Computer Science Department Technical Report
University of California
Los Angeles, CA 90024-1596

PROPOSITIONAL SEMANTICS FOR DEFAULT LOGIC

R. Ben-Eliyahu
R. Dechter

August 1991
CSD-910033

Propositional Semantics for Default Logic (Extended Abstract)

Rachel Ben-Eliyahu
Cognitive Systems Laboratory
Computer Science Department
University of California
Los-Angeles, California 90024
rachel@cs.ucla.edu

Rina Dechter
Information & Computer Science
University of California
Irvine, California, 92717
dechter@ics.uci.edu

August 14, 1991

1 Introduction

Reiter's default logic [Rei80] is one of three leading formalisms for nonmonotonic reasoning, and is perhaps the simplest to state and the one most compatible with logic programs. There are, however, two major obstacles to the practicality of default logic: the first is the lack of intuitive semantics for the set of conclusions that the logic ratifies, and the second is the high computational complexity required for drawing such conclusions.

This paper introduces a new semantics for propositional default logic that greatly overcomes these two shortcomings; it is both more intuitive and easier to calculate then previous proposals. In addition, it leads to the identification of new tractable subsets for default logic.

Our approach is based on the concept of meta-interpretations: truth functions that assign truth values to *clauses* rather than logical symbols. Such a truth function is treated as a model for a given default theory if each sentence it satisfies is in some extension of the theory. By studying the properties of these models we were 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 logics reduce to simpler 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 in the literature. Moreover, our method provides, for the first time, a deterministic algorithm for computing extensions of any finite propositional default logic. Other known algorithms [Eth87] are guaranteed to produce an extension only for *ordered* default theories.

In general, the translation we provide is NP-Hard. 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*. Another important subclass of 2-DT is "disjunctions free default theories" in which formulas with disjunction are forbidden. It has been shown [GL90b] 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 extension for 2-DT's are applicable for computing logic programs as well.

Once a default theory is expressed as propositional theory, it invites many heuristics and algorithms that have been studied in the literature on propositional satisfiability. In particular, we show how topological considerations can be used to identify new tractable subsets of default theories and how constraint satisfaction techniques can be effectively applied to tasks of default reasoning.

The paper is organized as follows: we start with preliminary definitions and then introduce the concepts of a meta-interpretation and a model for a default theory. In section 4 we state the features of our compilation, and discuss tractable subsets. Section 5 argues the applicability of our methods to extended logic programs and extended disjunctive data bases and section 6 provides concluding remarks.

2 Definitions and Preliminaries

We consider propositional default theories over a countable propositional language $\mathcal{L}$.

A *default theory* is a pair (D, W) , where D is a set of defaults and W is a set of formulas. 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 in α and I have no reason to believe that one of the β_i is false, then I can believe γ.

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 2.1 *([Rei80],theorem 2.1 reduced to the propositional case) Let $E \subseteq \mathcal{L}$ be a set of formulas, and let (D, W) be a propositional default theory. Define*

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

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

Given a default theory (D, W) and a set of formulas S, we would like to compute answers to the following queries: 1. (Coherence) Does (D, W) have an extension ? If so, find one; 2. (Set-Membership) Is S contained in some extension of (D, W) ? 3. (Set-Entailment)Is S contained in every extension of (D, W) ?

We denote clauses by $c, c_1, c_2, ...$ and the empty clause by Λ. The resolvent of two clauses c_1, c_2 is denoted by $res(c_1, c_2)$. An *interpretation* is a truth assignment for the letters in $\mathcal{L}$. Given a set of formulas S and a formula ω, a *model* for S is an interpretation that satisfies S. $S \vdash \omega$ means that ω is provable from premises S, and $S \models \omega$ means that S entails ω - i.e. that every model of S is a model ω as well. For propositional formulas, $S \vdash \omega$ iff $S \models \omega$. Hence we will use them interchangeably. The *logical closure* of a set of formulas S is the set $\{\omega | S \vdash \omega\}$.

A set that represents the logical closure of a finite theory can be generated using the notion of *prime implicants* [RdK87],

Definition 2.2 *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 propositional theory S, $PI(S)$ denotes its set of prime implicants. A brute force method of computing $PI(S)$ is to repeatedly resolve pairs of clauses of S, add the resolvents to S, and delete subsumed clauses, until a fixed point is reached [1]. There are some improvements to that method (see for example [MR72]), but clearly the general problem is NP-Hard since it solves satisfiability. Nevertheless, for some special cases like 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.

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

3 Propositional Semantics for Default Logic

Our goal is to equip propositional default logic with the semantics of propositional logic. The main advantage of this approach is that it leads to the development of effective methods for computing extensions.

We would have liked an extension for a default logic to be represented as a classical interpretation for propositional logic such that a sentence is satisfied by the interpretation iff it belongs to the extension. However, it is quite obvious that a straightforward representation is not possible: in an interpretation for propositional logic, if a sentence s is not satisfied, then its negation is, while it could certainly be the case that both s and $\neg s$ are not members of a given extension.

Our solution is to use the notion of *meta-interpretation*, where truth values are assigned to sentences rather then letters. Thus, if both s and $\neg s$ are absent from an extension, both s and $\neg s$ will be assigned **false** in the meta-interpretation. Since every sentence can be represented in CNF the atomic syntax structures to which our interpretations assigns truth values will be clauses.

The question is which set of clauses should be represented as atomic symbols in our meta-interpretation. Clearly, it has to be a set of clauses containing all the prime implicants of any extension, In the full paper we show that CLOUSES((D, W)), defined below, is indeed a superset of all the prime implicants of any possible extension of (D, W) .

Definition 3.1 *Let (D, W) be a default theory. The set CLOUSES((D, W)) is the union of C_D, $\rho((D, W)$) $- \{\Lambda\}$ and $PI(W)$, where C_D is the set of the conclusions of all defaults in D, and $\rho((D, W)$) is the resolution closure of C_D and $PI(W)$ with the restriction that no two resolvents are from $PI(W)$. A clause belonging to CLOUSES((D, W))will be called* an atomic clause.□

Definition 3.2 *A* meta-interpretation *for a default theory (D, W) is a classical propositional interpretation for the set of symbols $\mathcal{L}_{(D,W)} = \{c | c \in CLOUSES((D, W)$)$\}$. i.e. - θ is a meta-interpretation for (D, W) iff it is a function from $\mathcal{L}_{(D,W)}$ to* {**true**, **false**}. □

The symbol that represents the clause c in $\mathcal{L}_{(D,W)}$ will be denoted sometimes by I_c [2].

Definition 3.3 *A meta-interpretation θ* satisfies *a clause c ($\theta \not\approx c$) iff either c is a tautology in classical propositional logic or there is an atomic clause $c' \subseteq c$ such that $\theta(I_{c'}) =$* **true**. *A meta-interpretation θ* satisfies *the sentence $c_1 \wedge c_2 \wedge ... \wedge c_n$ ($\theta \not\approx c_1 \wedge c_2 \wedge ... \wedge c_n$) iff forall $1 \leq i \leq n$ $\theta \not\approx c_i$.* □

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

We now want to define when a meta-interpretation for (D, W) is a model for (D, W) , in the sense that each sentence that this model satisfies is in some extension of (D, W) .

[2] We chose this notation to match our intuition that c will be assigned **true** in a meta-interpretation that represents some extension iff it is In that extension.

Definition 3.4 *Let (D, W) be a default theory, and let θ be a meta-interpretation for (D, W) . θ is a* weak model *for (D, W) iff the following conditions hold:*

1. *For each $c \in W$, $\theta \not\approx c$.*
2. *For each default from D, if θ satisfies its preconditions and does not satisfy the negation of each of its justifications, then it satisfies its conclusion.*
3. *For each two atomic clauses c, c' such that $c \subset c'$, if $\theta(c) =$ **true** then $\theta(c') =$ **true**.*
4. *for each two atomic clauses c, c', if $\theta \not\approx c, c'$ then $\theta \not\approx res(c, c')$*
5. *For each atomic clause c such that $\theta | \approx c$ and $c \notin W$ at least one of the following conditions hold:*
   - *There is an atomic clause c_1 such that $c_1 \subset c$ and $\theta \not\approx c_1$.*
   - *There are atomic clauses c_1, c_2 such that $\theta \not\approx c_1, c_2$ and $c = res(c_1, c_2)$.*
   - *There is a default $\alpha : \beta_1, ..., \beta_n / \gamma$ such that $\theta \not\approx \alpha$, for each $1 \leq i \leq n$ $\theta \not\approx \neg\beta_i$ and $\gamma = c$.* □

Definition 3.5 *A* model *for (D, W) is a* minimal *weak model for (D, W) .*

Minimality is defined w.r.t. the following partial order between meta-interpretation: $\theta_1 \leq \theta_2$ iff the set of clauses that θ_1 satisfies is a subset of the set of clauses that θ_2 satisfy. We will say that θ is minimal among a set of meta-interpretations T iff there is no $\theta' \neq \theta$ such that $\theta' \leq \theta$.

Our central claim is that if a meta-interpretation is a model for a default theory Δ, then the set of sentences satisfied by a model for Δ is an extension of Δ, and vice versa. Formally:

Theorem 3.6 *Let (D, W) be a default theory. A set of sentences E is an extension for (D, W) iff there is a model θ for (D, W) such that $E = \{s | \theta \not\approx s\}$.* □

Can the minimality of a model be recognized without having to compare it with all the other weak models ? We will show that a weak model is minimal iff each clause that it satisfies has a proof, where a proof is a sequence of defaults that derive the clause from W. In order to ensure that each proof is well-founded we assign each atomic clause an index which 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. We next show that it is easier to verify well foundedness when the theory is acyclic:

Definition 3.7 *Let (D, W) be a default theory. The* dependency graph *of (D, W) ($G_{(D,W)}$) is defined as follows: For each $c \in CLOUSES((D, W))$ there is a node in the graph. There is an edge from node c to node c' iff at least one of the following conditions hold:*

1. *$c \subset c'$*
2. *There is a clause $c'' \in CLOUSES((D, W))$ such that $c' = res(c, c'')$.*
3. *There is a default $\alpha : \beta_1, ..., \beta_n / c'$ in D and $c \in \alpha$.*

A default theory (D, W) is acyclic *iff $G_{(D,W)}$ is acyclic.* □

The following theorem states why acyclicity is a significant property:

Theorem 3.8 *Every weak model for an acyclic default theory (D, W) is a model for (D, W) .* □

4 Expressing a Default Theory as a Propositional Theory

A model for a default theory (D, W) is actually a classical interpretation over the symbols of $\mathcal{L}_{(D, W)}$. In the full paper we show how we can identify these classical models with a propositional theory which they satisfy (in the classical sense). This is possible because the conditions of definition 3.4 were set in such a way that they can be expressed in propositional logic. This results in an algorithm called *translate* that transforms a a finite default theory (D, W) into a propositional theory $\mathcal{P}_{(D, W)}$ that characterizes its models : every classical model for that propositional theory is a model for (D, W) , and vice versa - every model for (D, W) is a classical model for $\mathcal{P}_{(D, W)}$. Due to space restrictions we omit the algorithm and state only its main features. In the sequel, $\mathcal{P}_{(D, W)}$ stands for the output of the algorithm *translate*.

For notational convenience we use the macro in(). It takes as an input a clause c over $\mathcal{L}$ and returns a sentence in $\mathcal{L}_{(D, W)}$ such that a meta-interpretation M classically satisfies $in(c)$ ($M \models in(c)$) iff M satisfies c ($M \approx c$). $in(c)$ is simply a disjunction of all the symbols in $\mathcal{L}_{(D, W)}$ that represent atomic clauses that are subsets of c.

Theorem 4.1 *Let (D, W) be a finite propositional theory. Suppose $\mathcal{P}_{(D, W)}$ is satisfiable and θ is a model for $\mathcal{P}_{(D, W)}$, and let $E = \{c | \theta(I_c) = \mathbf{true}\}$.*
Then :

1. *E contains all its prime implicants.*
2. *E^* is an extension of (D, W) .* □

Theorem 4.2 *Let E^* be an extension of (D, W) . There is a model θ for $\mathcal{P}_{(D, W)}$ such that for each clause c, $\theta(in(c)) = \mathbf{true}$ iff $c \in E^*$.* □

Consequently, we can first compile a given default theory (D, W) into $\mathcal{P}_{(D, W)}$ and then answer queries as follows: to test if (D, W) has an extension, we test satisfiability of $\mathcal{P}_{(D, W)}$, to see if a set S of clauses is a member in some extension, we test satisfiability of $\mathcal{P}_{(D, W)} + \{in(c) | c \in S\}$ and to see if S is included in the intersection of all the extension, we do not have to compute all the extensions: we simply test if for every $c \in S$, $\mathcal{P}_{(D, W)} + \neg[in(c)]$ is a contradiction.

5 Complexity Considerations

In general, the transformation we proposed is NP-Hard. However, for some special classes it is tractable. For instance, for the class of 2-default theories the transformation can be accomplished in polynomial time and the size of the propositional theory produced is polynomial in the size of the default theory.

Definition 5.1 *A* 2-default theory *(2-DT) is a propositional default theory* (D, W) *where all the sentences in* W *are in 2-CNF, and for each default* $\alpha : \beta_1, ..., \beta_n/\gamma$ *in* D, α *is in 2-CNF, each* β_i *is in 2-DNF and* γ *is a clause of size 2.*

Once we have a tractable transformation, we can apply techniques developed in the *constraints satisfaction* (CSP) literature that further characterize tractable subsets by considering the topological structure of the problem (for a survey, see [Dec91]). For instance, we can characterize the tractability of 2-DT theories as a function of the topology of their *interaction graph.* The interaction graph is an undirected graph, where each clauses in $\mathcal{L}_{(D,W)}$ is associated with a node and, for every $\delta = c_1, ..., c_n : c_{n+1}, ..., c_{n+m}/c_0$ there are arcs connecting $c_0, c_1, ..., c_{n+m}$ in a clique.

The first theorem considers the *induced width* of the interaction graph.

Definition 5.2 *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.*

Theorem 5.3 *For a 2-DT* (D, W) *whose interaction graph has an induced width* w^**, existence, membership and entailment can be decided in* $O(n * 2^{w^*+1})$ *steps when the theory is acyclic and* $O(n^{w^*+2})$ *steps when the theory is cyclic.* □

The second theorem relates the complexity to the size of the cycle cutset. A cycle cutset of a graph is a set of nodes that, once removed, would render the constraint graph cycle-free. For more details about this method, see [Dec90].

Theorem 5.4 *For a 2-DT* (D, W) *whose interaction graph has a cycle cutset of cardinality* c*, existence, membership and entailment can be decided in* $O(n * 2^c)$ *steps when the theory is acyclic and* $O(n^{c+1})$ *steps when the theory is cyclic.* □

6 Application to Disjunctive Databases

We will now demonstrate how our results can be applied to disjunctive databases. In [GL90b], Gelfond and Lifschitz have presented an "answer set" semantics for logic programs with classical negation. They showed that if you identify a rule

$$p_0 \longleftarrow p_1, ..., p_m, \text{not } p_{m+1}, ..., \text{not } p_{m+n}$$

where each p_i is a literal and **not** is the "negation as failure operator",
with the default:

$$p_1, ..., p_m : {\sim}p_{m+1}, ..., {\sim}p_{m+n}/p_0$$

where ${\sim}p$ stands for the literal opposite to p (${\sim}P = \neg P$, ${\sim}\neg P = P$), you get that each disjunction free default theory without contradictory justifications[3] is what they call an extended logic program. They then establish a 1-1 correspondence between the answer sets

[3]They actually claim that each justification should be a literal, but it can be generalized as above

of a program and its extensions. Consequently, our algorithms (see also [BED91a]) can be used for computing answer sets of logic programs and their intersections.

Gelfond and Lifschitz extend the "answer set" semantics so that it can be applied to extended disjunctive databases (EDD) as well [GL90a]. An EDD is a set of rules of the form

$$q_1|...|q_k \longleftarrow p_1, ..., p_m, \text{not } p_{m+1}, ..., \text{not } p_{m+n}$$

We define the *dependency graph* of an EDD to be a directed graph where each literal is a node and there is an edge from p to q iff there is a rule where p appears in the body without the negation as failure operator and q appears in the head. An EDD is acyclic iff its dependency graph is acyclic.

Following [GPLT91] (section 6), for each EDD D, D' will denote the EDD obtained by replacing each rule of the above form with k normal rules

$$q_1 \longleftarrow p_1, ..., p_m, \text{not } p_{m+1}, ..., \text{not } p_{m+n}, \text{not } q_2, ..., \text{not } q_k$$
$$...$$
$$q_k \longleftarrow p_1, ..., p_m, \text{not } p_{m+1}, ..., \text{not } p_{m+n}, \text{not } q_1, ..., \text{not } q_{k-1}$$

Note that D' is an extended logic program (with no disjunctions).

Our next theorem follows in part from Theorems 6.1 and 7.2 in [GPLT91]. It identifies a large class of EDDs that have an equivalent extended (nondisjunctive) logic program which is not much larger.

Theorem 6.1 *If an EDD D is acyclic then S is an answer set for D iff S is an answer set for D'.*

Thus, if we have an acyclic EDD we can first transform it into an extended logic program and then compute its answer set using the methods presented in [BED91a] and here.

7 Conclusions

This paper generalizes the results presented at [BED91a] and [BED91b] and provides propositional semantics to default theories with disjunctions. This new semantics leads to effective algorithms for computing extensions and membership in intersections of extensions of any finite propositional default theories. It also leads to the discovery of new tractable subsets for default logic. Related results for autoepistemic logic were reported in [MT91] where it was shown that the question of membership in every expansion of an autoepistemic theory can be reduced to propositional provability.

We have also discussed the applicability of our results to Extended Disjunctive Databases (EDD's). As pointed out in [GL90a], the embedding of extended programs into Reiter's default logic cannot be generalized to disjunctive databases. We showed however, that when the EDD is acyclic, we can find an extended logic program (or equivalently, a disjunction free default theory) that has the same answer sets (extensions). In contrast, disjunctive default theories cannot be reduced to disjunction free default theories even if they are acyclic: for example, we can not find an equivalent disjunction free theory for the acyclic theory $D = \emptyset$, $W = \{p \vee q\}$. Thus, acyclic disjunctive default theories are more expressive then acyclic disjunctive databases.

References

[BED91a] Rachel Ben-Eliyahu and Rina Dechter. Default logic, propositional logic and constraints. In *The national conference on AI*, pages 379–385, Anaheim,CA,USA, July 1991.

[BED91b] Rachel Ben-Eliyahu and Rina Dechter. Inference in inheritance networks using propositional logic and constraints networks techniques. Technical Report R-163, Cognitive systems lab, UCLA, 1991. Presented at the Bar-Ilan synposium on the foundations of AI, June 1991, Ramat-Gan, Israel.

[Dec90] Rina Dechter. Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition. *Artificial Intelligence*, 41:273–312, 1990.

[Dec91] Rina Dechter. Constraints networks. To appear in the 2nd edition of the encyclopedia of AI, 1991.

[Eth87] David W. Etherington. Formalizing nonmonotonic reasoning systems. *Artificial Intelligence*, 31:41–85, 1987.

[GL90a] Michael Gelfond and Vladimir Lifschitz. Classical negation in logic programs and disjunctive databases. Submitted for publication, 1990.

[GL90b] Michael Gelfond and Vladimir Lifschitz. Logic programs with classical negation. In *The 7th international conference on Logic Programming*, pages 579–597, Jerusalem,Israel, June 1990.

[GPLT91] Michael Gelfond, Halina Przymusinska, Vladimir Lifschitz, and Miroslaw Truszczynski. Disjunctive defaults. In *The second international conference on principles of knowledge representation and reasoning*, Cambridge,MA, 1991.

[MR72] Eliana Minicozzi and Ray Reiter. A note on linear resolution strategies in consequence-finding. *Artificial Intelligence*, 3:175–180, 1972.

[MT91] Wiktor Marek and Miroslaw Truszczynski. Computing intersection of autoepistemic expansions. In *1st International workshop on Logic Programming and Nonmonotonic Reasoning*, pages 37–50, Washington DC, July 1991.

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