

UC Irvine

ICS Technical Reports

Title

Propositional semantics for default logic

Permalink

<https://escholarship.org/uc/item/7s38b6n1>

Authors

Ben-Eliyahu, Rachel
Dechter, Rina

Publication Date

1992

Peer reviewed

Z
699
C3
no. 92-65
Rev.

Propositional Semantics for Default Logic

Rachel Ben-Eliyahu
rachel@cs.ucla.edu

Cognitive Systems Laboratory
Computer Science Department
University of California, Los Angeles, CA 90024

Rina Dechter
dechter@ics.uci.edu

Information and Computer Science
University of California, Irvine, CA 92717

Technical Report 92-65

April, 1992
revised July, 1992

Notice: This Material
may be protected
by Copyright Law
(Title 17 U.S.C.)

Appeared in the *Proceedings of the Nonmonotonic Workshop*, Vermont, 1992.

This work was supported in part by the Air Force Office of Scientific Research, AFOSR 900136, NSF grant IRI-9157636, GE Corporate R&D and by Toshiba of America.

Presented at the 4th International workshop
nonmonotonic reasoning, Vermont, May 1992

Technical Report
R-163
July 1992

Propositional Semantics for Default Logic

Rachel Ben-Eliyahu

rachel@cs.ucla.edu

Cognitive Systems Laboratory

Computer Science Department

University of California

Los Angeles, California 90024

Rina Dechter

dechter@ics.uci.edu

Information & Computer Science

University of California

Irvine, California 92717

July 27, 1992

Abstract

We present new semantics for propositional default logic based on the notion of *meta-interpretations* — truth functions that assign truth values to clauses rather than letters. This leads to a propositional characterization of default theories: for each such finite 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 an extension and answering questions 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* which is a superset of *network default theories* and *disjunction-free default theories*. This leads to the observation that coherence and membership for the class 2-DT is NP-complete and entailment is co-NP-complete.

Since propositional satisfiability can be regarded as a constraint satisfaction problem (CSP), this work also paves the way for applying CSP techniques to default reasoning. In particular, we use the taxonomy of tractable CSP 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 [Rei80] very attractive and have used it widely for declarative representations of problems in many areas, including diagnosis, inheritance networks, logic programs and natural language Processing.

However, while knowledge can be specified in a natural way in default logic, the concept of extension, as presented by Reiter, is very tricky. Moreover, as Reiter has shown, there is no procedure that computes extensions of an arbitrary default theory, and recent research has indicated that the complexity of answering basic queries on propositional default logic is very high (Σ_2^P or Π_2^P complete [Sti92, Got]) and even for very simple propositional default theories, the problem is NP-hard [KS91, Sti90].

In this paper we attempt to overcome the above difficulties by introducing a new semantics for propositional default logic which is more in the spirit of the traditional semantics for classical logic. Our approach leads to effective ways of computing extensions and testing membership and 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 logical symbols — and define when such a truth function is a model for a given default theory. By studying the properties of these models, we are able 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. Thus, queries about coherence and entailment in default logic reduce 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 been explored extensively. Moreover, our method introduces a deterministic algorithm for computing extensions of any finite propositional default theory. Previous algorithms [Eth87a] are guaranteed to produce an extension only for *ordered default theories*.

In general, our translation is exponential. However, there is an important sublanguage, which we call *2-default theories* (2-DT), that is tractable and includes the so-called *network default theories* — the default-logic version of inheritance networks [Eth87a]. Another important subclass of 2-DT

comprises the *disjunction-free default theories*, in which formulas with disjunction are forbidden. It has been shown [GL91] that this sublanguage can embed extended logic programs, in the sense that answer sets of the latter coincide with extensions of the former. Thus, 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 entailment problem for the class 2-DT is NP-complete, and the entailment problem for the class 2-DT is co-NP-complete. The translation also provides a general framework for identifying more NP-complete subclasses.

Once a default theory is expressed as a propositional theory, we can use many heuristics and algorithms that exist in the literature 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.

This paper is organized as follows: In the Section 2 we briefly review Reiter's default logic and discuss its applicability for specifying inheritance networks and logic programs. After introducing some necessary basic definitions in Section 3, we introduce in Section 4 a new semantics for default logic by defining the concept of a model for a default theory. In Sections 5 and 6 we show how these models can be treated as classical models of propositional logic, and we associate with each finite default theory a classical propositional theory that characterizes its models. Then, in Section 7, we use constraint satisfaction techniques to show how our approach leads to the discovery of new tractable subsets for default logic. Section 8 provides concluding remarks.

1.1 Default logic, inheritance networks, and logic programs

1.1.1 Reiter's default logic

The following is a brief introduction to Reiter's default logic [Rei80]. 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 (wff) in \mathcal{L} . A *default* is a rule of the form

$$\frac{\alpha : \beta_1, \dots, \beta_n}{\gamma}, \quad (1)$$

where $\alpha, \beta_1, \dots, \beta_n$, and γ are formulas in \mathcal{L} .¹

A default δ can also be written using the syntax $\alpha : \beta_1, \dots, \beta_n / \gamma$. α is called the prerequisite (notation: $pre(\delta)$); β_1, \dots, β_n the justifications (notation: $just(\delta)$); and γ the conclusion (notation: $concl(\delta)$). The intuition behind a default can be stated as "If I believe in 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 can be deduced 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) ([Rei80], 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 $i \geq 0$ $E_{i+1} = E_i^* \cup \{ \gamma \mid \alpha : \beta_1, \dots, \beta_n / \gamma \in D \text{ where } \alpha \in E_i \text{ and } \neg \beta_1, \dots, \neg \beta_n \notin E_i \}$.

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} .) \square

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

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 ?²

Set-Entailment: Given a set of clauses T , is T contained in every extension of Δ ?³

It has been shown that default logic is a formalism powerful enough to embed both inheritance networks and logic programs. The following two subsections elaborate on this.

¹Empty justifications are equivalent to the identically true proposition **true**.

²if $|T| = 1$ this problem will be called *Membership*.

³if $|T| = 1$ this problem will be called *Entailment*.

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 “inherit” from that abstraction. Inheritance from multiple classes is also allowed. For more information on this subject, see [Eth87a] or [Tou84].

Etherington ([Eth87a]) 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) [Eth87a] *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 seminormal defaults of the form: $\alpha : \beta / \beta$ or $\alpha : \beta \wedge \gamma_1 \wedge \dots \wedge \gamma_n / \beta$ where α , β , and γ_i are literals. \square*

Etherington suggests formalizing inheritance relations in network default theories. His translation is as follows:

Strict IS-A: “A’s are always B’s”. Etherington suggests translating this to the first-order formula $\forall x.A(x) \rightarrow B(x)$. Since we restrict our treatment to propositional theories, we will translate this link to the propositional rule schema $A(x) \rightarrow B(x)$.

Membership: “The individual a belongs to the class A ”. This is represented by the fact $A(a)$ (which denotes here a propositional literal).

Strict ISN’T-A: “A’s are never B’s”. Etherington translates this to the first-order formula $\forall x.A(x) \rightarrow \neg B(x)$. We will translate this link to the propositional rule schema $A(x) \rightarrow \neg B(x)$.

Nonmembership: “The individual a does not belong to the class A ”. This is represented by the fact $\neg A(a)$.

Default IS-A: “A’s are normally B’s, but exceptions are allowed”. This can be represented by the default rule schema $A(x) : B(x)/B(x)$.

Default ISN’T-A: “Normally A’s are not B’s, but exceptions are allowed”. This can be represented by the default rule schema $A(x) : \neg B(x)/\neg B(x)$.

Exception: “Normally A’s are (not) B’s, unless they have at least one of the properties C_1, \dots, C_n ”. This translates to the default rule schema $A(x) : B(x) \wedge \neg C_1(x) \wedge \dots \wedge \neg C_n(x)/B(x)$,
 $(A(x) : \neg B(x) \wedge \neg C_1(x) \wedge \dots \wedge \neg C_n(x)/\neg B(x))$.

An extension of a network default theory then corresponds to a set of coherent conclusions one could draw from the inheritance network it represents. Thus all the queries defined above (coherence, set-membership, set-entailment) are still very relevant when dealing with network default theories.

1.2.1 Default theories and logic programs

Logic programming is a paradigm for representing programs and data in a declarative way using symbolic logic. Originally, the language used by logic programs was restricted to Horn clauses, but later its expressive power was greatly improved when the use of negation in the body of the rules was introduced. This negation was usually interpreted as “negation by default” rather than classical negation, so that a grounded predicate is considered false iff it can not be proved from the program. For an overview of this field, see [KH92].

The idea behind the logic programming paradigm is that programs will be written in a declarative way using logic and will be independent of any specific mechanism for processing them. Thus, programmers will be concerned only with the meaning of their programs rather than with implementation issues. Therefore, the meaning of the program— in other words, its *semantics*— should be clear and well understood by the user.

The search for an appropriate semantics for logic programs with negation has occupied the logic programming community for years. The main approaches are well summarized in [PP90].

One of the most prominent semantics for logic programs, *stable model semantics*, was proposed by Gelfond and Lifschitz [GL91]. They have also shown how stable model semantics can be generalized naturally 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

$$r_0 \leftarrow p_1, \dots, p_m, \text{not } q_1, \dots, \text{not } q_n, \quad (2)$$

where each of r, p , and q is a literal 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 have 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

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

where $\sim q$ stands for 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 [BF87], who showed that logic programs can be specified within the formalism of default logic.

The above discussion leads to the conclusion that any algorithm that computes extensions of a default theory also computes answer sets of logic programs under stable model semantics. Furthermore, any semantics attached to a default theory gives meaning to a logic program as well.

2 Definitions and Preliminaries

We denote propositional symbols by uppercase letters P, Q, R, \dots , propositional literals (e.g. $P, \neg P$) by lowercase letters p, q, r, \dots , sentences by α, β, \dots , conjunctions of literals by d, d_1, \dots , and disjunctions of literals (clauses) by c, c_1, c_2, \dots . The empty clause is denoted by Λ . The set of all resolvents of two clauses c_1, c_2 will be denoted by $\text{res}(c_1, c_2)$. 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 sentence is in a conjunctive normal form (CNF) iff it is a conjunction of clauses. A sentence is in disjunctive normal form (DNF) iff it is a disjunction of conjunctions of literals. Each sentence has equivalent sentences⁴ in CNF and DNF. The function $CNF(\alpha)$ ($DNF(\alpha)$) returns a sentence in CNF (DNF) that is equivalent to α . When convenient, we will refer to a clause, a sentence in CNF, and a sentence in DNF as a set of literals, a set of clauses, and a set of conjunctions of literals, respectively.

A propositional theory (in brief, a theory) is a set of propositional sentences. 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 sentences 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 \leq between interpretations is defined as follows: $\theta_1 \leq \theta_2$ 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 $\theta' \leq \theta$.

The *logical closure* of a theory T , denoted T^* , is the set $\{\omega | T \vdash \omega\}$. How do we compute the logical closure of a theory T ? Since the 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 closure by using the notion of *prime implicants* as presented by Reiter and de Kleer [RdK87].

Definition 2.1 A *prime implicant* 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 $T \models c'$. \square

The prime implicants 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, until a fixed point is reached⁵. There are some improvements to that

⁴Two sentences α, β are equivalent iff $\alpha \models \beta$ and $\beta \models \alpha$.

⁵It is clear that this method will not generate all the tautologies, but these exceptions are easy to detect and handle.

method (see for example [MR72]), 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 implicants can be computed in $O(n^3)$ time.

Throughout the paper, and unless stated otherwise, we will assume w.l.g. 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 sentence in the justification part of a default is consistent⁶.

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 an agent might not be able to decide whether a piece of information is true or false, namely, whether a sentence or 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 $\mathcal{L}_S = \{I_c | c \in S\}$ ⁷. That is, f is a function from \mathcal{L}_S into $\{\mathbf{true}, \mathbf{false}\}$. A clause belonging to S will be called an atomic clause. \square*

We are usually interested in a belief set of a rational agent. If such an agent has the clause c in her belief set, she will automatically have all supersets of this clause in her belief set, since these are trivial consequences of

⁶Note that if a default has an inconsistent justification we can simply ignore it.

⁷We chose this notation because intuitively, $I_c = \mathbf{true}$ means that c is in the belief set.

c. 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 will be considered **true** iff all the clauses in one of its conjunctive normal forms 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 ($\theta \models c$) iff either c is a tautology in classical propositional logic or there is an atomic clause $c' \subseteq c$ such that $f(I_{c'}) = \mathbf{true}$. A meta-interpretation $\theta = (S, f)$ satisfies the sentence $c_1 \wedge c_2 \wedge \dots \wedge c_n$ ($\theta \models c_1 \wedge c_2 \wedge \dots \wedge c_n$) iff for all $1 \leq i \leq n$ $\theta \models c_i$. A meta-interpretation satisfies a sentence s in propositional logic iff it satisfies $CNF(s)$. \square*

Note that this definition of satisfiability has the desirable property that it is not the case that, for a given sentence s , $\theta \models s$ iff $\theta \not\models \neg s$.

Example 3.3 *Consider the meta-interpretation $M2$ in Table 1.*

$M2 \models P, M2 \not\models \neg P.$ \square

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 analog 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 our meta-interpretation. We suggest that it will be a set of clauses that contains all the prime implicants of any possible extension, because this way we can make sure that each clause in the 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 implicants of any possible extension of Δ . \square*

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 . \square

It is easy to find a closure S of a given a default theory $\Delta = (D, W)$. We can choose S to be the set of all clauses in the language of Δ , for example, 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 $\text{prime}(\Delta)$, 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 conclusions⁸ of defaults in D , that is,

$$C_D = \{c | \alpha : \beta_1, \dots, \beta_n / c \in D\}$$

$\rho(\Delta)$ is the resolution closure of C_D and $PI(W)$ with the restriction that no two resolvents are from $PI(W)$.

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

Proposition 3.7 (prime(Δ) is a closure) Let Δ be a default theory. $\text{prime}(\Delta)$ is a closure of Δ . \square

Example 3.8

Consider the following default theory:

$$\begin{aligned} D &= \{A : P/P, : A/A, : \neg A/\neg A\}, \\ W &= \{\neg P \vee B\}. \end{aligned}$$

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

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

$$\begin{aligned} \text{Extension 1 (E1): } & \{A, P, B\}^* \\ \text{Extension 2 (E2): } & \{\neg A, \neg P \vee B\}^* \end{aligned}$$

and indeed $\text{prime}(\Delta)$ is a superset of all prime implicants 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 $\text{prime}(\Delta) \cup \{\neg P\}$. So we get

	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

Table 1: Three meta-interpretations

$\mathcal{L}_S = \{I_{\neg P \vee B}, I_P, I_{\neg P}, I_A, I_{\neg A}, I_B\}$. Since $|\mathcal{L}_S| = 6$, we have 2^6 different interpretations over this fixed S . Table 1 lists three of them. \square

In classical propositional logic, a model for a theory is an interpretation that satisfies the theory. The set of sentences satisfied by the model is a set that is consistent with the theory, and a sentence is entailed by the theory if it belongs to 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 sentences that a model for the default theory satisfies will be an extension of that default theory. If we practice skeptical reasoning, a sentence will be entailed by the default theory if it belongs to all of its models.

Since each 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 clause c' and $I_{c'}$ is true, then I_c will be 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, c' such that $c \subset c'$, if $f(I_c) = \mathbf{true}$ then $f(I_{c'}) = \mathbf{true}$.*
2. *For each two atomic clauses c, c' , if $f(I_c) = \mathbf{true}$ and $f(I_{c'}) = \mathbf{true}$ then then $\theta \models \text{res}(c, c')$. \square*

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

⁸Note that we have assumed that the conclusion of each default is a single clause.

Definition 3.10 (satisfying a default theory) A meta-interpretation θ satisfies a given default theory Δ iff

1. For each $c \in W$, $\theta \models c$.
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. \square

Finally, we would also like every clause that the model 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 $\theta \models c$, at least one of the following conditions holds:

1. c is a tautology.
2. There is an atomic clause c_1 such that $c_1 \subset c$ and $\theta \models c_1$.
3. There are atomic clauses c_1, c_2 such that $\theta \models c_1, c_2$ and $c \in \text{res}(c_1, c_2)$.
4. $c \in W$.
5. There is a default $\alpha : \beta_1, \dots, \beta_n / c$ in D such that $\theta \models \alpha$, and for each $1 \leq i \leq n$ $\theta \not\models \neg \beta_i$. \square

Example 3.12 Consider the following default theory Δ :

$$W = \{ \}$$

$$D = \frac{P : R}{Q}.$$

Clearly, $\{Q\}$ is a closure of Δ , and the meta-interpretation θ that assigns true to the clause 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 . \square

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 w.r.t. θ and Δ is a sequence of defaults $\delta_1, \dots, \delta_n$, such that the following three conditions hold:

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

□

Definition 3.15 (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 w.r.t. 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 sentences that it satisfies is an extension of Δ , and vice versa. Formally,

Theorem 3.16 (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 $E = \{s \mid \theta \models s\}$. □

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 sentences 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 3.17 (*Example 3.8 continued*)

$M1$ and $M2$ are models for Δ . The set of sentences that $M1$ satisfies is equal to $E1$. The set of sentences that $M2$ satisfies is equal to $E2$. $M3$ is not a model for Δ , because $M2$ is a model and $M2 \leq M3$. \square

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 insure this is to assign each atomic clause an index that is a non-negative 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 [Elk90] used the same technique to insure that the justifications supporting a node in a TMS are noncircular.

Theorem 3.18 (indexing and proofs) *A weak model $\theta = (S, f)$ for Δ is a model for Δ iff there is a function $\rho : S \rightarrow N^+$ 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 $\delta = \alpha : \beta_1, \dots, \beta_n / c \in D$ 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 $c' \subset c$ such that $\theta \models c'$ and $\rho(c') < \rho(c)$.

\square

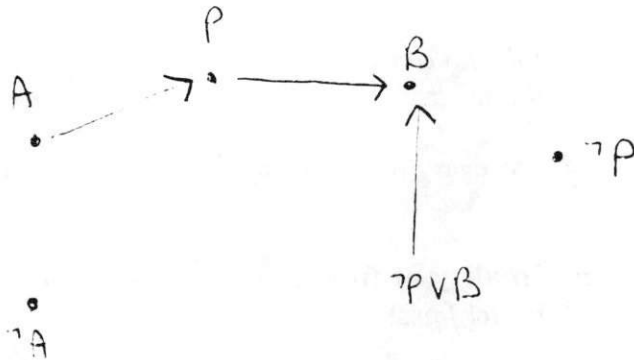


Figure 1: Dependency graph

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 3.19 (dependency graph) Let Δ be a default theory and S a closure of Δ . The dependency graph of Δ w.r.t. S , $G_{\Delta,S}$, is 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 hold:
 - (a) $c \subset c'$
 - (b) There is a clause $c'' \in S$ such that $c' \in \text{res}(c, c'')$.
 - (c) There is a default $\alpha : \beta_1, \dots, \beta_n / c'$ in D and $c \in \alpha$.

A default theory Δ is acyclic w.r.t. a closure S iff $G_{\Delta,S}$ is acyclic. \square

Hence, if Δ is acyclic w.r.t. S , the order that $G_{\Delta,S}$ induces on S satisfies the conditions of Theorem 3.18. So we can conclude the following:

Theorem 3.20 (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 . \square

Example 3.21 (example 3.8 continued) *The dependency graph of Δ is shown in Figure 1. Δ is acyclic w.r.t. S . \square*

We can also show that every model for a default theory is a *minimal weak model*.

Theorem 3.22 (minimality of models) *Every model of a default theory Δ is a minimal weak model for Δ .*

For meta-interpretations over a fixed set of atomic clauses, minimality is defined w.r.t. the following partial order: $\theta \leq \theta'$ 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 $\theta' \leq \theta$. Given a default theory Δ and a closure of Δ , S , an interpretation (S, θ) is minimal iff θ is minimal among the set of all meta-interpretations over S .

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 interpretation over S , where each clause in S is treated as a propositional symbol. Our next task is to identify those classical interpretations that are models of Δ by constructing a propositional theory that they satisfy (in the classical sense). In this section we will concentrate on acyclic default theories. Given a finite acyclic default theory Δ and a 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 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 the default logic Δ in example 3.8. Δ is acyclic. For this theory, $\mathcal{P}_{\Delta, S}$ is the following set of sentences:

- $$\begin{array}{lll}
(1) & I_{\neg P} \longrightarrow I_{\neg P \vee B}, & I_B \longrightarrow I_{\neg P \vee B}, & I_P \wedge I_{\neg P \vee B} \longrightarrow I_B \\
(2) & I_{\neg P \vee B}, & I_A \wedge \neg I_{\neg P} \longrightarrow I_P, & \neg I_{\neg A} \longrightarrow I_A, & \neg I_A \longrightarrow I_{\neg A} \\
(3) & I_A \longrightarrow \neg I_{\neg A}, & I_{\neg A} \longrightarrow \neg I_A, & I_P \longrightarrow I_A \wedge \neg I_{\neg P}, \\
& I_B \longrightarrow I_P \wedge I_{\neg P \vee B}, & \neg I_{\neg P}
\end{array}$$

The first group of sentences 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 sentence $I_{\neg B} \wedge I_{\neg P \vee B} \longrightarrow I_{\neg P}$ since $\neg B$ does not belong to S at all.

The second group of sentences expresses the requirement that the model should satisfy Δ . For example, since $\neg P \vee B$ belongs to W , the first sentence 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 sentence 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 sentences 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 sentence 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 sentence $\neg I_{\neg P}$.

The reader can verify that $M1$ and $M2$ 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 sentences that $M1$ satisfies is equal to the extension $E1$ and the set of sentences that $M2$ satisfies is equal to the extension $E2$. \square

Before presenting the translation algorithm, some assumptions and definitions are needed. From now on we will assume that S contains all the clauses that appear in Δ , including clauses that appear in the CNF of the negation of each justification. We will also need the following notational shortcuts: For a given Δ and a closure of Δ , S , we will define the macros $in()$ and $cons()$ which translate sentences over \mathcal{L} into sentences 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 $in(\alpha) = \mathbf{true}$.

2. If α is an atomic clause c that is not a tautology, then $in(\alpha) = I_c$.
3. If α is not an atomic clause c and is not a tautology, then $in(\alpha) = \bigvee_{c' \text{ is atomic, } c' \subseteq_c I} c'$.
4. If $\alpha = c_1 \wedge \dots \wedge c_n$, then $in(\alpha) = \bigwedge_{1 \leq i \leq n} in(c_i)$
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 figure 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. θ is a classical model for $\mathcal{P}_{\Delta, S}$ iff (θ, f) is a model for Δ . \square*

Theorem 4.3 *Let Δ be a finite acyclic default theory. Suppose $\mathcal{P}_{\Delta, S}$ is satisfiable and $\theta = (S, f)$ is a classical model for $\mathcal{P}_{\Delta, S}$, and let $E = \{c | c \in S, \theta \not\models c\}$. Then*

1. E contains all its prime implicants.
2. E^* is an extension of Δ . \square

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

Algorithm TRANSLATE-1

begin:

1. $\mathcal{P}_{\Delta,S} = \emptyset$
2. $\mathcal{P}_{\Delta,S} = \mathcal{P}_{\Delta,S} + \{I_c | c \in W\}$
3. $\mathcal{P}_{\Delta,S} = \mathcal{P}_{\Delta,S} + \{in(\alpha) \wedge cons(\beta_1) \wedge \dots \wedge cons(\beta_n) \longrightarrow I_c \mid \alpha : \beta_1, \dots, \beta_n / c \in D\}$
4. $\mathcal{P}_{\Delta,S} = \mathcal{P}_{\Delta,S} + \{I_{c_1} \longrightarrow I_{c_2} \mid c_1, c_2 \in S, c_1 \subset c_2\}$
5. $\mathcal{P}_{\Delta,S} = \mathcal{P}_{\Delta,S} + \{I_{c_1} \wedge I_{c_2} \longrightarrow I_{c_3} \mid c_1, c_2, c_3 \in S, \text{ and } c_3 \in res(c_1, c_2)\}$
6. For each atomic clause c , define:

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

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

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

$$\text{SUBSET-reasons}(c) = [\vee_{c_1 \in S_c} in(c_1)]$$

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

$$\text{DEFAULT-reasons}(c) = [\vee_{(\alpha, \beta_1, \dots, \beta_n) \in D_c} [in(\alpha) \wedge cons(\beta_1) \wedge \dots \wedge cons(\beta_n)]]$$

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

end.

Figure 2: An algorithm that translates an acyclic default theory into a propositional theory

will use Theorem 3.18, 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 gets 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). Using these new index variables and formulas, we can now express the conditions of theorem 3.18 in propositional logic over the language $\mathcal{L}'_S = \mathcal{L}_S \cup \{\#I_c | I_c \in \mathcal{L}_S\}$ in a way similar to the translation in algorithm TRANSLATE-1.

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 $G_{\Delta,S}$: 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 $G_{\Delta,S}$.

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 [Tar72]. Note that if the default theory is acyclic, we do not need any indexing. This was also implied by Theorem 3.20.

We summarize all the above discussions by giving an algorithm for computing $\mathcal{P}_{\Delta,S}$ for a given finite default theory Δ and a given closure of Δ , S . In addition to the one-place macro $\text{in}()$, the algorithm uses a two-place macro $\text{in}(\alpha, c)$ 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 ".

Algorithm TRANSLATE-2, step 6

6. For each atomic clause c , define:

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

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

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

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

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

$$\text{DEFAULT-reasons}(c) =$$

$$[\bigvee_{(\alpha, \beta_1, \dots, \beta_n) \in D_c} [\text{in}(\alpha, c) \wedge \text{cons}(\beta_1) \wedge \dots \wedge \text{cons}(\beta_n)]]$$

Figure 3: Step 6 of algorithm TRANSLATE-2

The function $\text{in}(\alpha, c)$ is defined as follows⁹.

1. If α is a tautology, then $\text{in}(\alpha, c) = \mathbf{true}$.
2. If $\alpha = c'$ where c' is a clause not in the same component in the dependency graph as c , then $\text{in}(\alpha, c) = I_{c'}$.
3. If $\alpha = c'$ where c' is a clause in the same component in the dependency graph as c , then $\text{in}(\alpha, c) = [I_{c'} \wedge [\#c' < \#c]]$.
4. If $\alpha = c_1 \wedge \dots \wedge c_n$, then $\text{in}(\alpha, c) = \bigwedge_{1 \leq i \leq n} \text{in}(c_i, c)$
5. If α is not in CNF, then $\text{in}(\alpha, c) = \text{in}(\text{CNF}(\alpha), c)$.

Except for Step 6, which is shown in Figure 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 sentences resulting from translating a given finite propositional theory Δ and a closure of Δ , S , using algorithm TRANSLATE-2.

⁹Note that $\text{in}(\alpha, c)$ is 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.

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

1. E contains all its prime implicants.
2. E^* is an extension of Δ . \square

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 $\theta \models c$. \square*

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

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

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 $\mathcal{P}_{\Delta,S} \models in(c)$. \square*

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 logic programs ("disjunction-free default theories").

Definition 5.6 A 2-default theory (2-DT) is a propositional default theory Δ where all the sentences 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. \square

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 (in number of conjuncts or disjuncts in the formulas appear in it). We assume that S , which is the closure of Δ , is the union of $\text{prime}(\Delta)$, the set of all clauses appearing in Δ , and the set of all clauses that appear in the CNF of all negations of justifications¹⁰. Note that S can be computed in $O(n^3 + |D| * d)$ steps (see algorithm in appendix). We denote by l the length of the longest acyclic path in any component of $G_{\Delta, S}$, by d_c the maximal size of D_c , and by r , the maximal number of pairs of clauses in S that yield the same clause when resolved. Note that $r \leq n$. p denotes the maximum number of clauses that appear in any prerequisite and reside on a cycle on 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 $p \leq \min(d, n)$).

step 2 Takes $O(n^2)$ time. Produces no more than $O(n^2)$ sentences 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. We get that this step can be done in time $O(|D| * d)$ time and produces $|D|$ sentences of size $O(d)$.

step 4 Takes $O(n^2)$ time. Produces $O(n^2)$ sentences of size 2.

step 5 Takes $O(n^3)$ time. Produces $O(n^3)$ sentences 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

¹⁰Note that the justifications are in 2-DNF, and hence their negation translates very easily into a 2-CNF.

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 ≤ 2 , the size of R_c is $O(r)$, and the size of D_c is $O(d_c)$. Computing $in(c', c)$ takes $O(l^2)$ time and produces a sentence of size $O(l^2)$; computing $in(\alpha, c)$ takes $O(l^2 * p)$ time and produces a sentence of size $O(l^2 * p)$. Therefore, for each clause c , computing SUBSET-reasons takes $O(l^2)$ time and produces a sentence of size $O(l^2)$. Computing DEFAULT-reasons takes $O(d_c * (d + pl^2))$ time and produces a sentence of this size. Computing RESOLUTION-reasons takes $O(n)$ time and produces a sentence of size $O(r)$. Since we have $O(n^2)$ clauses, the whole step takes $O(n^2(l^2 + n) + |D| * (d + pl^2))$ time and produces $O(n^2)$ sentences of size $O(\max(d_c * (d + pl^2), r))$. Note that $\max(d_c * (d + pl^2), r) \leq d_c * (d + nl^2)$.

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

The above theorem 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 $l = 1$.

For disjunction-free default theories we have a lower upper bound:

Proposition 5.8 *For disjunction-free default theories, the above transformation takes $O(|W| + |D| * (d + pl^2))$ time and produces $O(|W| + |D|)$ sentences of size $O(|D| * (d + pl^2))$. \square*

Note that if the disjunction-free default theory is acyclic, then it can be compiled into a propositional theory in linear time.

Combining Proposition 5.7 and Corollaries 5.3-5.5 we get the following results on the complexity of the class 2-DT:

Corollary 5.9 *The coherence problem (i.e. extension existence) for the class 2-DT is NP-complete.*

Corollary 5.10 *The membership problem for the class 2-DT is NP-complete.*

Corollary 5.11 *The entailment problem for the class 2-DT is co-NP-complete.*

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. So 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 theory, the constraint graph associates a node with each propositional letter and connects any two nodes whose associated letters appear in the same propositional sentence.

Various parameters of 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^* , 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* ([Fre85],[DP88], [Dec90]). 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 assessing 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* [DP89], which we will describe in detail, and *cycle-cutset decomposition* [Dec90], which we will discuss briefly.

6.1 Tree-clustering for default theories

We will show how tree-clustering can be used in default reasoning in two steps: we will show first how this technique can be used to solve satisfiability and then how it can be adopted for answering queries about a default theory.

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 be justified when the size of the resulting tree is manageable and many queries on the same theory are expected. The algorithm is summarized in Figure 4. It uses the *triangulation* algorithm, which transforms any graph into a chordal¹¹ graph by adding edges to it [TY84]. The triangulation algorithm consists of two steps:

1. Select an ordering for the nodes (various heuristics for good orderings are available).
2. Fill in edges recursively between any two nonadjacent 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 5), the time and space complexity of this preliminary phase is $O(|T| * n * 2^{|C|})$, where $|C|$ is the size of the largest clique, $|T|$ the size of the theory and n is the number of letters used in T . It can be shown that $|C| = w^* + 1$, where w^* is the *width*¹² of the ordered chordal graph (also called *induced width*). As a result, for classes having a *bounded induced width*, this method is tractable.

¹¹A graph is *chordal* if every cycle of length at least four has a chord.

¹²The *width* of a node in an ordered graph is the number of edges connecting it to nodes lower in the ordering. The width of an ordering is the maximum width of nodes in that ordering, and the width of a graph is the minimal width of all its orderings

Tree building(T, G)

input: A propositional theory T and its constraint graph G .

output: A tree representation of all the models of T .

1. Use the *triangulation algorithm* to generate a chordal constraint graph.
2. Identify all the *maximal cliques* in the graph. Let C_1, \dots, C_t be all such cliques indexed by the rank of their highest nodes.
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**.
4. Compute \mathcal{M}_i , the set of models over C_i that satisfy the set of all sentences from T composed only of letters in C_i .
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). \square

Figure 4: 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. 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 if none of its \mathcal{M}_i 's 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 (false), we arbitrarily select a clique containing P and test whether one of its models satisfies (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 reapply arc consistency. A model satisfying A exists iff none of the resulting \mathcal{M}_i 's becomes empty. The complexity of this step is $O(n * m * \log m)$. \square

We next summarize how tree-clustering can be applied to default reasoning (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):

1. Translate the 2-DT to propositional logic (generates $O(n^2)$ sentences of size $O(d_c * (d + nl^2))$).
2. Build a default database from the propositional sentences using the *tree-building* method (takes $O(|T|n^2 * \exp(w^* + 1))$, 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:

In order to find a satisfying model we 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, unite 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 $[\bigvee_{c' \subseteq c, c' \in T} I_{c'}]$ (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 $[\bigwedge_{c' \subseteq c, c' \in T} \neg I_{c'}]$ (bounded by $O(k^2 n^2 m)$ steps).

As was stated before, given a default theory Δ and a closure of Δ , S , the complexity of the algorithm presented above depends on the topology of the constraint graph of the propositional theory $\mathcal{P}_{\Delta, S}$. One of the advantages of the algorithm we suggest is that it is possible to assess the cost of the whole process by examining the default theory prior to the translation step. We are able to show a characterization of the tractability of default theories as a function of the topology of their *interaction graph*.

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

$$\frac{c_1, \dots, c_n : d_{n+1}, \dots, d_{n+m}}{c_0},$$

there are arcs connecting $c_0, c_1, \dots, c_n, CNF(\neg d_{n+1}), \dots, CNF(\neg d_{n+m})$ in a clique; every two clauses c, c' are connected iff $c \subseteq c'$ or if there exist c'' such that $c = res(c', c'')$.

The next theorem characterizes the complexity of our algorithm in terms of the *induced width* of the interaction graph.

Theorem 6.1 *For a 2-DT Δ whose interaction graph has an induced width w^* , existence, membership, and entailment can be decided in $O(n^2 * (n + 2^{w^*+1}) + |D| * d)$ steps when the theory is acyclic and $O(n^2(n^{w^*+1} + n + l^2) + |D| * (d + pl^2))$ steps when the theory is cyclic. \square*

Note that an upper bound to w^* is $2n^2$ and that the upper bound is always at least as large as the size of the largest default in the theory. We believe,

however, that this algorithm is especially useful for temporal reasoning in default logic, where the temporal persistence principal causes the knowledge base to have a repetitive structure. as the following example demonstrates:

Example 6.2

Suppose I park my car in the parking lot at time t_1 . If it was not removed from the lot by being stolen or towed between time t_1 and t_n , I expect my car to be there during any time t_i between t_1 and t_n . This can be formalized in the following default theory, where we have n defaults of the form

$$\frac{\text{parked}(t_i) : \text{parked}(t_{i+1})}{\text{parked}(t_{i+1})},$$

and in W we have n sentences for each of the forms:

$$\begin{aligned} \text{stolen}(t_i) &\longrightarrow \text{moved}(t_i) \\ \text{towed-away}(t_i) &\longrightarrow \text{moved}(t_i) \\ \text{moved}(t_i) &\longrightarrow \neg \text{parked}(t_{i+1}) \end{aligned}$$

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

$$\begin{aligned} &\frac{pt_i : pt_{i+1}}{pt_{i+1}} \\ &st_i \longrightarrow mt_i \\ &tt_i \longrightarrow mt_i \\ &mt_i \longrightarrow \neg pt_{i+1} \end{aligned}$$

The interaction graph of this theory for the closure $\{ pt_i, \neg pt_i, \neg mt_i, \neg st_i, \neg tt_i, \neg mt_i \vee \neg pt_i, \neg st_i \vee mt_i, \neg tt_i \vee mt_i \}$ is shown in Figure 5. If we use the ordering shown in the figure, we find that $w^* \leq 2$ 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 [KS91], this family of theories belongs to a class for which the complexity of answering such queries is NP-hard.

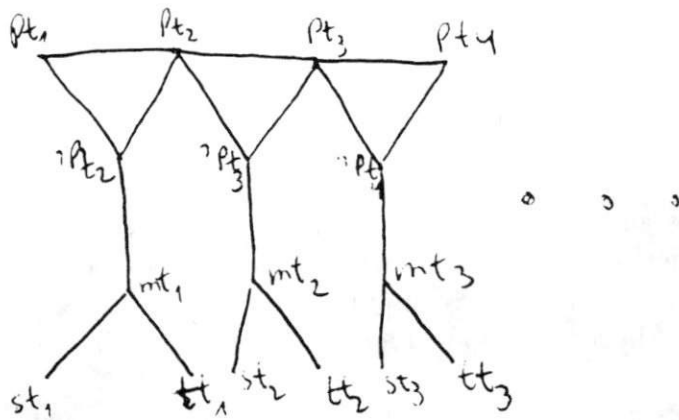


Figure 5: Constraints graph for Example 6.1

6.2 Cycle-cutset for default theories

The cycle-cutset algorithm is another method for solving CSPs that exploits special features 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 [Dec90].

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 c , existence, membership, and entailment can be decided in $O(n^2 * (n + 2^c) + |D| * d)$ steps when the theory is acyclic and $O(n^2(n^c + n + l^2) + |D| * (d + pl^2))$ steps when the theory is cyclic. \square*

7 Conclusions

Reiter's default logic is a useful formalism for nonmonotonic reasoning. However, the usefulness of default logic 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 presented a new semantics for default logic which clarifies the concept of extension, have shown a procedure that computes an extension for any finite propositional default theory, and have identified new tractable default theories.

Using the concept of *meta-interpretations*, we defined a model for a propositional default theory. We then showed 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 largely explored. For instance, in order to compute whether a formula is in every extension of a default theory, we need not 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 is 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 autoepistemic logic were reported in [MT91], where it was shown that the question of an atom's membership in every expansion of an autoepistemic theory can be reduced to propositional provability. Also, Elkan [Elk90] 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.

There have been attempts in the past to relate default logic to other forms of nonmonotonic reasoning systems, such as autoepistemic logic, circumscription, and TMS [Kon88, MT89, Eth87b, JK90]. We believe that embedding default logic in classical logic is just as valuable since classical logic is a very well-understood formalism supported by a large body of computational knowledge.

8 Proofs

8.1 Useful theorems and definitions

Definition 8.1 ([tL67]) 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 . \square

Definition 8.2 ([tL67]) If S is any set of clauses, then the n -th resolution of S , denoted by $R^n(S)$, is defined for $n \geq 0$ as follows: $R^0 = S$, and for $n \geq 0$, $R^{n+1}(S) = R(R^n(S))$. \square

Theorem 8.3 ([tL67]) Given a set S of clauses, if a clause c is a logical consequence¹³ of S , then for some $n \geq 0$, there exist a clause $c' \in R^n(S)$, such that $c' \subseteq c$. \square

Proposition 8.4 Suppose c, c_1, c_2, c'_1, c'_2 are clauses, $c'_1 \subseteq c_1$, $c'_2 \subseteq c_2$, and $c = \text{res}(c_1, c_2)$. 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 $c' \in \text{res}(c'_1, c'_2)$.

Proof: Suppose

$$\begin{aligned}c_1 &= c_3 \vee P \\c_2 &= c_4 \vee \neg P \\c &= c_3 \vee c_4\end{aligned}$$

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

$$\begin{aligned}c'_1 &= c_5 \vee P \\c'_2 &= c_6 \vee \neg P\end{aligned}$$

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 . \square

¹³We assume c is not a tautology.

Corollary 8.5 *If a meta-interpretation is deductively closed, then if it satisfies c_1, c_2 , it must also satisfy $res(c_1, c_2)$. \square*

Theorem 8.6 ([Rei80], Corollary 2.2) *A closed default theory (D, W) has an inconsistent extension iff W is inconsistent. \square*

Proposition 8.7 *Let θ be a deductively closed meta-interpretation, and suppose θ satisfies a set of clauses A , and $A \models c$. Then θ satisfies c .*

Proof: By Theorem 8.3, there is i such that for some $c' \in R^i(A)$, $c' \subseteq c$. We will prove by induction on this i that $\theta \models c'$, and hence θ satisfies c . For $i=0$, the assertion is clear. For $i > 1$, there must be $c_1, c_2 \in R^{i-1}(A)$ such that $c' = res(c_1, c_2)$. By the induction hypothesis, θ satisfies c_1 and c_2 . So by Corollary 8.5, θ satisfies c' . \square

8.2 Proofs

Proof of Proposition 3.7 (prime(Δ) is a closure) Let Δ be a default theory. $prime(\Delta)$ is a closure of Δ .

Proof: Suppose E is an extension of $\Delta = (D, W)$. 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 $prime(\Delta)$ such that $c' \subseteq c$. The proof will go by induction on $min(c)$, where $min(c)$ is the minimal i such that $c \in E_i$.

Case $min(c) = 0$: In this case, it must be that $c \in W$, and our claim must be true since $prime(\Delta)$ contains $PI(W)$.

Case $min(c) = 1$: In this case, it must be that either $W \models c$ or $c \in C_D$. In any case, the assertion clearly holds.

Assume the claim is true for $min(c) = n$, where $n > 0$, show that it is true for $m = n + 1$ ($m > 1$). From now on we assume that $c \neq \Lambda$, since by Lemma 8.6, if $c = \Lambda$ then $c \in PI(W)$, so $min(\Lambda)$ must be ≤ 1 .

Suppose c was introduced first at E_m . So either $c \in C_D$ or $E_n \models c$. If $c \in C_D$, then clearly our assertion holds. Assume $E_n \models c$. So by Theorem 8.3, for some j , there is $c' \in R^j(E_n)$ such that $c' \subseteq c$. We will show by induction on such a minimal j that there

is $c' \in \text{prime}(\Delta)$ 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 $c'' \in \text{res}(c_1, c_2)$. By the induction hypothesis, there are c'_1, c'_2 in $\text{prime}(\Delta)$ such that $c'_1 \subseteq c_1, c'_2 \subseteq c_2$. By Proposition 8.4, either $c'_1 \subseteq c''$ or $c'_2 \subseteq c''$ or there is c_3 in $\text{res}(c'_1, c'_2)$ such that $c_3 \subseteq c''$. Since $\text{min}(c) > 1$, it can't be that both c'_1 and c'_2 are in $PI(W)$, so $c_3 \in \rho(\Delta) - \Lambda \subseteq \text{prime}(\Delta)$. \square

Proof of Theorem 3.16 (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 $E = \{s | \theta \approx s\}$. \square

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 extension¹⁴ of Δ .

We define

1. $E_0 = W$,
2. For $i \geq 0$ $E_{i+1} = E_i^* \cup \{c | \alpha : \beta_1, \dots, \beta_n / c \in D \text{ where } \alpha \in E_i \text{ and } \neg\beta_1, \dots, \neg\beta_n \notin A \text{ and } c \in A\}$, 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 so by Theorem 1.1 A is an extension of Δ .

Let $c \in A$. By definition, c has a proof w.r.t. (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 . We will show that $\theta = (S, f')$ is a model of Δ , where f' is defined as

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

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

1. $E_0 = W$,

¹⁴w.l.g., we assume in this proof that an extension is a set of clauses.

2. For $i \geq 0$ $E_{i+1} = E_i^* \cup \{c \mid \alpha : \beta_1, \dots, \beta_n / c \in D \text{ where } \alpha \in E_i \text{ and } \neg\beta_1, \dots, \neg\beta_n \notin E\}$, and
3. $E = \bigcup_{i=0}^{\infty} E_i$.

By induction on the minimal i such that an arbitrary clause c belongs to E_i , we can show that c has a proof in θ and that θ is based on Δ . \square

Proof of Theorem 3.18 (indexing and proofs) A weak model $\theta = (S, f)$ for Δ is a model iff there is a function $\rho : S \rightarrow N^+$ 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 $\delta = \alpha : \beta_1, \dots, \beta_n / c \in D$ 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 $c' \subset c$ such that $\theta \models c'$ and $\rho(c') < \rho(c)$.

Proof: We can show that each atomic clause has a proof w.r.t. θ and Δ by induction on $\rho(c)$. \square

Proof of Theorem 3.20 (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 3.18. \square

Proof of Theorem 3.22 (minimality of models) A model for Δ is a minimal weak model for Δ .

Suppose that $\theta = (S, f)$ is a model for Δ . It is obviously a weak model. We want to show that it is minimal. By definition, for each atomic clause c in S there is a proof of c w.r.t. 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

$$A^- = \{c | c \text{ is atomic, } f'(c) = \mathbf{true}\}$$

$$A = \{c | c \text{ is atomic, } f(c) = \mathbf{true}\}$$

We will show that if c has a proof w.r.t. Δ and θ , it must be satisfied by θ' , and so $A \subseteq A^-$ — a contradiction. The proof will go 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 case the proof of c uses the defaults $\delta_1, \dots, \delta_{n+1}$, we observe, using the induction hypothesis, that $(W \cup \{\text{concl}(\delta_1), \dots, \text{concl}(\delta_n)\})^*$ is satisfied by θ' . So since θ' must satisfy Δ , it must satisfy $\text{concl}(\delta_{n+1})$ as well, and since it is deductively closed, it must satisfy $W \cup \{\text{concl}(\delta_1), \dots, \text{concl}(\delta_{n+1})\}^*$, so it satisfies c . \square

A 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: $L' = \{P, P = 1, P = 2, \dots, P = m | P \in L\}$, where $P = i$ for $i = 1, \dots, m$ denote propositional letters with the intuition "P will get the number i" behind it. For each P in L' , let N_P be the following set of sentences :

$$\begin{aligned} & P = 1 \vee P = 2 \vee \dots \vee P = m \\ P = 1 & \longrightarrow [\neg(P = 2) \wedge \neg(P = 3) \wedge \dots \wedge \neg(P = m)] \\ P = 2 & \longrightarrow [\neg(P = 3) \wedge \neg(P = 4) \wedge \dots \wedge \neg(P = m)] \\ & \quad \cdot \\ & \quad \cdot \\ & \quad \cdot \\ & P = m - 1 \longrightarrow \neg(P = m). \end{aligned}$$

The set N_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 sentences:

$$\begin{aligned}
 &P = 1 \wedge Q = 2, P = 1 \wedge Q = 3, \dots, P = 1 \wedge Q = m \\
 &P = 2 \wedge Q = 3, \dots, P = 2 \wedge Q = m \\
 &\quad \cdot \\
 &\quad \cdot \\
 &\quad \cdot \\
 &P = m - 1 \wedge Q = m.
 \end{aligned}$$

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

Acknowledgements

We thank Gerhard Brewka and Kurt Konolige for very useful comments on an earlier version of this paper and Judea Pearl for fruitful discussions.

References

- [BF87] N. Bidoit and C. Froidevaux. Minimalism subsumes default logic and circumscription in stratified logic programming. In *LICS-87: Proceedings of the IEEE symposium on logic in computer science*, pages 89–97, Ithaca, NY, USA, June 1987.
- [Dec90] Rina Dechter. Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition. *Artificial Intelligence*, 41:273–312, 1990.
- [DP88] Rina Dechter and Judea Pearl. Network-based heuristics for constraint satisfaction problems. *Artificial Intelligence*, 34:1–38, 1988.
- [DP89] Rina Dechter and Judea Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 38:353–366, 1989.

- [Elk90] Charles Elkan. A rational reconstruction of nonmonotonic truth maintenance systems. *Artificial Intelligence*, 43:219-234, 1990.
- [Eth87a] David W. Etherington. Formalizing nonmonotonic reasoning systems. *Artificial Intelligence*, 31:41-85, 1987.
- [Eth87b] David W. Etherington. Relating default logic and circumscription. In *IJCAI-87: Proceedings of the 10th international joint conference on artificial intelligence*, pages 489-494, Detroit, MI, USA, August 1987.
- [Fre85] E.C. Freuder. A sufficient condition for backtrack-bounded search. *J. ACM*, 32(4):755-761, 1985.
- [GL91] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. *New Generation Computing*, 9:365-385, 1991.
- [Got] George Gottlob. Complexity results for nonmonotonic logics. In the working notes of the 4th International Workshop on Nonmonotonic Reasoning, Plymouth, Vermont, May 1992.
- [JK90] Ulrich Junker and Kurt Konolige. Computing the extensions of autoepistemic and default logics with a TMS. In *AAAI-90: Proceedings of the 8th National Conference on Artificial Intelligence*, Boston, MA, 1990.
- [KH92] R.A. Kowalski and C.J. Hogger. Logic programming. In Stuart C. Shapiro, editor, *Encyclopedia of Artificial Intelligence*, pages 873-891. John Wiley & Sons, 2nd edition, 1992.
- [Kon88] Kurt Konolige. On the relation between default and autoepistemic logic. *Artificial Intelligence*, 35:343-382, 1988.
- [KS91] Henry A. Kautz and Bart Selman. Hard problems for simple default logics. *Artificial Intelligence*, 49:243-279, 1991.
- [MR72] Eliana Minicozzi and Ray Reiter. A note on linear resolution strategies in consequence-finding. *Artificial Intelligence*, 3:175-180, 1972.

- [MT89] Wiktor Marek and Mirosław Truszczyński. Relating autoepistemic and default logic. In Ronald J. Brachman, Hector J. Levesque, and Raymond Reiter, editors, *KR-89: Proceedings of the first international conference on principles of knowledge representation and reasoning*, pages 276–288, San Mateo, CA, 1989. Morgan Kaufmann.
- [MT91] Wiktor Marek and Mirosław Truszczyński. Computing intersection of autoepistemic expansions. In *Logic Programming and Non-monotonic Reasoning: Proceedings of the 1st International workshop*, pages 37–50, Washington, DC USA, July 1991.
- [PP90] Halina Przymusińska and Teodor Przymusiński. Semantic issues in deductive databases and logic programs. In R. B. Banerji, editor, *Formal Techniques in Artificial Intelligence: A sourcebook*, pages 321–367. North-Holland, New York, 1990.
- [RdK87] Raymond Reiter and Johan de Kleer. Foundations of assumption-based truth maintenance systems: Preliminary report. In *The national conference on AI*, pages 183–188, Seattle, WA, July 1987.
- [Rei80] Raymond Reiter. A logic for default reasoning. *Artificial Intelligence*, 13:81–132, 1980.
- [Sti90] Jonathan Stillman. It's not my default: The complexity of membership problems in restricted propositional default logics. In *AAAI-90: Proceedings of the 8th national conference on artificial intelligence*, pages 571–578, Boston, MA, 1990.
- [Sti92] Jonathan Stillman. The complexity of propositional default logics. In *AAAI-92: Proceedings of the 10th national conference on artificial intelligence*, pages 794–799, San Jose, CA, 1992.
- [Tar72] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM Journal on Computing*, 1:146–160, 1972.
- [tL67] Char tung Lee. *A completeness theorem and a Computer Program for finding theorems derivable from given axioms*. PhD thesis, University of California, Berkeley, 1967.

- [Tou84] David S. Touretzky. Implicit ordering of defaults in inheritance systems. In *AAAI-84: Proceedings of the national conference on artificial intelligence*, pages 322-325, Austin, TX, 1984.
- [TY84] Robert E. Tarjan and M. Yannakakis. Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs and selectively reduce acyclic hypergraphs. *SIAM Journal on Computing*, 13(3):566-579, 1984.

MAY 27 1993

UC IRVINE LIBRARY



3 1970 01005 6130