meta-interpretations and models for propositional default theories, we presented an algorithm that compiles any finite default theory into a classical propositional theory, such that models of the last coincide with extensions of the first. This means that queries on default theories are reducible to propositional satisfiability, a problem that has been comprehensively explored. For instance, in order to compute whether a formula is in every extension of a default theory, we no longer need to compute or count all the extensions, since the problem of entailment in default logic is reduced to propositional provability.

In general, the translation algorithm is exponential, but it is polynomial for the class 2-DT, which is expressive enough to embed inheritance networks and logic programs. This leads to the observation that membership and coherence are NP-complete and entailment is co-NP-complete for the class 2-DT. Using constraint satisfaction techniques, we have identified tractable subclasses of 2-DT. We have shown how problems in temporal reasoning can be solved efficiently using the tree-clustering algorithm.

Related results for auto-epistemic logic were reported in [24], where it was shown that the question of an atom's membership in every expansion of an auto-epistemic theory is reducible to propositional provability. Also, Elkan [10] has shown that stable models of a logic program with no classical negation can be represented as models of propositional logic. Thus our work extends his results for the full power of default logic. In [3], we used a technique similar to the one presented here for computing stable models of disjunctive logic programs. We have also shown that there is an interesting relationship between the translation presented in this paper and what is called Clark's predicate completion [6]. A preliminary version of this work appears in [2].

There have been attempts in the past to relate default logic to other forms of nonmonotonic reasoning systems, such as auto-epistemic logic, circumscription, and TMS [12,18,20,23]. We believe that embedding default logic in classical logic is just as valuable since classical logic is a well-understood formalism supported by a large body of computational knowledge.

Appendix A. Proofs

A.1. Useful theorems and definitions

Definition A.1 (*see* [22]). If S is any set of clauses, then the resolution of S, denoted by $R(S)$, is the set consisting of the members of S together with all the resolvents of the pairs of members of S.

Definition A.2 (*see* [22]). If S is any set of clauses, then the nth resolution of S, denoted by $R^n(S)$, is defined for $n \geqslant 0$ as follows: $R^0 = S$, and for $n \geqslant 0$, $R^{n+1}(S) = R(R^n(S))$.

Theorem A.3 (see [22]). *Given a set S of clauses, if a clause c is a logical consequence of S which is not a tautology, then for some $n \geqslant 0$, there exists a clause $c' \in R^n(S)$, such that $c' \subseteq c$.*

Proposition A.4. *Suppose c, c_1, c_2, c_1' and c_2' are clauses, $c_1' \subseteq c_1$, $c_2' \subseteq c_2$, and $c \in res(c_1, c_2)$. Then at least one of the following conditions must hold:*
(1) $c_1' \subseteq c$.
(2) $c_2' \subseteq c$.
(3) *There is $c' \subseteq c$ such that $c' \in res(c_1', c_2')$.*

Proof. Suppose

$$c_1 = c_3 \vee P,$$
$$c_2 = c_4 \vee \neg P,$$
$$c = c_3 \vee c_4,$$

and suppose that both conditions (1) and (2) do not hold. Then it must be that

$$c_1' = c_5 \vee P, \qquad c_2' = c_6 \vee \neg P,$$

where c_5 is a subset of c_3 and c_6 is a subset of c_4. Clearly, $c_5 \vee c_6$ is both a resolvent of c_1' and c_2' and a subset of c.

Theorem A.5. (see [29, Corollary 2.2]) *A closed default theory (D, W) has an inconsistent extension iff W is inconsistent.*

A.2. Proofs of propositions and theorems

Proposition 3.7 (*prime*(Δ) is a closure)). *Let Δ be a default theory. prime(Δ) is a closure of Δ.*

Proof. Suppose E is an extension of $\Delta = (D, W)$. Since $PI(E) \subseteq E$, it is sufficient to show that for each $c \in E$ there is a clause c' in *prime*(Δ) such that $c' \subseteq c$. If E is inconsistent, then by Theorem A.5, W is inconsistent, so $\Lambda \in PI(W)$, and so $\Lambda \in$

prime(Δ). Suppose E is consistent. By Definition 1.1, for some ordering, $E = \bigcup_{i=0}^{\infty} E_i$, where E_i is as defined there. We will show that for each $c \in E$, there is a clause c' in *prime*$(\Delta) \cap E$ such that $c' \subseteq c$. The proof is by induction on $\min(c)$, where $\min(c)$ is the minimum i such that $c \in E_i$.

Case $\min(c) = 0$. In this case, it must be that $c \in W$. Our claim is true since $PI(W) \subseteq$ *prime*$(\Delta) \cap E$.

Induction step. Assume the claim is true for $\min(c) = n$, where $n \geqslant 0$, show that it is true for $n + 1$. Note that $c \neq \Lambda$, since E is consistent.

Suppose c was introduced first at E_{n+1}. So either $c \in C_D$ or $E_n \models c$. If $c \in C_D$, then clearly our assertion holds. Assume $E_n \models c$. By Theorem A.3, for some j, there is $c'' \in R^j(E_n)$ such that $c'' \subseteq c$. We will show by induction on a minimum such j that there is $c' \in$ *prime*$(\Delta) \cap E$ such that $c' \subseteq c$. For $j = 0$, this is clear due to the induction hypothesis on n. For $j > 0$, let c_1, c_2 be clauses in $R^l(E_n)$, $l < j$, such that $c'' \in res(c_1, c_2)$. By the induction hypothesis, there are c_1', c_2' in *prime*$(\Delta) \cap E$ such that $c_1' \subseteq c_1$, $c_2' \subseteq c_2$. By Proposition A.4, either $c_1' \subseteq c''$ or $c_2' \subseteq c''$ or there is c_3 in $res(c_1', c_2')$ such that $c_3 \subseteq c''$. *prime*$(\Delta) \cap E$ is closed by resolution (unless the resolvent is Λ, but $c_3 \in E$ and hence $c_3 \neq \Lambda$). So $c_3 \in$ *prime*$(\Delta) \cap E$.

Theorem 3.17 (Model-extension). *Let Δ be a default theory. A theory E is an extension for Δ iff there is a model θ for Δ such that $E = \{s \mid \theta \approx s\}$.*

Proof. Let $\Delta = (D, W)$ be a default theory and $\theta = (S, f)$ a model of Δ. Let A be the set of all clauses that θ satisfies. We will show that A is an extension[23] of Δ.

We define

(1) $E_0 = W$,

(2) for $i \geqslant 0$, $E_{i+1} = E_i^* \cup \{c \mid \alpha : \beta_1, \ldots, \beta_n / c \in D$ where $\alpha \in E_i$ and $\neg\beta_1, \ldots, \neg\beta_n \notin A$ and $c \in A\}$, and

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

It is easy to verify that $E \subseteq A$. We will show that $A \subseteq E$, and thus by Definition 1.1, A is an extension of Δ.

Let $c \in A$. By definition, c has a proof with respect to (S, f) and Δ. By induction on the number of defaults used in the shortest proof, we can easily show that $c \in E$.

To prove the other direction, suppose E is an extension of Δ. Let $S =$ *prime*(Δ). We will show that $\theta = (S, f')$ is a model of Δ, where f' is defined as

$$\text{for all } c \in S, \quad f'(c) = \textbf{true} \iff c \in E.$$

It is easy to verify that θ is deductively closed and satisfies Δ. By Definition 1.1, there are sets $E_0, E_1, \ldots$ such that

(1) $E_0 = W$,

(2) for $i \geqslant 0$, $E_{i+1} = E_i^* \cup \{c \mid \alpha : \beta_1, \ldots, \beta_n / c \in D$ where $\alpha \in E_i$ and $\neg\beta_1, \ldots, \neg\beta_n \notin E\}$, and

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

[23] Without loss of generality, we assume in this proof that an extension is a set of clauses, and all formulas in Δ are in CNF.

By induction on the minimal i such that an arbitrary clause c belongs to E_i, we can show that c has a proof with respect to θ and Δ. So every atomic clause that θ satisfies has a proof with respect to θ and Δ. It is left to show that θ is based on Δ. Let c be an atomic clause. By induction on i, the minimum number of defaults used in a proof for c, we will show that one of the conditions of Definition 3.11 holds for c.

Case $i = 0$. It must be the case that $W \models c$, and hence c is in every extension of Δ. Let E be an extension of Δ. Since S includes all prime implicates of E, and θ satisfies c, there must be a clause $c' \in S$ such that $c' \subseteq c$ and θ satisfies c'. If $c' \neq c$ we are done. Else, c is a prime implicate of E, and so there must be two clauses c_1, c_2 in E such that $c \in res(c_1, c_2)$. By definition, θ satisfies c_1 and c_2. So item (3) of Definition 3.11 holds for c.

Case $i > 0$. So either c is a consequence of some default δ and item (5) of Definition 3.11 holds for c, or c is a logical consequence of some set of clauses $C \subseteq E$, in which case one of items (1)–(3) must hold for c. □

Theorem 3.19 (Indexing and proofs). *A weak model $\theta = (S, f)$ for Δ is a model iff there is a function $\rho : S \longrightarrow N^+$ such that for each atomic clause c the following conditions hold:*

(1) *$c \in W$ iff $\rho(c) = 0$.*

(2) *If $c \notin W$ then at least one of the following conditions hold:*

  (a) *There is a default $\delta = \alpha : \beta_1, \ldots, \beta_n / c \in D$ such that θ satisfies α and does not satisfy any of $\neg\beta_i$, and for all $c_1 \in CNF(\alpha)$, there is an atomic clause $c_2 \subseteq c_1$ such that $\rho(c_2) < \rho(c)$.*

  (b) *There are two atomic clauses c_1 and c_2 such that c is a resolvent of c_1 and c_2, θ satisfies c_1 and c_2, and $\rho(c_1), \rho(c_2) < \rho(c)$.*

  (c) *There is an atomic clause $c' \subset c$ such that $\theta \models_{\approx} c'$ and $\rho(c') < \rho(c)$.*

Proof. We can show that each atomic clause has a proof with respect to θ and Δ by induction on $\rho(c)$.

Case $\rho(c) = 0$. In this case $c \in W$, so clearly c has a proof.

Case $\rho(c) > 0$. In this case c follows from other clauses using classical logic or the default rules. Those other clauses have proofs by the induction hypothesis. Hence c has a proof as well. □

Theorem 3.21 (Models for acyclic theories). *If $\theta = (S, f)$ is a weak model for an acyclic default theory Δ, then θ is a model for Δ.*

Proof. If the theory is acyclic, the dependency graph induces on S an ordering that complies with the requirements stated in Theorem 3.19. □

Theorem 3.23 (Minimality of models). *Every model of a default theory Δ is a minimal weak model for Δ.*

Proof. Suppose that $\theta = (S, f)$ is a model for Δ. Obviously, it is a weak model. We want to show it is minimal. By definition, for each atomic clause c in S there is a proof of c with respect to θ and Δ. Assume by contradiction that θ is not minimal. So there must be a weak model $\theta' = (S, f')$ such that $A^- \subset A$, where

$$A^- = \{c \mid c \text{ is atomic}, f'(c) = \textbf{true}\},$$
$$A = \{c \mid c \text{ is atomic}, f(c) = \textbf{true}\}.$$

We will show that if c has a proof with respect to Δ and θ, it must be satisfied by θ', and so $A \subseteq A^-$—a contradiction. The proof will proceed by induction on n, the number of defaults used in the proof of c. If $n = 0$, the assertion is clear since $c \in W^*$. In the event that the proof of c uses the defaults $\delta_1, \ldots, \delta_{n+1}$, we observe, using the induction hypothesis, that $(W \cup \{concl(\delta_1), \ldots, concl(\delta_n)\})^*$ is satisfied by θ'. Therefore, since θ' must satisfy Δ, it must also satisfy $concl(\delta_{n+1})$, and since it is deductively closed, it must satisfy $W \cup \{concl(\delta_1), \ldots, concl(\delta_{n+1})\}^*$, so it satisfies c. □

Theorem 6.3. *For a* 2-DT *whose interaction graph has a cycle-cutset of cardinality k, existence, clause-membership, and set-entailment can be decided in* $O(\alpha * 2^k)$ *steps, where α is polynomial in the size of the input.*

Proof. Satisfiability of a theory whose constraint graph has a cycle-cutset of cardinality k can be solved in time $O(n2^k)$, where n is the number of letters in the theory [7]. The interaction graph of a default theory Δ with a closure S is isomorphic to the constraint graph of $\mathcal{P}_{\Delta,S}$. Now, let Δ be a 2-DT with a closure S. Since Δ is a 2-DT, any clause in S is of size $\leqslant 2$, and $\mathcal{P}_{\Delta,S}$ can be computed in time polynomial in the length of Δ. Since the constraint graph of $\mathcal{P}_{\Delta,S}$ has a cycle-cutset of size k, satisfiability of $\mathcal{P}_{\Delta,S}$ can be checked in time $O(\alpha_0 2^k)$ where α_0 is polynomial in the size of $\mathcal{P}_{\Delta,S}$. By Corollary 5.3, Δ is coherent iff $\mathcal{P}_{\Delta,S}$ is satisfiable. Hence coherence of Δ can be checked in time $O(\alpha_1 2^k)$, where α_1 is polynomial in the size of the input. By Corollary 5.4, to check whether a clause c is a member of an extension of Δ, we have to check whether there is a model of $\mathcal{P}_{\Delta,S}$ which satisfies some subset of c which belongs to S. Since Δ is 2-DT, there are at most $O(|c|^2)$ clauses c' such that $c' \subseteq c$, $c' \in S$, and so clause-membership for the class 2-DT can be computed in time $O(\alpha_2 2^k)$ where α_2 is polynomial in the size of the input. By Corollary 5.5, to answer whether a set of clauses T is included in all the extensions of Δ, it is enough to check whether there is some clause c in T which some model of $\mathcal{P}_{\Delta,S}$ does not satisfy. Hence we have to check whether there is a model of $\mathcal{P}_{\Delta,S}$ that satisfies $\neg c'$ for some $c' \in S$ which is a subset of some $c \in T$. Since Δ is 2-DT, for any c in T there are at most $O(|c|^2)$ clauses c' such that $c' \subseteq c$, $c' \in S$, and so set-entailment for the class 2-DT can be computed in time $O(\alpha_3 2^k)$ where α_3 is polynomial in the size of the input. Take α to be the maximum of $\{\alpha_i \mid i = 0, 1, 2, 3\}$. □

Appendix B. Expressing indexes in propositional logic

Suppose we are given a set of symbols L to each of which we want to assign an index variable within the range 1–m.

We define a new set of symbols: $L' = \{P, P = 1, P = 2, \ldots, P = m \mid P \in L\}$, where $P = i$ for $i = 1, \ldots, m$ denote propositional letters with the intuition "P will get the number i" behind it. For each P in $\mathcal{L}'$, let $\# P$ be the following set of formulas:

$$P=1 \vee P=2 \vee \cdots \vee P=m,$$
$$P=1 \supset [\neg(P=2) \wedge \neg(P=3) \wedge \cdots \wedge \neg(P=m)],$$
$$P=2 \supset [\neg(P=3) \wedge \neg(P=4) \wedge \cdots \wedge \neg(P=m)],$$
$$\vdots$$
$$P=m-1 \supset \neg(P=m).$$

The set $\# P$ simply states that p must be assigned one and only one number.

For each P and Q in $\mathcal{L}'$, let $[\# P < \# Q]$, which intuitively means "the number of P is less than the number of Q", denote the *disjunction* of the following set of formulas:

$$P=1 \wedge Q=2,\ P=1 \wedge Q=3,\ \ldots,\ P=1 \wedge Q=m,$$
$$P=2 \wedge Q=3,\ \ldots,\ P=2 \wedge Q=m,$$
$$\vdots$$
$$P=m-1 \wedge Q=m.$$

Thus, for each symbol P to which we want to assign an index, we add $\# P$ to the theory, and then we can use the notation $[\# P < \# Q]$ to express the order between indexes.

Appendix C. Tree-clustering for default reasoning

The tree-clustering scheme [9] has a *tree-building* phase and a *query-processing* phase. The first phase of tree-clustering is restated for propositional theories in Fig. C.1. It uses the *triangulation* algorithm, which transforms any graph into a chordal[24] graph by adding edges to it [36]. The triangulation algorithm consists of two steps:

Step 1. Select an ordering for the nodes (various heuristics for good orderings are available).

Step 2. Fill in edges recursively between any two non-adjacent nodes that are connected via nodes higher up in the ordering.

Since the most costly operation within the *tree-building* algorithm is generating all the submodels of each clique (Step 4), the time and space complexity is $O(|T|n2^{|C|})$, where $|C|$ is the size of the largest clique in the chordal constraint graph generated in Step 1, $|T|$ the size of the theory and n is the number of letters used in T. It can be shown that $|C| = w^* + 1$. As a result, for classes having a bounded induced width, this method is tractable.

[24] A graph is *chordal* if every cycle of length at least 4 has a chord.

Tree-building(T, G)
Input: A propositional theory T and its constraint graph G.
Output: A tree representation of all the models of T.

Step 1. Use the *triangulation algorithm* to generate a chordal constraint graph.
Step 2. Identify all the *maximal cliques* in the graph. Let $C_1, \ldots, C_t$ be all such cliques indexed by the rank of their highest nodes.
Step 3. Connect each C_i to an ancestor C_j $(j < i)$ with whom it shares the largest set of letters. The resulting graph is called a *join tree.*
Step 4. Compute $\mathcal{M}_i$, the set of models over C_i that satisfy the set of all formulas from T composed only of letters in C_i.
Step 5. For *each* C_i and for *each* C_j adjacent to C_i in the join tree, delete from $\mathcal{M}_i$ every model M that has no model in $\mathcal{M}_j$ that agrees with it on the set of their common letters (this amounts to performing *arc consistency* on the join tree).

Fig. C.1. Propositional tree-clustering: tree-building phase.

Once the tree is built it always allows an efficient query-answering process, that is, the cost of answering many types of queries is linear in the size of the tree generated [9]. The query-processing phase is described below (m bounds the number of submodels for each clique):

Propositional tree-clustering—query processing.

(1) T is satisfiable iff none of the $\mathcal{M}_i$ is empty, a property that can be checked in $O(n)$.

(2) To see whether there is a model in which some letter P is true (respectively false), we arbitrarily select a clique containing P and test whether one of its models satisfies (respectively does not satisfy) P. This amounts to scanning a column in a table, and thus will be linear in m. To check whether a set of letters A is satisfied by some common model, we test whether all the letters belong to one cluster C_i. If so, we check whether there is a model in $\mathcal{M}_i$ that satisfies A. Otherwise, if the letters are scattered over several cliques, we temporarily eliminate from each such clique all models that disagree with A, and then re-apply arc consistency. A model satisfying A exists iff none of the resulting $\mathcal{M}_i$ becomes empty. The complexity of this step is $O(|A| nm \log m)$.

We next summarize how tree-clustering can be applied to default reasoning within the class 2-DT[25] (now n stands for the number of symbols in the default theory, m for the number of submodels in each clique; note that m is bounded by the number of the extensions that the theory has):

[25] The process described here can be applied to any default theory. The complexity analysis is the only issue appropriate only for 2-DTs.

(1) Translate the default theory into a propositional theory T (See Section 5.1).

(2) Build a default database from the propositional formulas using the *tree-building* method (takes $O(|T|n^2 * \exp(w^* + 1))$ time, where $|T|$ is the size of the theory generated at Step 1).

(3) Answer queries on the default theory using the produced tree:

  (a) To answer whether there is an extension, test whether there is an empty clique. If so, no extension exists (bounded by $O(n^2)$ steps).

  (b) To find an extension, solve the tree in a backtrack-free manner: Pick an arbitrary node C_i in the join tree, select a model M_i from $\mathcal{M}_i$, select from each of its neighbors C_j a model M_j that agrees with M_i on common letters, combine all these models, and continue to the neighbors's neighbors, and so on. The set of all models can be generated by exhausting all combinations of submodels that agree on their common letters (finding one model is bounded by $O(n^2 * m)$ steps).

  (c) To answer whether there is an extension that satisfies a clause c of size k, check whether there is a model satisfying

$$\left[\bigvee_{c' \subseteq c, I_{c'} \in T} I_{c'}\right]$$

(this takes $O(k^2 n^2 m \log m)$ steps). To answer whether c is included in all the extensions, check whether there is a solution that satisfies

$$\left[\bigwedge_{c' \subseteq c, I_{c'} \in T} \neg I_{c'}\right]$$

(bounded by $O(k^2 n^2 m \log m)$ steps).

Acknowledgements

We thank Gerhard Brewka and Kurt Konolige for very useful comments on an earlier version of this paper and Judea Pearl and Mirek Truszczyński for fruitful discussions. Michael Gelfond and Vladimir Lifschitz drew our attention to the close connection between our translation and Clark's predicate completion.

This work was supported in part by grants from the Air Force Office of Scientific Research, AFOSR 900136, by NSF grants IRI-9200918 and IRI-91573636, by Northrop MICRO grant 92-123, by Rockwell MICRO grant 92-122, by grants from Toshiba of America and Xerox Palo Alto research center, and by an IBM graduate fellowship to the first author.

References

[1] R. Ben-Eliyahu, Nonmonotonic reasoning in classical logic, Ph.D. Thesis, University of California, Los Angeles, CA (1993).

[2] R. Ben-Eliyahu and R. Dechter, Default logic, propositional logic and constraints, in: *Proceedings AAAI-91*, Anaheim, CA (1991) 379–385.
[3] R. Ben-Eliyahu and R. Dechter, Propositional semantics for disjunctive logic programs, *Ann. Math. Artif. Intell.* **12** (1994) 53–87; short version in: *Proceedings 1992 Joint International Conference and Symposium on Logic Programming* (1992).
[4] N. Bidoit and C. Froidevaux, Minimalism subsumes default logic and circumscription in stratified logic programming, in: *Proceedings IEEE Symposium on Logic in Computer Science*, Ithaca, NY (1987) 89–97.
[5] C.-L. Chang and R.C.-T. Lee, *Symbolic Logic and Mechanical Theorem Proving* (Academic Press, New York, 1987).
[6] K.L. Clark, Negation as failure, in: H. Gallaire and J. Minker, eds., *Logic and Databases* (Plenum Press, New York, 1978) 293–322.
[7] R. Dechter, Enhancement schemes for constraint processing: backjumping, learning, and cutset decomposition, *Artif. Intell.* **41** (1990) 273–312.
[8] R. Dechter and J. Pearl, Network-based heuristics for constraint satisfaction problems, *Artif. Intell.* **34** (1988) 1–38.
[9] R. Dechter and J. Pearl, Tree clustering for constraint networks, *Artif. Intell.* **38** (1989) 353–366.
[10] C. Elkan, A rational reconstruction of nonmonotonic truth maintenance systems, *Artif. Intell.* **43** (1990) 219–234.
[11] D.W. Etherington, Formalizing nonmonotonic reasoning systems, *Artif. Intell.* **31** (1987) 41–85.
[12] D.W. Etherington, Relating default logic and circumscription, in: *Proceedings IJCAI-87*, Milan, Italy (1987) 489–494.
[13] F. Fages, Consistency of Clark's completion and existence of stable models, *Meth. Logic Comput. Sci.* **2** (1992).
[14] K. Fine, The justification of negation as failure, *Logic Meth. Philos. Sci.* **8** (1989) 263–301.
[15] E.C. Freuder, A sufficient condition for backtrack-bounded search, *J. ACM* **32** (1985) 755–761.
[16] M. Gelfond and V. Lifschitz, Classical negation in logic programs and disjunctive databases, *New Generation Comput.* **9** (1991) 365–385.
[17] G. Gottlob, Complexity results for nonmonotonic logics, *J. Logic Comput.* **2** (1992) 397–425.
[18] U. Junker and K. Konolige, Computing the extensions of autoepistemic and default logics with a TMS, in: *Proceedings AAAI-90*, Boston, MA (1990) 278–283.
[19] H.A. Kautz and B. Selman, Hard problems for simple default logics, *Artif. Intell.* **49** (1991) 243–279.
[20] K. Konolige, On the relation between default and autoepistemic logic, *Artif. Intell.* **35** (1988) 343–382.
[21] R.A. Kowalski and C.J. Hogger, Logic programming, in: S.C. Shapiro, ed., *Encyclopedia of Artificial Intelligence* (Wiley, New York, 2nd ed., 1992) 873–891.
[22] C.-T. Lee, A completeness theorem and a Computer Program for finding theorems derivable from given axioms, Ph.D. Thesis, University of California, Berkeley, CA (1967).
[23] W. Marek and M. Truszczynski, Relating autoepistemic and default logic, in: R.J. Brachman, H.J. Levesque and R. Reiter, eds., *Proceedings First International Conference on Principles of Knowledge Representation and Reasoning*, Toronto, Ont. (Morgan Kaufmann, San Mateo, CA (1989) 276–288.
[24] W. Marek and M. Truszczyński, Computing intersection of autoepistemic expansions, in: *Proceedings First International Workshop on Logic Programming and Non-Monotonic Reasoning*, Washington, DC (1991) 37–50.
[25] R.E. Mercer, Using default logic to derive natural language presupposition, in: R. Goebel, ed., *Proceedings Seventh Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, Edmonton, Alta. (Morgan Kaufmann, Los Altos, CA, 1988) 14–21.
[26] E. Minicozzi and R. Reiter, A note on linear resolution strategies in consequence-finding, *Artif. Intell.* **3** (1972) 175–180.
[27] R.C. Moore, Semantical consideration on nonmonotonic logic, *Artif. Intell.* **25** (1985) 75–94.
[28] C.R. Perrault, An application of default logic to speech act theory, Tech. Rept. CSLI-87-90, SRI International, Menlo Park, CA (1987).
[29] R. Reiter, A logic for default reasoning, *Artif. Intell.* **13** (1980) 81–132.
[30] R. Reiter, A theory of diagnosis from first principles, *Artif. Intell.* **32** (1987) 57–95.
[31] R. Reiter, Personal communication (1992).

[32] R. Reiter and J. de Kleer, Foundations of assumption-based truth maintenance systems: preliminary report, in: *Proceedings AAAI-87*, Seattle, WA (1987) 183–188.

[33] J. Stillman, It's not my default: the complexity of membership problems in restricted propositional default logics, in: *Proceedings AAAI-90*, Boston, MA (1990) 571–578.

[34] J. Stillman, The complexity of propositional default logics, in: *Proceedings AAAI-92*, San Jose, CA (1992) 794–799.

[35] R. Tarjan, Depth-first search and linear graph algorithms, *SIAM J. Comput.* **1** (1972) 146–160.

[36] R.E. Tarjan and M. Yannakakis, Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs and selectively reduce acyclic hypergraphs, *SIAM J. Comput.* **13** (1984) 566–579.

[37] D.S. Touretzky, Implicit ordering of defaults in inheritance systems, in: *Proceedings AAAI-84*, Austin, TX (1984) 322–325.