Decision procedure for Default Logic

W. Marek[1] and A. Nerode[2]

Abstract

Using a proof-theoretic approach to non-monotone reasoning we introduce an algorithm to compute all extensions of any (propositional) default theory (D, W) and we also prove correctness of the method. This algorithm is based on a modified analytic tableaux method.

1 Preliminaries

In this paper we give a neat tableaux method of computing extensions for default theories expressed in propositional calculus with some nonmonotone default rules. In a sequel ([Marek, Nerode and Remmel, 1990] we will give a general theory of such nonmonotone formal systems, of which the present paper is the simplest case. This later theory will include nonmonotonic propositional and predicate logic, classical or intuitionistic, and also modal logic, propositional and predicate. First, we explain the default propositional logic case in general terms which will be used in the sequel.

[1]Department of Computer Science, University Kentucky, Lexington, KY 40506–0027. Currently in Mathematical Sciences Institute at Cornell University. Work partially supported by N.S.F. grant RII 8610671 and Kentucky EPSCoR program and the ARO contract DAAL03-89-K-0124

[2]Mathematical Sciences Institute, Cornell University, Ithaca, NY 14853, Work partially supported by NSF grant MCS-83-01850 and ARO contract DAAG629-85-C-0018.

In a (monotone) classical or intuitionistic propositional and predicate logic there is a simple notion of "semantic deductive closure $D(I)$" of a set I of premises, namely $D(I)$ is the collection of all statements true in all models in which I is true. Then a completeness proof shows that this notion has a syntactical characterization, namely $D(I)$ is generated by closing I under a set of "monotonic rules of inference" of the form:

$$\text{"if } \alpha_1, \ldots, \alpha_n \text{ are in } D(I), \text{ then } \gamma \text{ is in } D(I)\text{"}.$$

such a $D(I)$ can be interpreted as a unique total set of beliefs you should hold, grounded in knowing I. In nonmonotonic logics such as default logic of Reiter, the situation is more complicated. For a given consistent set I there may be no or competing, sometimes even mutually inconsistent, candidates for the total set of beliefs that you should hold given that you know I. These possible "total systems of beliefs grounded in knowing I" are called *extensions* S of I. Formally, an extension S of I is set of statements containing I with properties (i), (ii) below.

(i) S is closed under a given set N of (possibly) nonmonotone rules of form

$$\text{If } \alpha_1, \ldots, \alpha_n \text{are in } S \text{ and } \beta_1, \ldots, \beta_k \text{are not in } S \text{ then } \gamma \text{ is in } S. \tag{1}$$

(ii) S is the deductive closure, in the usual monotone sense, under those monotone rules:

$$\alpha_1, \ldots, \alpha_n \text{ are in } S \text{ implies } \gamma \text{ is in } S \tag{2}$$

arising from those nonmonotone rules of form (1) in N with $\beta_1, \ldots, \beta_k$ not in S.

This suggests adding apparatus to the usual monotone apparatus appropriate for (2) to allow the effect of (1). This amounts to building a logical calculus to deal with extensions. In this paper we build tableaux in which putative extensions S are made into first class objects (as opposed to metamathematical objects) to reveal what extensions S are possible, and to carry out tableaux proofs with them. This amounts to allowing statements of the form $T(\varphi \in S)$, and $F(\varphi \in S)$ into a Smullyan type tableaux for determination of extensions. We give a formal description of default logic, in more details, below.

Default logic, introduced by Reiter [1980] and subsequently investigated by various authors ([Reiter nad Criscuolo, 1983], [Etherington, 1987], [Lukaszewicz, 1985], [Konolige, 1988]) is one of most important forms of non-monotonic reasoning. Default logic attempts to capture not only the way correct reasoning is performed but also how (and in what circumstances) we are able to "jump to conclusions" in a logical way. This method of reasoning operates as follows: We have a number of "hard facts" (formulas of the underlying language $\mathcal{L}$) and a collection of "default rules". These rules informally look like this:

"rule d: If φ has been established then, as long as the situation is normal, derive ϑ". Here "normality" of the situation is not a statistical mode of reasoning but logical, "normality for rule d" means that no statement witnessing to abnormality has been established. In particular, if the agent's knowledge is T, then the normality with respect to rule d means that some formulas $\psi_1, ..., \psi_n$ are not in T. To formalize this concept completely, we see that we deal with the rules of form: $\frac{\varphi : \psi_1, ..., \psi_n}{\vartheta}$ and the

meaning of such rule is: "If φ has been established and $\neg\psi_1, \ldots, \neg\psi_n \notin T$ then derive ϑ". (This is slightly different from our presentation above, since we talk about negation of formulas. In general nonmonotone systems we do not have to have negation at all, and so there we follow the general scheme outlined above. For uniformity with other papers in the field we follow Reiter's notation in this paper).

This approach lead Reiter [1980] to formulate the notion of extension as a fixed point of the following operator Γ. Let (D, W) be a default theory, where $W \subseteq \mathcal{L}$, and D is a collection of default rules. The operator $\Gamma = \Gamma_{D,W}$ is defined as follows: If $S \subseteq \mathcal{L}$, then $\Gamma(S)$ is the least collection of formulas U with these properties:

1) $W \subseteq U$.

2) U is closed under propositional consequence.

3) Whenever $d \in D$, $d = \frac{\varphi:\psi_1,\ldots,\psi_n}{\vartheta}$ and $\varphi \in U$ and $\neg\psi_1 \notin S$, ..., $\neg\psi_n \notin S$, then $\vartheta \in T$.

An extension is then a fixed-point of the operator Γ, i.e. T is an extension of (D, W) if and only if $T = \Gamma(T)$.

It is possible to construct all the extensions of a finite default theory (D, W) using a roundabout route: Firstly, we could translate the default theory into modal logic using the Konolige's [1988] translation. Next, we could find, using the algorithm of Niemelä [1988], all the expansions of the translation. Then, we could test each of these extensions for the property of "strong groundedness" (for details check [Konolige, 1988], including his correction [Konolige, 1989]). The "objective parts" of the strongly grounded expansions will then be the extensions. Etherington [1987] refers to his algorithm from [Etherington, 1982] for finding extensions without a correctness

proof. An alternative approach is used in [Zhang and Marek, 1989]. Again there the algorithm does not use standard techniques of logic.

We shall give an algorithm for finding all expansions of a (propositional) default theory. We shall prove that our algorithm is correct. The algorithm works for infinite default theories as well- in the sense that the branches of the tree we construct, if meeting the test, are extensions. In case of infinite theories the tree, however, is denumerable, and potentially has $2^{\aleph_0}$ branches. The recursion-theoretic and topological properties of this collection of branches (under appropriate coding) are discussed in [Marek, Nerode and Remmel, 1990].

The correctness proof for the algorithm is based on a novel characterization of default extensions as closures of the initial knowledge W under monotonic projections of non-monotonic rules from D.

The method we use is a variant of analytic tableaux method (see [Smullyan, 1968]). On the other hand, following [Nerode, 1989] we put on a tableau more than just subformulas of a formula to be analyzed. In addition we shall use a number of auxiliary tableaux to decide which signed formulas will be put on a tableau.

A tableau is a tree containing signed formulas. A signed formulas we consider here are of three kinds: Firstly there are formulas of form $T(\varphi)$ or $F(\varphi)$, Then, we have formulas of form $T(\varphi \in S)$ and $F(\varphi \in S)$ (S is a new symbol here). Finally we have formulas of form ["d is used"].

We shall subsequently apply rules to close branches in the trees we shall construct. Those rules involve constructing auxiliary tableaux (in principle we can execute them on the same tableau using additional markings) and comparing the content of branches.

In the next section we shall use two types of tableaux. First, we shall use ordinary tableaux with the usual development rules. These development rules involve extending and splitting branches of the tree. There are two types of extension rules:

Unary, (extensions without splitting).

(a) If $T(\neg\varphi)$ is on the branch, extend it by adding $F(\varphi)$.

(b) If $F(\neg\varphi)$ is on the branch, extend it by adding $T(\varphi)$.

(c) If $F(\varphi \vee \psi)$ is on the branch, extend it by adding $F(\varphi)$ and $F(\psi)$.

(d) If $T(\varphi \wedge \psi)$ is on the branch, extend it by adding $T(\varphi)$ and $T(\psi)$.

(e) If $F(\varphi \Rightarrow \psi)$ is on the branch, extend it by adding $T(\varphi)$ and $F(\psi)$.

Binary, (extension with splitting into two branches).

(f) If $F(\varphi \wedge \psi)$ is on the branch, split and extend one branch by adding $F(\varphi)$, the other branch by adding $F(\psi)$.

(g) If $T(\varphi \vee \psi)$ is on the branch, split and extend one branch by adding $T(\varphi)$, the other branch by adding $T(\psi)$.

(h) If $T(\varphi \Rightarrow \psi)$ is on the branch, split and extend one branch by adding $F(\varphi)$, the other branch by adding $T(\psi)$.

Secondly, we shall consider another tableau, called main tableau. On this tree we put three types of markings:

a) $T(\varphi \in S)$ (S is a new, auxiliary symbol),

b) $F(\varphi \in S)$

c) ["d is used"]

The specific rules for development the main tableau are given in the next section.

2 Decision method

Given a default theory (D, W), with D and W finite, we shall construct a tree (main tableau) whose open branches describe and construct all extensions of (D, W). In the process of building the tree we shall build a number of auxiliary tableaux. The auxiliary tableaux will be used to decide how the main tableau is expanded, and which branches are closed.

Initially we put on the tableau all the formulas $T(\varphi \in S)$ for all the formulas $\varphi \in W$.

We shall describe now expansion procedure for the main tableau $T_{D,W}$. We need to define how a branch is expanded. Assume that b is a branch of our main tableau $T_{D,W}$. Form two auxiliary sets of formulas, S_b and nS_b by defining: $S_b = \{\varphi: T(\varphi \in S) \in b\}$, $nS_b = \{\varphi: F(\varphi \in S) \in b\}$. We say that the default d is *applicable* for branch b if the statement ["d is used"] is not on the branch b. In order to extend a branch we consider only applicable defaults. These split in three categories: of *active*, *unusable* and *sleeping* defaults.

Test procedure for deciding the type of a default:

If $d = \frac{\chi:\psi_1,\ldots,\psi_r}{\vartheta}$ is a default, then d is active if $\chi \in Cn(S_b)$, and for all $i \leq r$, $\neg\psi_i \notin Cn(S_b)$. Testing for activity requires development of $r+1$ ordinary tableaux. If the test for activity fails because the first tableau is open (and it should be closed, of course) then the default d is called sleeping. If one of the other tableaux is closed, it is called unusable. It should be clear that once a default becomes unusable, it will stay this way at any extension of the branch.

Test now all applicable defaults for their activity. Take the first active default on the branch- if there is one. Extend the branch b in two branches as follows:

Branch to the left extends the branch b with several new formulas: First, we put ["d is used"]. Second, we put the following formulas on the branch: $F(\neg\psi_1 \in S)$, ..., $F(\neg\psi_r \in S)$, $T(\vartheta \in S)$.

Branch to the right extends the branch b by having there only the formula ["d is used"].

If there is no active default (i.e. every default in D is not applicable for b or is unusable or is sleeping), then the branch b is completed and will not be extended.

In this way the main tableau has been completely described. It should be clear that this is a finite tree. We shall indicate now how we close branches.

The procedure to close branches proceeds in two phases. First we shall close branches corresponding to theories that are too big to be an extension. Second, we shall eliminate theories that are too small to be an extension.

Phase I of closing branches of the main tableau. Let b be a branch. For each formula $\psi \in nS_b$ test (using an ordinary tableau) if $\psi \in Cn(S_b)$. If so, close the branch. Otherwise pass to the phase II.

Phase II of closing branches of the main tableau. Consider only branches not yet closed (in phase I). For each such branch b form an auxiliary family of rules D_b as follows:

(1) If a default $d = \frac{\varphi:\psi_1,...,\psi_r}{\vartheta}$ is a default, has the property that for some $i \leq r$, $\neg\psi_i \in Cn(S_b)$, then eliminate the default d altogether.

(2) Otherwise, (monotone) rule $\frac{\varphi}{\vartheta}$ belongs to D_b.

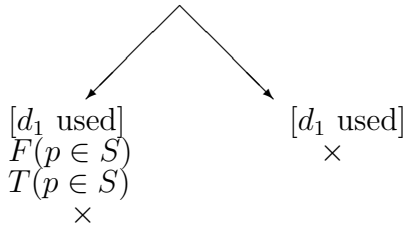
(Readers familiar with the stable semantics for logic programs ([Gelfond and Lifschitz, 1988]) recognize the similarity).

For every rule $\frac{\chi}{\vartheta}$ in the auxiliary family D_b, test the following, using ordinary tableaux: Either $\chi \notin Cn(S_b)$ (that is a tableau built by putting $T(\varphi)$ for $\varphi \in S_b$ and $F(\chi)$ stays open) or else both χ, ϑ belong to $Cn(S_b)$ (that is two tableaux built as above are closed). If for for every auxliary rule in D_b this test is passed, the branch b stays open, otherwise it is closed.

The output of our algorithm is the collection of S_b for branches b **open** after the second phase of our algorithm if that collection is nonempty, or a statement that $< W, D >$ has no extensions, otherwise.

Before we prove the correctness of our algorithm, we shall consider two examples.

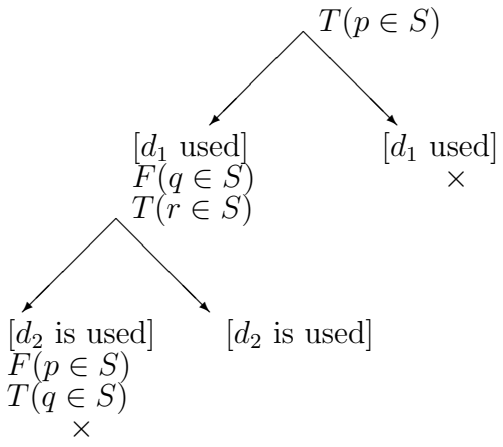
Example 2.1 *Let* $W = \emptyset$, $D = \{\frac{:\neg p}{p}\}$



picture 1.

The left branch is closed in the phase I, the right branch during the phase II. Thus the tableau $T_{D,W}$ has no open branches and we conclude that (D, W) has no extension.

Example 2.2 *Let* $W = \{p\}$, $D = \{\frac{p:\neg q}{r}, \frac{r:\neg p}{q}, \frac{q:s}{t}\}$



picture 2.

The leftmost branch b_1 is closed in the phase I. The rightmost branch b_3 is also closed, in phase II. Indeed, the collection D_{b_3} consists of (monotonic) rules $\frac{p}{r}$, and

$\frac{q}{t}$. The rule $\frac{p}{r}$ has the property that p is provable from S_{b_3}, but r is not. Hence the branch b_3 is closed. It is easily checked that the remaining branch b_2 is not closed, hence it generates an extension, that is, $Cn(p, r)$ is an extension of the theory under consideration.

Proposition 2.1 *If $T = S_b$ and b is a open branch of $T_{D,W}$, then T is an extension of (D, W).*

Proposition 2.2 *Every extension of (D, W) is of form $Cn(S_b)$ for some open branch b of $T_{D,W}$.*

Corollary 2.3 *Our algorithm correctly computes (bases for) all extensions of finite (D, W).*

Let $(D, W) \mid\sim_s \varphi$ denotes that φ belongs to some extension of (D, W), and $(D, W) \mid\sim_a \varphi$ denotes that φ belongs to all extensions of (D, W). Then we have:

Corollary 2.4 *Relations $(D, W) \mid\sim_s \varphi$ and $(D, W) \mid\sim_a \varphi$ are computable over finite (D, W).*

The proof of proposition 2.1 is based on a characterization theorem for extensions in terms of projections of families of default rules. First some definitions.

Let D be a collection of default rules, and let $S \subseteq \mathcal{L}$. The projection D_S of the collection D by S is the collection $\{d_S : d \in D\}$, where d_S is defined as follows:
Let $d = \frac{\varphi : \psi_1, \ldots, \psi_r}{\vartheta}$ then:

(1) If for some $i \leq r$, $\neg\psi_i \in S$ then $d_S = <>$

(2) If for every $i \leq r$ $\neg\psi_i \notin S$ then $d_S = \frac{\varphi}{\vartheta}$.

Condition (1) amounts to elimination of rule d altogether. D_S is a family of monotone rules. The collection D_b above is nothing else but D_{S_b}. With this family of rules we associate the (monotone) consequence operator Cn_S by taking as closure conditions provability with usual rules of propositional calculus and additionally the rules $\{d_S : d \in D\}$.

Proposition 2.5 *$Cn_S(T)$ is the least collection of formulas containing T, closed under Cn and the rules in D_S.*

Given a monotonic rule $\frac{\chi}{\vartheta}$, let us introduce two auxiliary tableau rules:

(i) If $F(\vartheta)$ is on a branch, extend it by $F(\chi)$.

(j) If $T(\chi)$ is on a branch, extend it by $T(\vartheta)$.

(that is for every rule in D_S such pair is introduced).

Proposition 2.6 *The tableau procedure based on the usual rules for analytic tableaux and the two rules (i) and (j) above is complete for Cn_S.*

Next, we have the following characterization of extensions:

Theorem 2.7 *Let (D, W) be a default theory, and let $S \subseteq \mathcal{L}$. Then S is an extension of (D, W) if and only if $S = Cn_S(W)$.*

Proof: We need to prove that our condition $S = Cn_S(W)$ is equivalent to the fact that $S = \Gamma(S)$. This, of course, follows from the folowing fact that we prove

below: $\Gamma(S) = Cn_S(S)$. (Bear in mind that $\Gamma(\cdot)$ is a non-monotone operator, $Cn_S(\cdot)$ is monotone, but it carries a parameter (namely S) and in this parameter it is not monotone). Recall that $\Gamma(S)$ is the least set of formulas U such that:

(1) $W \subseteq U$,

(2) $Cn(U) = U$,

(3) If $d = \frac{\varphi:\psi_1,\ldots,\psi_r}{\vartheta} \in D$ $\neg\psi_j \notin S$ for all $j \leq r$, and $\varphi \in S$ then $\vartheta \in U$.

We first check that $Cn_S(W)$ possesses properties (1)-(3). This follows from the proposition 2.5. Then we analyse the condition (3) and we find that it means that U is closed under rules d_S. This, using again the proposition 2.5, means that $Cn_S(W) = \Gamma(S)$. □.

A similar characterization (in terms of context-dependent proofs) was given in [Marek and Truszczyński, 1989].

Now, we are in a position to prove proposition 2.1.

Proof of proposition 2.1: Let b be a branch of the main tableau that have not been closed after the two phases of closing. The second part of closing procedure tests whether the collection S_b has the property that $Cn(S_b)$ is identical with $Cn_{S_b}(W)$. Thus, we test there (according to the theorem 2.7) if $Cn(S_b)$ is an extension. □

There is an important difference between the phases I and II of closing branches. The phase I is "local", and the check for superfluous formulas can be made immediately after putting formulas of form $T(\varphi \in S)$ or $F(\varphi \in S)$ on the branch. The phase II has to be made, however, after the whole main tree is completed.

We know now that open branches of the main tableau produce extensions. Next we see that in fact all the extensions will appear as the open branches.

Proof of proposition 2.2. Let $D' = GD((D, W), T)$ be the collection of generating defaults for T (c.f. [Reiter, 1980]). Consider now the ordering of D' according to the ordering of D. Proceed as follows:

If a default $d \notin D'$ becomes the first active, then go to the right that is put only ["d is used"] on the extended branch.

If $d \in D'$ and is first active then go to the left that is put all the possible formulas: ["d is used"], $T(\vartheta \in S)$ and all the formulas $F(\neg\psi_j \in S)$ $(j \leq r)$ on the extended branch.

This construction assures that if a formula of form $T(\varphi \in S)$ is on the constructed branch then either φ belongs to W or φ is the conclusion of a generating default. On the other hand all the generating defaults of an extension eventually become active (This needs a proof, which follows easily by induction). Thus for every generating default d, with conclusion ϑ, the formula $T(\vartheta \in S)$ is put on b.

All the formulas ψ such that $F(\psi \in S)$ appears on b come from generating defaults. Thus $\psi \notin T$. Consequently the phase I test is met, and the branch b stays open after the first phase. We need to show that the same happens after the phase II. Look at the collection D_b. If a rule $\frac{\chi}{\vartheta}$ is there then either $\chi \in Cn(S_b) = T$ and $\frac{\chi}{\vartheta}$ comes from a generating default, and whatever is proved using it is already in T, i.e. $Cn(S_b)$, or else $\chi \notin T$. But this means precisely that the phase II does not close the branch. This completes the proof. □

The propositions 2.1 and 2.2 tell us that as long as we are interested in testing

whether the theory (D, W) possesses at least one expansion, we do not need to construct all the main tableau. In fact, if we want to find just one extension, then we can construct the the main tableau using the depth-first search, until we find the first open branch. That is we can construct the leftmost branch, if this one happens to be closed, then the next one etc. This construction involves construction and traversal of the main tableau in a fixed order. If no extension exists, the whole tableau will be constructed and all the branches will be closed. Then we conclude, by proposition 2.2, that (D, W) has no extension.

The complexity of our algorithm is high. The reason for it is that many times during the construction of the main tree we check if some formula possesses proof. This could be done by a resolution theorem prover, rather than by tableaux. The complexity result should not be surprising, as in [Marek and Truszczyński, 1988] it is shown that for the simplest possible nonmonotonic default theories (the prerequisite part empty, one negated atom in justification, an atom as a conclusion) the problem is already NP-complete and reducible in a most simple way to the problem of existence of a kernel in graph.

References

[Etherington, 1982] D.W. Etherington. M.Sc. Thesis, Vancouver, B.C.

[Etherington, 1987] D.W. Etherington. Formalizing Nonmonotonic Reasoning Systems. *Artificial Intelligence Journal* 31:41–85, 1987.

[Gelfond and Lifschitz, 1988] M. Gelfond, V. Lifschitz. The Stable Semantics for Logic Programming. In: *Proceedings of 5th International Symposium/Conference on Logic Programming*, Seattle, 1988, MIT Press, pp. 1070-1080.

[Konolige, 1988] K. Konolige. On the Relation between Default and Autoepistemic Logic. *Artificial Intelligence* 35:343–382, 1988.

[Konolige, 1989] K. Konolige. Correction to [Konolige, 1988]

[Lukaszewicz, 1985] W. Lukaszewicz. Two results on default logic. In: *Proceedings of the Int. Joint Conf. on Artificial Intelligence*, IJCAI-85, pp. 459-461.

[Marek and Truszczyński, 1988] W. Marek and M. Truszczyński. Autoepistemic Logic, *Journal of the A.C.M.*, to appear.

[Marek and Truszczyński, 1989] W. Marek and M. Truszczyński. Relating Autoepistemic and Default Logics, In: *Principles of Knowledge Representation and Reasoning*, pages 276 – 288, Morgan Kaufman, San Mateo, 1989.

[Nerode, 1989] A. Nerode. Some lectures on modal logic, A manuscript.

[Marek, Nerode and Remmel, 1990] W. Marek, A. Nerode and J.Remmel. A theory of nonmonotone rule systems I, Technical Report, Mathematical Sciences Institute, (Abriged version to appear in: Proceedings of Symposium on Logic in Computer Science.)

[Niemelä, 1988] I. Niemelä. Decision Procedure for Autoepistemic Logic., *Lecture Notes in Computer Science* 310:675-684, Springer Verlag, 1988.

[Reiter, 1980] R. Reiter. A Logic for Default Reasoning, *Artificial Intelligence*, 13:81–132, 1980.

[Reiter, 1987] R. Reiter. Nonmonotonic Reasoning, *Ann. Rev. Comput. Sci.*, 2:147–186, 1987.

[Reiter nad Criscuolo, 1983] R. Reiter and R. Criscuolo. On interacting defaults, In: *Proceedings of the Int. Joint Conf. on Artificial Intelligence*, IJCAI-81, pp. 270-276.

[Smullyan, 1968] R.M. Smullyan. *First-Order Logic*, Springer Verlag, 1968.

[Zhang and Marek, 1989] A. Zhang and W. Marek. On the classification and existence of extensions in default logic, To appear in *Fundamenta Informaticae*.