Inference in Inheritance Networks using Propositional Logic and Constraint Networks Techniques

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

Abstract

This paper focuses on *network default theories*. Etherington [Etherington, 1987] has established a correspondence between inheritance networks with exceptions and a subset of Reiter's default logic called network default theories, thus providing a formal semantics and a notion of correct inference for such networks. We show that any such propositional network default theory can be compiled in polynomial time into a classical propositional theory such that the set of models of the latter coincides with the set of extensions of the former. We then show how constraint satisfaction techniques can be used to compute extensions and to identify tractable network default theories. For any propositional network theory, our algorithms compute all its extensions and verifies if a given conclusion is in one or all extensions.

1 Introduction

Research in multiple inheritance networks has focused on two main issues: developing fast algorithms that will operate on the network links to produce conclusions that match our intuition, and providing formal semantics for such networks. Clearly, the second is crucially important for adequate evaluation of the correctness of the first.

Etherington [Etherington, 1987] had approached the semantic issue by formalizing inheritance networks, called *network default theories*, within Reiter's default logic. While his framework has been criticized for demanding all exceptions be listed explicitly, his approach is still valuable in that it embeds the notion of inheritance within this general and widely studied framework of default logic.

Our paper focuses on the computational aspects of such network theories. We first present a necessary and sufficient condition for their coherence, namely, for deciding whether or not they have an extension. Then, using constraint satisfaction techniques, we present effective schemes for computing the extensions for *any* such network. In contrast, Etherington's procedure is only applicable to a subclass of networks theories called "ordered network theories". Moreover, the complexity of our schemes is related to the sparseness of the networks, as captured by the parameter of *induced width*.

The approach leading to these results has already been applied to the subclass of propositional disjunction-free semi-normal default theories [Ben-Eliyahu and Dechter, 1991a]. We have shown there that any such default theory can be compiled in polynomial time into a propositional theory, such that each of its models corresponds to an extension of the default theory. Constraint network techniques are then applied to compute extensions and to identify, analyze and solve tractable subclasses of this default logic. A generalization of this approach to network theories requires allowing size-two disjunctions in the default theory.

Our results pave the way for applying constraint networks techniques to logic programming as well since it has been shown that there is a one-to-one correspondence between stable models of logic programs and extensions of each of their default interpretations [?]. Elkan [Elkan, 1990] has also shown that stable models of a logic program with no classical negation can be represented as models of propositional logic.

The paper is organized as follows: in section 2 we briefly introduce default logic and inheritance networks. In section 3 we describe how tasks of default theories are mapped into equivalent tasks in propositional logic. This mapping is exploited in section 4 where we present new procedures for query processing and identify tractable classes using constraint networks techniques. Section 5 provides concluding remarks. Due to space considerations all proofs are omitted. For more details see [Ben-Eliyahu and Dechter, 1991b].

2 Default logic and inheritance networks

2.1 Reiter's default logic

Following is a brief introduction to Reiter's default logic [Reiter, 1980]. Let $\mathcal{L}$ be a first order language. A *default theory* is a pair $\Delta = (D, W)$, where D is a set of defaults and W is a set of closed wffs (well formed formulas) in $\mathcal{L}$. A *default* is a rule of the form $\alpha : \beta_1, ..., \beta_n / \gamma$, where $\alpha, \beta_1, ...\beta_n$ and γ are formulas in $\mathcal{L}$. The intuition behind a default can be: 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 can be deduced from W using the defaults in D. Let $Th(E)$ denote the logical closure of E in $\mathcal{L}$. We use the following definition of an extension:

Definition 2.1 *([Reiter, 1980],theorem 2.1) Let $E \subseteq \mathcal{L}$ be a set of closed wffs, and let $\Delta = (D, W)$ be a closed default theory. Define*[1]

- $E_0 = W$
- *For $i \geq 0$ $E_{i+1} = Th(E_i) \bigcup \{\gamma | \alpha : \beta_1, ..., \beta_n/\gamma \in D$ where $\alpha \in E_i$ and $\neg\beta_1, ...\neg\beta_n \notin E\}$.*

E is an extension for Δ iff for some ordering $E = \bigcup_{i=0}^{\infty} E_i$ □

Most 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 formulas S, Is S contained in some extension of Δ?

Set-Entailment: Given a set of formulas S, Is S contained in every extension of Δ?

In this paper we show how, for a subclass called "network default theories", the above queries can be reduced to propositional satisfiability.

2.2 Inheritance networks and network default theories

The following brief introduction is adopted from [Etherington, 1987] and [Touretzky, 1984].

Inheritance networks are 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.

Usually, the inheritance network is a directed graph whose nodes represent individuals and abstractions ("classes"), and whose arcs denote relations between those nodes. The most common relations are "IS-A" and "ISN'T-A".

Consider the following information:

- Mammals are warm-blooded.
- Dolphins are mammals.
- Flipper is a dolphin.

This information can be encoded in the inheritance network shown in figure 1 (where a solid arrow represents an "IS-A" relation). A reasonable conclusion would be that Flipper is warm-blooded.

When exceptions to inheritance are allowed, the inference in those systems becomes non-monotonic, namely, conclusions might change in light of new evidence. Suppose we start with the following set of axioms:

[1] Note the appearance of E in the formula for E_{i+1}.

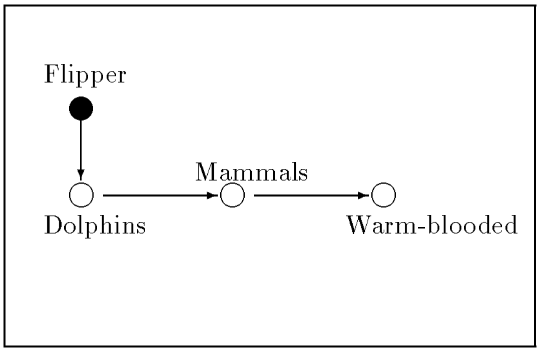

Figure 1: An inheritance network with no exceptions

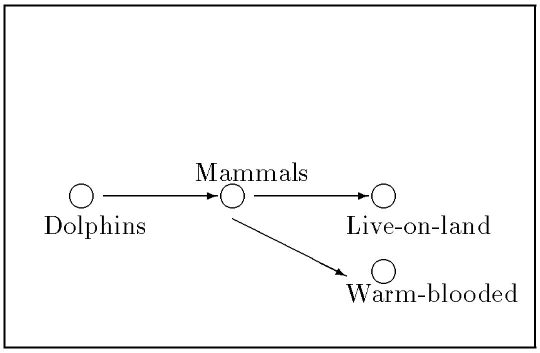

Figure 2: An inheritance network with exceptions

- Mammals are warm-blooded.
- Mammals live on land.
- Dolphins are mammals.
- Dolphins do not live on land.

This is an example of an inheritance network with exception: dolphins are mammals who live in the water. The network in figure 2 represents this knowledge ("canceled" arrows denote "ISN'T-A" relation). Given the information that Flipper is a mammal, we will conclude that he lives on land, but the additional evidence that he is a dolphin will force us to retract that conclusion and adopt the belief that he does not live on land[2].

Etherington [Etherington, 1987] proposed a subclass of default theories called "network default theories" (in short, "network theories") as suitable to provide formal semantics and a notion of sound inference for those networks:

Definition 2.2 *(*Network default theory*) [Etherington, 1987] A default theory Δ is a* network default theory *iff it satisfies the following conditions:*

- *W contains only:*
  - *literals (i.e atomic formulae or their negations), and*
  - *disjuncts of the form $(\alpha \vee \beta)$ where α and β are literals.*
- *D contains only normal and seminormal defaults of the form: $\alpha : \beta/\beta$ or $\alpha : \beta \wedge \gamma_1 \wedge ... \wedge \gamma_n/\beta$ where α, β and γ_i are literals.* □

[2] This conclusion is supported by the convention that features of a subclass override those of a super-class.

Etherington suggests a way to formalize inheritance relations in network theories. His translation is as follows:

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

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 $\forall x.A(x) \longrightarrow \neg B(x)$. We will translate this link to the propositional rule schema $A(x) \longrightarrow \neg B(x)$.

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 $C_1, ..., C_n$". This translates to the default rule schema $A(x) : B(x) \wedge \neg C_1(x) \wedge ... \wedge \neg C_n(x)/B(x)$
$(A(x) : \neg B(x) \wedge \neg C_1(x) \wedge ... \wedge \neg C_n(x)/\neg B(x)$)

Example 2.3 *The inheritance network in figure 2 will be translated to the following network theory:*

$$D = \{\frac{Mammal(x) : Lives\text{-}on\text{-}land(x) \wedge \neg Dolphine(x)}{Lives\text{-}on\text{-}land(x)}, \frac{Dolphine(x) : \neg Lives\text{-}on\text{-}land(x)}{\neg Lives\text{-}on\text{-}land(x)}\}$$

$$W = \{\ Dolphine(x) \longrightarrow Mammal(x), Mammal(x) \longrightarrow Warm\text{-}blooded(x)\ \} \ \square$$

An extension of a network 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 theories. Etherington has a nondeterministic procedure to compute an extension of a default theory. If the theory is what he calls an *ordered* network theory, then his procedure is guaranteed to produce an extension.

In the sequel we will show a procedure that computes all extensions for *any propositional* network theory. In fact, we deal with a superclass in which the prerequisite of a default is a conjunction of literals rather than just a single literal. We will assume, w.l.g., that W is consistent, since when W is inconsistent, the extension is the inconsistent one. We also assume w.l.g. that each default has a single literal as a consequent.

3 Definitions and preliminaries

We denote propositional symbols by upper case letters $P, Q, R...$, propositional literals (i.e. $P, \neg P$) by lower case letters $p, q, r...$, clauses by $c_1, c_2, ...$. The number of literals in the clause c is denoted by $|c|$.

The operator $\sim$ over literals is defined as follows: If $p = \neg Q$, $\sim p = Q$, If $p = Q$ then $\sim p = \neg Q$. If $\delta = \alpha : \beta/\gamma$ is a default, we define $\text{pre}(\delta) = \alpha$, $\text{just}(\delta) = \beta$ and $\text{concl}(\delta) = \gamma$.

Given a set of formulas S and a formula ω, $S \vdash \omega$ means that ω is provable from premises S, and $S \models \omega$ means that S entails ω - i.e. that every model of S satisfies ω as well. For propositional formulas, $S \vdash \omega$ iff $S \models \omega$, hence we will use these notations interchangeably.

The *logical closure* of a set of formulas S is the set $\{\omega | S \vdash \omega\}$. We denote by $Th(S)$ the *logical closure* of a set of formulas S.

An extension of a default theory is a logically closed set of formulas. How do we compute the logical closure of a set of clauses? Since the logical closure is an infinite set, we will not be able to compute the closure in a finite time. However, if the initial set of clauses is finite, we can compute a set which will represent the logical closure using the notion of *prime implicants* as presented by Reiter and de Kleer [Reiter and de Kleer, 1987]:

Definition 3.1 *A prime implicant of a set S of clauses is a clause c such that*

1. *$S \models c$, and*
2. *there is no proper subset c' of c such that $S \models c'$.*

Given a set of formulas S, S^+ will denote the set of its prime implicants. As Reiter and de Kleer note, a brute force method of computing S^+ is to repeatedly resolve pairs of clauses of S, add the resolvents to S, and delete subsumed clauses, until a fixed point is reached[3]. There are some improvements to that method, but it is clear that the general problem is NP-Hard since it also solves satisfiability. Nevertheless, for size-2 clauses the prime implicants can be computed in polynomial time since a resolvent of two clauses of size ≤ 2 is also of size ≤ 2.

The following proposition suggests that for network theories it is enough to consider extensions of a network theory containing clauses of size one or two only:

Proposition 3.2 *Let E^* be an extension of a network theory, and let $E' = \{c | c \in E^*, |c| \leq 2\}$. Then E' contains all prime implicants of E^*.* $\square$

We say that a set of clauses E *satisfies the preconditions of* δ if $\text{pre}(\delta) \in \text{Th}(E)$ and the negation of $\text{just}(\delta)$ is not in $\text{Th}(E)$. We say that E *satisfies a default* δ if it does not satisfy the preconditions of δ or else, it satisfies its preconditions and $\text{Th}(E)$ contains its conclusion.

A *proof of a clause* c w.r.t. a given set of clauses E and a given network theory $\Delta = (D, W)$ is a sequence of rules $\delta_1, ..., \delta_n$, $n \geq 0$, such that the following three conditions hold:

1. $c \in \text{Th}(W \bigcup \{\text{concl}(\delta_1), ..., \text{concl}(\delta_n)\})$.
2. For all $1 \leq i \leq n$, the negation of $\text{just}(\delta_i)$ is not in $\text{Th}(E)$.
3. For all $1 \leq i \leq n$, $\text{pre}(\delta_i)$ is a subset of $\text{Th}(W \bigcup \{\text{concl}(\delta_1), ..., \text{concl}(\delta_{i-1})\})$.

[3] It is clear that this method will not generate all the tautologies, but it is easy to handle this exception.

The following lemma is instrumental throughout the paper:

Lemma 3.3 *Th(E) is an extension of a network theory Δ iff Th(E)is a logical closure of a set of clauses E that satisfies:*

1. *$W \subseteq E$*
2. *E satisfies each rule in D.*
3. *For each clause $c \in E$, there is a proof of c in E.* $\square$

We define the *dependency graph* $G_{(D,W)}$ of a network theory Δ to be a directed graph constructed as follows: Each literal p appearing in D or in W is associated with a node, and an edge is directed from p to r iff there is a default rule where p appears in its prerequisite and r is its consequent or there is a clause $p \longrightarrow r$ in W. An *acyclic network theory* is one whose dependency graph is acyclic, a property that can be tested linearly.

4 Compiling a network theory into a propositional theory

In this section we show how we can compile a given network theory Δ into a propositional theory $\mathcal{P}_\Delta$ such that $\mathcal{P}_\Delta$ has a model iff Δ has an extension, and vice-versa, every model of $\mathcal{P}_\Delta$ has a corresponding extension for Δ.

The common approach for building an extension, (used by Etherington [Etherington, 1987], Kautz and Selman [Kautz and Selman, 1991], and others), is to increment W using rules from D. We make a declarative account of this process by formulating the conditions of lemma 3.3 as a set of constraints that the default theory impose on the set of its extensions. This frees us from worrying about ordering, however, it requires adding a constraint guaranteeing that if a formula is in the extension, then it has a non-circular proof. To enforce this restriction, we associate an *index variable* with each literal in the transformed language, and require that p is in the extension only if it is the consequent of a rule whose prerequisite's indexes are smaller. Elkan [Elkan, 1990] used the same technique to insure that the justifications supporting a node in a TMS are noncircular.

Let $\#p$ stand for the "index associated with p", and let k be its number of values. These "multi-valued variables" (as opposed to propositional variables which are bi-valued) can be expressed in propositional logic using additional $O(k^2)$ clauses and literals (see [Ben-Eliyahu and Dechter, 1991b]). For simplicity, however, we will use the multi-variable notations, viewing them as abbreviations to their propositional counterparts.

Let $\mathcal{L}$ be the underlying propositional language of Δ. For each propositional symbol in $\mathcal{L}$, we define two propositional symbols, I_P and $I_{\neg P}$. For each pair of literals p and q in $\mathcal{L}$, we define the symbol $I_{p\vee q}$. We get a new set of symbols : $\mathcal{L}' = \{I_P, I_{\neg P} | P \in \mathcal{L}\} \bigcup \{I_{p\vee q} | p, q \in \mathcal{L}\}$. Intuitively, I_P stands for "P is in the extension", $I_{\neg P}$ stands for "$\neg P$ is in the extension", and $I_{p\vee q}$ means that "$p \vee q$ is in the extension". For notational convenience I_p and $I_{p\vee p}$ will stand for the same propositional letter (same for $I_{\sim p \vee q}$ and $I_{p \longrightarrow q}$).

> Procedure translate-1(Δ)
>
> 1. Compute W^+, the set of prime implicants of W.
> 2. For each $c \in W^+$ put I_c into $\mathcal{P}_\Delta$.
> 3. For each $p \longrightarrow q$ in W add $I_p \longrightarrow I_q$ into $\mathcal{P}_\Delta$.
> 4. For each $\alpha : \beta / p \in D$, add $in(\alpha) \wedge cons(\beta) \longrightarrow I_p$ to $\mathcal{P}_\Delta$.
> 5. For each $p \notin W^+$ do the following :
>   Let $C_p = \{[in(q_1 \wedge q_2 ... \wedge q_n) \wedge cons(\beta)]$
> $\wedge [\#q_1 < \#p] \wedge ... \wedge [\#q_n < \#p]$
> $| \exists \delta \in D$ such that $\delta = q_1 \wedge q_2 ... \wedge q_n : \beta / p \}$.
>   Let $L_p = \{[in(q) \wedge [\#q < \#p]] \, | q \longrightarrow p \in W^+\}$.
>   Let $S_p = C_p \bigcup L_p$.
>   If S_p is not empty then add to $\mathcal{P}_\Delta$ the formula $I_p \longrightarrow [\vee_{\alpha \in S_p} \alpha]$.
>   Else, if $S_p = \emptyset$ add $\neg I_p$ to $\mathcal{P}_\Delta$.
> 6. For each $p \vee q \notin W^+$, $p \neq q$, add $\neg I_{p\vee q}$ into $\mathcal{P}_\Delta$.

Figure 3: Algorithm to compile a network theory into a propositional theory

To further simplify the notation we use the notions of $in(\omega)$ and $cons(\omega)$ that stand for "ω is in the extension", and "ω is consistent with the extension", respectively. Formally, $in(\omega)$ and $cons(\omega)$ are defined as follows:

- if $\omega = p$ then $in(\omega) = I_p$, $cons(\omega) = \neg I_{\sim p}$.
- if $\omega = p \vee q$ then $in(\omega) = I_{p\vee q}$.
- if $\omega = p_1 \wedge p_2 \wedge ... \wedge p_n$, then $in(\omega) = in(p_1) \wedge in(p_2) \wedge ... \wedge in(p_n)$, $cons(\omega) = \wedge_{i,j \in \{1,...,n\}} \neg I_{\sim p_i \vee \sim p_j}$. (Note that $p_1 \wedge ... \wedge p_n$ is "consistent with the extension" iff $\neg[p_1 \wedge ... \wedge p_n]$ is not in the extension iff (since all prime implicants are of size ≤ 2) for all i, j, $\sim p_i \vee \sim p_j$ is not in the extension.)

Procedure **translate-1** in figure 3 compiles any network theory over $\mathcal{L}$ into a propositional theory over $\mathcal{L}''$. This translation requires adding n index variables, n being the number of literals in $\mathcal{L}$, each having at most n values. Since expressing an *inequality* in propositional logic requires $O(n^2)$ clauses, and since there are at most n possible inequalities per default, the resulting size of this transformation is bounded by $O(|D|n^3)$ propositional sentences. Note also that the complexity of generating W^+ is at most $O(n^3)$.

The following theorems summarize the properties of our transformation. In all of them, $\mathcal{P}_\Delta$ is the set of sentences resulting from translating a given network theory Δ using translate-1.

Theorem 4.1 *Let Δ be a network theory. Suppose $\mathcal{P}_\Delta$ is satisfiable and θ is a model for $\mathcal{P}_\Delta$, and let $E = \{c | \theta(I_c) = \textbf{true}\}$.*
Then:

1. *E contains all its prime implicants.*
2. *Th(E) is an extension of Δ.* $\square$

Theorem 4.2 *Let Th(E) be an extension for* Δ. *Then there is a model* θ *for* $\mathcal{P}_\Delta$ *such that* $\theta(I_c) = \textbf{true}$ *iff* $c \in Th(E)$ *and* $|c| \leq 2$. □

The above two theorems suggest a necessary and sufficient condition for the coherence of a network theory:

Corollary 4.3 *A network theory* Δ *has an extension iff* $\mathcal{P}_\Delta$ *is satisfiable.* □

For the next corollaries, we define for each clause c a formula prime(c) as follows: if $c = p_1 \vee p_2 \vee \ldots \vee p_n$ prime(c) $= [\vee_{i,j \in \{1 \ldots n\}} I_{p_i \vee p_j}]$

Corollary 4.4 *A set of clauses, C, is contained in an extension of* Δ *iff there is a model for* $\mathcal{P}_\Delta$ *which satisfies the set* $\{prime(c) | c \in C\}$.

Corollary 4.5 *A clause c is in every extension of a network theory* Δ *iff* $\mathcal{P}_\Delta \models prime(c)$. □

These theorems suggest that we can first translate a given network theory Δ to $\mathcal{P}_\Delta$ and then answer queries as follows: to test if Δ has an extension, we test satisfiability of $\mathcal{P}_\Delta$; to see if a set S of clauses is a member in some extension, we test satisfiability of $\mathcal{P}_\Delta \bigcup \{\text{prime}(c) | c \in S\}$; and to see if S is included in every extension, we test if $\mathcal{P}_\Delta$ entails the formula $[\wedge_{c \in S} \text{prime}(c)]$.

Example 4.6

Consider again the network theory from example 2.3 together with the evidence that Flipper is a mammal (predicates are abbreviated by their initials; parameters are omitted since Flipper is the only individual):

$D = \{M : L \wedge \neg D / L,\ D : \neg L / \neg L\}$
$W = \{D \longrightarrow M, M \longrightarrow Wb, M\}$

This is an acyclic network theory, thus no indices are required. When translating Δ to $\mathcal{P}_\Delta$ we get:

$W^+ = W \bigcup \{Wb, D \longrightarrow Wb\}$
$\mathcal{P}_\Delta = \{$
following step 2:
$I_{D \longrightarrow M}, I_{M \longrightarrow Wb}, I_{Wb}, I_M, I_{D \longrightarrow Wb}$
following step 3:
$I_D \longrightarrow I_M,\ I_M \longrightarrow I_{Wb},\ I_D \longrightarrow I_{Wb}$
following step 4:
$I_M \wedge \neg I_{\neg L} \wedge \neg I_D \wedge \neg I_{\neg L \vee D} \longrightarrow I_L,$
$I_D \wedge \neg I_L \longrightarrow I_{\neg L}$
following step 5:
$I_L \longrightarrow I_M \wedge \neg I_{\neg L} \wedge \neg I_D \wedge \neg I_{\neg L \vee D},$
$I_{\neg L} \longrightarrow I_D \wedge \neg I_L,\ \neg I_{\neg M},\ \neg I_D,\ \neg I_{\neg D},\ \neg I_{\neg Wb}$
following step 6:
$\{\neg I_{x \longrightarrow y} | x \longrightarrow y \notin \{D \longrightarrow M, M \longrightarrow Wb,$
$D \longrightarrow Wb\}\}$

This set of sentences has only one model in which all and only the following literals are true:

$$I_M, I_{Wb}, I_L, I_{D \longrightarrow M}, I_{M \longrightarrow Wb}, I_{D \longrightarrow Wb}$$

which correspond to the extension

$$Th(\{M, Wb, L, D \longrightarrow M, M \longrightarrow Wb\})$$

Example 4.7

Suppose we add the information that Flipper is a dolphin to what we knew in the previous example. This amounts to adding the proposition D to W. So we have to take $\neg I_D$ out of $\mathcal{P}_\Delta$ and add I_D to $\mathcal{P}_\Delta$. The model for $\mathcal{P}_\Delta$ is:

$$I_D, I_M, I_{Wb}, I_{\neg L}, I_{D \longrightarrow M}, I_{M \longrightarrow Wb}, I_{D \longrightarrow Wb}$$

which corresponds to the extension

$$Th(\{D, M, Wb, \neg L, D \longrightarrow M, M \longrightarrow Wb\})$$

which is the only extension.

4.1 An improved translation

Procedure translate-1 can be improved. If a prerequisite of a rule is not on a cycle with its consequent, we do not need to index them, nor enforce the partial order among their indices. Thus, we need indices only for literals which reside on cycles in the *dependency graph*. Furthermore, since we will never have to solve cyclicity between two literals that do not share a cycle, the range of the index variables is bounded by the maximum number of literals that share a common cycle. In fact, we show that the index variable's range can be bounded by the maximal length of an acyclic path in any *strongly connected component* in $G_{(D,W)}$. 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. This improvement is discussed in detail in [Ben-Eliyahu and Dechter, 1991b].

5 Tractable network default theories

Processing the network theory using our approach requires two steps: first, compile the default theory into a propositional theory, and then solve satisfiability. We have shown that the first step is tractable. The second step, however, is known to be NP-complete in general. In this section we show how propositional satisfiability can be regarded as a constraint satisfaction problem, and how techniques borrowed from that field can be used to solve satisfiability and to identify tractable subsets of propositional and network theories.

In general, constraint satisfaction techniques exploit the structure of the problem through the notion of a "constraint graph". For a propositional theory, the constraint graph (also called a "primal constraint graph") associates a node with each propositional letter and connects any two nodes whose associated letters appear in the same clause. Various graph parameters have been shown as crucially related 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*. It can be shown that the worse-case complexity of deciding consistency is polynomially bounded by any one of these parameters.

Since these parameters can be bounded easily by simple processing of the given graph, they can be used for assessing tractability ahead of time. For instance,

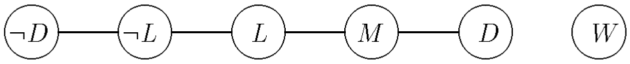

Figure 4: Interaction graph for the theory presented in example 4.6

when the constraint graph is a tree, satisfiability can be determined in linear time. In [Ben-Eliyahu and Dechter, 1991a] we have demonstrated the potential of this approach using one specific technique called *Tree-Clustering* [Dechter and Pearl, 1989], customized for solving propositional satisfiability, and emphasized its effectiveness for maintaining a default database. We have also characterized the tractability of the default theories as a function of the *induced width*[4 5], w^*, of their *interaction graph*. We next generalize those results for network theories:

The *interaction graph* of a network theory is an undirected graph where each literal in the theory is associated with a node, and for each p and for every $\delta = \alpha : \beta / p$ in D, every $q \in \alpha$ and every $\sim r$ such that $r \in \beta$, there are arcs connecting all of them into one clique with p. Also, for each $p \longrightarrow q$ in W, there is an arc between p and q.

Theorem 5.1 *A network theory whose interaction graph has an induced width w^* can decide existence, membership and entailment in $O(n * 2^{w^*+1})$ when the theory is acyclic and $O(n^{w^*+2})$ when the theory is cyclic.* □

Example 5.2

Consider the set $\mathcal{P}_\Delta$ generated in example 4.6. The interaction graph is as shown in figure 4 (isolated nodes are omitted). This graph is already chordal, and if we take the ordering $\neg D, \neg L, L, M, D, W$ we see that $w^* \leq 3$, and so this network theory belongs to a class of networks for which the queries we posed can be answered in time bounded by exp(4). According to Stillman's classification [Stillman, 1990] this network theory belongs to a class whose membership problem is NP-complete. □

6 Summary and conclusions

We have presented a necessary and sufficient condition for coherence of propositional inheritance theories, provided a procedure that computes an extension, and identified tractable subsets of network default theories. The algorithm handles membership and entailment queries as well. Specifically, we have shown that network theories whose topologies have bounded induced width can be processed in polynomial time w.r.t. this parameter.

Our approach is to compile a network theory into a propositional theory such that the set of models of the latter coincides with the set of extensions of the former. Consequently, questions of coherence, membership and entailment on the network theory are equivalent to propositional satisfiability. This brings problems in non-monotonic reasoning into the familiar arenas of both propositional satisfiability and constraint satisfaction problems. Although we use here a two-step translation (from inheritance networks to default theories and then to propositional theories), it is easy to see that we can translate the inheritance network directly into a propositional theory.

Our work adds to previous research on network theories and inheritance reasoning. Etherington [Etherington, 1987] has shown a sufficient condition for coherence and presented a procedure that computes an extension of *ordered* network theories only. Stillman [Stillman, 1990] has shown that the membership problem for propositional network theories is NP-Complete and claimed to have polynomial algorithms for solving membership of a single literal in restricted subsets of network theories.

In the future we intend to extend our approach to handle preferred extensions, as formulated by Etherington and Touretzky [Touretzky, 1984], namely, to use only normal default rules, and define a partial order on the proof sequences. Using constraint network techniques, we hope to show that a most preferred extension can be obtained with the same complexity as those for finding an arbitrary one. In [Ben-Eliyahu and Dechter, 1991b] we show how this approach can be applied to any default theory.

Acknowledgements

This work was supported in part by Air Force Office of Scientific Research, AFOSR 900136 and by NSF grant IRI-9157936. The authors wish to thank Judea Pearl for many useful discussions.

References

[Ben-Eliyahu and Dechter, 1991a] Rachel Ben-Eliyahu and Rina Dechter. Default logic, propositional logic and constraints. In *AAAI-91: Proceedings of the 9th national conference on artificial intelligence*, pages 379–385, Anaheim, CA, USA, July 1991.

[Ben-Eliyahu and Dechter, 1991b] Rachel Ben-Eliyahu and Rina Dechter. Propositional semantics for default logic. Technical Report R-172, Cognitive Systems Lab, UCLA, 1991. This paper will be presented at the 4th international workshop on nonmonotonic reasoning, May 1992, Plymouth, Vermont. This paper is a generalization of the paper published in the proceedings of AAAI-91.

[4] The *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. The induced width is the width of the graph after it was completed to be a chordal graph.

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

[Dechter and Pearl, 1989] Rina Dechter and Judea Pearl. Tree clustering for constraint networks. *Artificial Intelligence*, 38:353–366, 1989.

[Elkan, 1990] Charles Elkan. A rational reconstruction of nonmonotonic truth maintenance systems. *Artificial Intelligence*, 43:219–234, 1990.

[Etherington, 1987] David W. Etherington. Formalizing nonmonotonic reasoning systems. *Artificial Intelligence*, 31:41–85, 1987.

[Kautz and Selman, 1991] Henry A. Kautz and Bart Selman. Hard problems for simple default logics. *Artificial Intelligence*, 49:243–279, 1991.

[Reiter and de Kleer, 1987] 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.

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

[Stillman, 1990] 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, USA, 1990.

[Touretzky, 1984] David S. Touretzky. Implicit ordering of defaults in inheritance systems. In *AAAI-84: Proceedings of the 2nd national conference on artificial intelligence*, pages 322–325, Austin,TX, 1984.