ICOT Technical Memorandum: TM-0788

TM-0788

A Note on Proof Procedures for Nonmonotonic Reasoning Systems

by
Nicolas Helft & K. Inoue

August, 1989

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03) 456-3191~5
Telex ICOT J32964

Institute for New Generation Computer Technology

A Note on Proof Procedures for Nonmonotonic Reasoning Systems *

Nicolas Helft helft@icot.jp

Katsumi Inoue inoue@icot.jp

ICOT Research Center
1-4-28, Mita, Minato-ku, Tokyo 108, Japan

May 10, 1989

1 Introduction

This paper is concerned with proof procedures for nonmonotonic reasoning systems. Although we believe many remarks and conclusions presented apply to most of such systems, we concentrate on predicate logic, and its largest computable subset, clausal logic with domain closure and uniqueness of names assumptions.

Very recently, algorithms for such nonmonotonic systems have appeared, and it seems that many of them are addressing exactly the same problem. These include Przymusinski's [4] and Ginsberg's [2] algorithms to compute circumscription, and Reiter and de Kleer's CMS [5], a generalization of de Kleer's ATMS, which we analyze in this paper.

Independently of these, Siegel [7] has defined a framework that clearly explains the computational aspect of such systems, and produced an efficient algorithm which can be applied to such problems.

This paper

- Explains Siegel's framework;
- Uses it to show the strong connection between the problems addressed in [4, 2, 5] as well as others involving nonmonotonic reasoning;
- Makes a detailed comparison between Siegel's and Przymusinski's algorithms, which appear to be very similar. Those conclusions are that 1) Siegel's is more general, and thus can be applied to other problems, and 2) it is more efficient as it includes restrictions on the search space not present in [4];

*This is a first draft.

- Shows how the computational aspect of the CMS, and in particular the ATMS can be solved with this algorithm;
- Makes a comparison between Ginsberg's algorithm and Przymusinski's, which becomes very easy, as Ginsberg uses an ATMS as part of his algorithm to compute circumscription.

We assume knowledge of [4, 2, 5], although we provide short summaries of the results presented in those papers.

2 Siegel's Framework

Siegel first defines two problems that help to understand the computational aspect of several nonmonotonic reasoning systems.

We fix a set of clauses called a *production field*, that will be denoted $\mathcal{P}$. These are intended to represent the "interesting" clauses to solve a certain problem, and will generally be infinite, and therefore defined implicitly: e.g., the set of positive clauses.

Problem 1: Given a theory T and $\mathcal{P}$, find the theorems of T that belong to $\mathcal{P}$, i.e. $Th(T) \cap \mathcal{P}$.

Problem 2: Given a theory T, $\mathcal{P}$ and a formula F, find the theorems of $T + F$ that belong to $\mathcal{P}$ and are not theorems of T, i.e. $(Th(T + F) - Th(T)) \cap \mathcal{P}$.

The only requirement on $\mathcal{P}$ is that it should have the following property:

Definition: A production field $\mathcal{P}$ is *stable* if for every formula F that belong to $\mathcal{P}$, $F' \models F$ implies that F' belongs to $\mathcal{P}$.

An easy property is then proved: the union or the intersection of stable production fields is a stable production field.

For example the set of positive clauses, and the set of clauses formed only from literals belonging to a certain vocabulary are stable production fields. From the result above we can conclude that the positive clauses belonging to a certain vocabulary are also stable.

As we will use these concepts subsequently, we give them a name; the following definition generalizes one given in [1].

Definition: We assume a production field $\mathcal{P}$ is given. T is a set of formulae, and F is a formula.

1. The *characteristic clauses* of T, denoted $Carc(T)$, are the theorems of T that belong to $\mathcal{P}$; $Carc(T) = Th(T) \cap \mathcal{P}$. Moreover, we require that this set contains no subsumed clauses, i.e. if C and C' are in the set and $C \models C'$, then $C' \models C$.
2. The *new characteristic clauses* of F wrt T are the characteristic clauses of $T + F$ which are not characteristic clauses of T; $Newcarc(T, F) = Carc(T + F) - Carc(T)$.

We will see how these concepts help to understand much of the work being done on the computational aspects of nonmonotonic reasoning.

3 Przymusinski's Theorems

Przymusinski [4] is concerned with the problem of computing circumscription. For this he provides both a pair of theorems and an algorithm. In this section, we will reexpress the theorems using the definitions given above. This will help to understand the meaning of the theorems and make clear the relationship with other work. We first consider the two theorems explicitly mentioned in the paper, and then show that these have been strengthened in the proofs of the completeness of the algorithm. We will be later concerned with the algorithm itself.

3.1 First Results: Theorems 2.5 and 2.6

Theorem 2.5 (Przymusinski): If F does not contain literals from Z, then $CIRC(T;P;Z) \models F$ iff there is no clause D such that (i) D does not contain literals from $Z + P^-$, and (ii) $T \models \neg F \vee D$ but $T \not\models D$.

The production field here is $\mathcal{P} = P^+ + Q$. [1] Condition (i) thus means D belongs to $\mathcal{P}$. $T \models \neg F \vee D$ can be written as $T + F \models D$. So we are looking for a clause D in the production field, implied by $T + F$ but not by T alone. In other words,

Theorem 2.5 (New version): Let F be a formula not containing literals from Z. Let $\mathcal{P}$ be $P^+ + Q$. Then

$$CIRC(T;P;Z) \models F \quad \text{iff} \quad Newcarc(T,F) = \phi.$$

For formulae containing predicates from Z, the following holds:

Theorem 2.6 (Przymusinski): F is any formula. $CIRC(T;P;Z) \models F$ iff either $T \models F$ or there is a formula G such that (i) G does not contain literals from $Z + P^-$; (ii) $T \models F \vee G$ and (iii) $CIRC(T;P;Z) \models \neg G$.

Now, $T \models F$ means $T \cup \neg F \models false$; note that in that case, $Newcarc(T, \neg F)$ would contain only $\square$ (the empty clause). Condition (i) again means that G belongs to $\mathcal{P}$; and condition (ii) can be written as $T \cup \neg F \models G$. If G is $\square$ (the formula $false$), then $\neg G$ is the formula $true$ and adding it to a set of formulae produces no new theorem: $Newcarc(T, true) = \phi$. We can now write

[1] We denote a production field simply as a union of collections of literals, but it also means the set of clauses composed only by those literals.

Theorem 2.6 (New version): Let F be any formula. Then

$CIRC(T;P;Z) \models F$ iff $Newcarc(T, \neg F)$ is a formula G [2] such that $Newcarc(T, \neg G) = \phi$.

Of course no important result has been given yet. But this formulation seems to be much more clear than the former. For example, the first theorem says the following. We want to know if a formula is true in the (P, Z)-minimal models of a theory. Now, certain clauses deducible from the theory are representative of those minimal models, in the sense that if adding the query to the theory produces a change in those clauses (i.e. a new one) then the addition of the query has produced a change in the minimal models as well. These representative clauses are of course the characteristic clauses. The existence of a new characteristic clause thus means the query has altered the minimal models: thus if *Newcarc* is empty, the addition of the query has no effect on the minimal models and thus the circumscription of the theory satisfies the query.

Let's review some of Przymusinski's examples with this new concepts. (The examples are numbered as in [4]).

Example 3.5: The following theory T is given, with $P = \{l, s, j\}$, and $Z = \phi$.

$l(A, L) \vee l(A, G) \vee \neg s(A)$

$\neg ms(A) \vee l(A, L)$

$s(A)$

$ms(A)$.

The production field is then $\mathcal{P} = P^+ + Q$, i.e., positive occurrences of literals whose predicate symbol is l, s, or j, and any occurrences of literals from ms.

The characteristic clauses of T are thus:

$l(A, L)$

$l(A, F) \vee l(A, S) \vee \neg j(A)$

$s(A)$

$ms(A)$.

The query $C = \neg l(A, G)$ has the same characteristic clauses: if added to T it generates no new theorem belonging to $\mathcal{P}$. Thus $Newcarc(T, C) = \phi$ and, as expected, $CIRC(T, P) \models C$.

Next an example involving queries containing predicates from Z.

[2] We use *Newcarc* as a single formula as well as a set of clauses (with an implicit conjunction).

Example 3.10: The theory T is

$$B(T)$$
$$\neg B(T) \vee Ab(T) \vee F(T)$$
$$\neg O(T) \vee \neg F(T).$$

In this classical bird example, of course $P = \{Ab\}$ and $Z = \{F\}$, so $\mathcal{P}$ contains positive occurrences of Ab, or any occurrence of B and O. The characteristic clauses are then:

$$B(T)$$
$$Ab(T) \vee \neg O(T).$$

Przymusinski shows that the circumscription of T does not satisfy $G = F(T)$ but satisfies $H = O(T) \vee F(T)$. Let's verify these facts.

Adding $\neg F(T)$ to T produces the new characteristic clause $Ab(T)$. We now add $\neg Ab(T)$ to T which gives another new characteristic clause: $\neg O(T)$, and thus $F(T)$ does not follow from the circumscribed theory.

Now we add $\neg H = \neg O(T) \wedge \neg F(T)$. $T + \{\neg O(T)\} + \{\neg F(T)\}$ has $\neg O(T)$ and $Ab(T)$ as new characteristic clauses. The negation of these two clauses is $O(T) \vee \neg Ab(T)$. Adding this to T produces no new characteristic clauses, as the only new theorems are $O(T) \vee \neg Ab(T)$ and $F(T)$ and they do not belong to $\mathcal{P}$. Thus, as expected, G is in the circumscribed theory.

3.2 Second Results: Completeness of MILO-resolution

Przymusinski then presents an algorithm, MILO-resolution, and proves its completeness regarding the problem of computing circumscription. In fact *this completeness result strengthens both of the above theorems*, showing that not all the characteristic clauses need to be produced. It is sufficient to consider a subset of these, those that result from the algorithm skipping all literals in the production field.

For a trivial example, if $\mathcal{P}$ contains both a and b, the origin clause is simply $\{a\}$, and the theory contains only the clause $\{\neg a, b\}$, the characteristic clauses are both $\{a\}$ and $\{b\}$, while MILO-resolution will only produce $\{a\}$.

The reason is obvious. Suppose that in a deduction from a theory T, we skip a literal from a clause C_1, and MILO-resolution end with a leaf C_2. Then resolving such a literal instead of skipping it will produce, as a leaf, a clause C_3, and it is clear that $T \cup \{C_2\} \models C_3$. This set is called the $Deriv(T, C)$, and due to this last observation,

$$Deriv(T, C) \models F \quad \text{iff} \quad Carc(T + C) \models F,$$

thus showing that only $Deriv$ is needed.

Now, the derivatives are defined only operationally: they are the output of MILO-resolution. The above remarks help to characterize them semantically: they are the minimal set of clauses of

the production field from which the characteristic clauses can be semantically entailed together with the original theory, i.e. the minimal set S such that

$$T \cup S \models Carc(T + C).$$

As we will refer to such a set S later, let's call it the *minimal precursor* of the characteristic clauses.

4 Generalizing Przymusinki's Results

At the light of the previous discussion, let's try to generalize the above results.

Generally speaking, in any problem involving nonmonotonic inference, some formulae will behave monotonically: there will always be a theorem saying that some formulae will be valid in the extensions of the theory (or minimal, or preferred models) if and only if they are semantically entailed by the theory (valid in all the models). For example, in Bossu & Siegel's subimplication [1] in which all literals are minimized, these monotonic formulas are the positive ones. Przymusinski's Corollary 3.6 [4] shows that for circumscription the monotonic formulae are those formed by literals ¿from $P^+ + Q$. In general, there will be a subset of the representation language $\mathcal{L}$ that we can call the *monotonic* subset and denote $\mathcal{L}_M$, such that, (using Shoham's [6] notation in which $\models_\sqsubset$ means entailement by the preferred models,) for every F of $\mathcal{L}_M$, $T \models_\sqsubset F$ iff $T \models F$.

Now, loosely speaking, the answer to the question $T \models_\sqsubset F$ will be "yes", if T and $T + F$ are in some sense equivalent: that means that F has not added any new information to T. To answer this question we thus need to compute the theorems of $T + F$ and compare them to the theorems of T alone. As this computation is very expensive (NP-hard for non-Horn theories), we should look for the smallest subset of the theorems for which the property holds, i.e. the smallest subset for which the addition of F produces a change if and only if the answer is "no".

This smallest subset is the one formed by the theorems that belong to the monotonic subset, i.e., $Carc(T + F)$ when $\mathcal{P} = \mathcal{L}_M$.

As once the characteristic clauses of $T + F$ are computed, we will need to test whether or not they are implied by T alone, and owing to the discussion at the end of last section, we need only compute the minimal precursor of these characteristic clauses. *This is thus the smallest set needed to be generated in order to answer such a query.*

We now examine how Siegel and Przymusinski have found very similar algorithms to compute this set.

5 Przymusinski's and Siegel's Algorithms

In [7], Siegel gives an algorithm to solve problems 1 and 2, i.e. to compute the theorems of a set of clauses belonging to any stable production field $\mathcal{P}$. Although using very different data structures and notations, it appears that this algorithm is very close to Przymusinski's MILO-resolution; it has, however, some important new features. Rather than giving a complete description of the

algorithm that would involve introducing all the formalism used by it, we give the results of the mapping from one to the other, using Przymusinski's data structures and notations.

The following changes in Przymusinski's algorithm gives Siegel's. Among them, the first is by far the most important. It is a generalization of Przymusinski's algorithm that shows the algorithm need not make any reference to circumscription or minimal models and thus can be applied to other problems as well. The second one is strongly related to the discussion at the end of the last section: it shows that MILO-resolution computes the minimal precursor of the characteristic clauses, while Siegel's can either be used to compute this minimal precursor or all of the characteristic clauses. The third one indicates that the efficiency can be improved by reducing the clauses with which the current clause is resolved upon, thus reducing the search space.

1. Siegel's algorithm is more general: *No reference at all is made of circumscription, or literals from P, Q, or Z.* Instead it produces theorems belonging to any stable production field $\mathcal{P}$. Taking $\mathcal{P} = P^+ + Q$ gives MILO-resolution. So the first modification of MILO-resolution to give Siegel's is the following: in step (i) replace the words "that belongs to $Z + P^-$" by the words "that do not belong to $\mathcal{P}$". (N.B. Of course the important thing here is that all reference to literals relevant to circumscription are avoided and any stable $\mathcal{P}$ can be used instead. The other change, i.e. replacing "belongs" to "not belongs" and taking the complement of $Z + P^-$ (i.e. $P^+ + Q$) as production field is only made to stress the importance of $\mathcal{P} = P^+ + Q$ as representative of the minimal models).

2. Przymusinski skips the resolution to literals from $\mathcal{P}$. Siegel either skips them or resolves on them. When all literals have been skipped, the clause belongs to $\mathcal{P}$, and thus the algorithm stops.

   We showed in the last section the differences between skipping literals from $\mathcal{P}$ or not: Siegel proves his algorithm produces all the characteristic clauses, and resolving in all possible ways all literals is necessary for this to hold. Przymusinski skips those literals; this produces a minimal precursor of the characteristic clauses, and this is enough to correctly answer queries involving model-preference nonmonotonic inference. But not skipping them, i.e. computing all the characteristic clauses instead of a minimal precursor can be necessary for other problems.

3. Siegel includes two additional results that makes its search space smaller (of course, without sacrificing completeness neither in the general case of computing all theorems belonging to the production field, nor in the particular case of computing circumscription):

   (a) it restricts the resolution of the current clause with clauses from T *not having literals equal to framed literals at the right of the literal resolved upon*, as this would produce only clauses subsumed by some previous current clause.
   Example:

   $$F = \{a, d\}.$$

   Suppose F resolves with the following clause of T

   $$\{\neg a, b, c\}$$

giving

$$\{b, c, [a], d\}\,.$$

Now suppose there is a clause

$$\{\neg b, a\}$$

in T. The above restriction tells us that this clause may safely be skipped in the deduction, as it contains a, a literal appearing framed in the current clause. In effect, it would give the clause

$$\{a, [b], c, [a], d\}$$

which is subsumed (in the sense of OL-resolution, i.e. taking into account only unframed literals), by a previous current clause, F.

Clearly, this has two advantages: 1) it restricts the search space and 2) it avoids many of the subsumption tests in step (iii) of MILO-resolution [4, Definition 3.1].

(b) Let $C = C_1\, l\, C_2$ be one of the current clauses (the C_i are sets of literals), in which l is the next literal to be resolved upon (i.e., the first literal not from $\mathcal{P}$). Then if the clause $C_1\, C_2$ appears later in the deduction, all remaining choices from C can be avoided, as they will only produce subsumed clauses.

In the most trivial example, if the current clause is

$$F = \{a, b\}$$

and T contains

$$\{\neg a, b\}$$

then the clause

$$\{b\}$$

is produced (we did not write the framed literals as they are not relevant to this example).

In this case $C_1 = \phi$ and $C_2 = \{b\}$. Now if T contains, e.g., $\{\neg a, c\}$ then when backtracking occurs the algorithm can safely avoid making this resolution. In effect, it would generate

$$\{c, b\}$$

which is subsumed by the previously generated clause

$$\{b\}.$$

Siegel then proves the following results:

1. Decidability: The algorithm stops.

2. Soundness: Every clause produced from $T + F$ is implied by $T + F$. [3]

[3]The notation $T + F$ is used in [4] so we keep it to make the comparison clear. It means both the set $T \cup \{F\}$ and the linear resolution from the set T with origin F.

3. (Strong) Completeness: If T does not imply G, and $T + F$ implies G, then there is a deduction *from* $T + F$ (i.e., whose origin is F) that produces G.

These results also hold if we specify any stable production field.

Basically, results similar to these are also implicitly included in the proofs Przymusinski gives at the end of his paper. But again, these are independent of circumscription or minimal models and thus the algorithm can be applied — as we will shortly see — to other problems as well. In other words, Przymusinski's proofs mixes up two completely different results, namely: 1) the validity of Theorems 2.5 and 2.6 listed above, and 2) correctness and completeness of a particular resolution algorithm, that itself has nothing to do with circumscription or minimal models.

The following are some remarks concerning efficiency.

- Przymusinski is not very much concerned with efficiency problems: keeping the initial theory as it is forces his algorithm to do the same inferences over and over for different queries. However, some of these inferences can be made once and for all, without knowing the query. That is of course the motivation of our definition of characteristic clauses: they represent the compilation of the theory, i.e. all the inferences that can be made before the query is issued.

  But the two situations — queries involving or not involving predicates from Z — must be handled differently. Not dealing with predicates from Z is easy; in this case, the following can be proven by applying the definition:

  $$Newcarc(T, F) = Newcarc(Carc(T), F).$$

  This shows the initial theory T can be compiled into $Carc(T)$. When a query is issued, we just need to add it to $Carc(T)$, and compute the new characteristic clauses.

  Unfortunately, this is not the case if literals from Z are concerned. An easy counterexample is the following.

  $$T = \{Bird,\ \neg Bird \vee Ab \vee Flies\}.$$

  $Carc(T)$ is simply $Bird$, as the other theorem, $Ab \vee Flies$, is not in $\mathcal{P}$ ($Z = \{Flies\}$). Now adding $\neg Flies$ produces Ab as a new characteristic clause, because it resolves against a literal ¿from Z and thus makes the resulting clause be back in the characteristic clauses. However, the situation is not so bad: the original set can yet be simplified, computing its reduction, i.e. replacing it by the set of subclauses implied by T. In this example, this means that although T cannot be compiled into $\{Bird\}$, its only characteristic clause, it can be compiled into $\{Bird, Ab \vee Flies\}$. We develop on this while comparing the compiled/interpreted approach in next section.

- A problem concerning nonmonotonic inference (computing circumscription being only one example), will always need some kind of saturation, i.e. computing *all* the formulae of a certain production field. An ideal algorithm would directly produce these clauses. Unfortunately, such an algorithm does not exist, as any known algorithm produces many redundant (i.e. subsumed) clauses. If such a subsumed clause is produced, all deductions

that use this clause will be redundant too, so the price payed for producing such redundant clauses is very high. Of course as soon as a clause is produced, a subsumption test can be made, but this would involve making too many subsumption tests, which is very expensive too. So the key problem is to produce as less as possible redundant clauses and yet performing as little as possible subsumption tests.

This is exactly the motivation behind condition 3-a above, neither present in OL-resolution nor in MILO-resolution, and the relative efficiency of the algorithm comes in part from this condition. If the original set is "clean" (contains no subsumed clauses), none of the current clauses will be subsumed by any previous one neither by a clause that participated in the deduction. So the current clauses (and in particular the final one) can only be subsumed by a clause that did not take part in its deduction. For example, from the set

$$\{\,\{a,b\},\ \{a,c\},\ \{b,d\}\,\}$$

the deduction with origin

$$\{\neg c, \neg d, e\}$$

will produce (we do not write the framed literals here)

$$\{a,b,e\}$$

which is subsumed by $\{a,b\}$, that was not called in the deduction.

This is in fact the reason why a the "strong-soundness" property (dual of the strong-completeness) is missing from the list of the 3 results listed above. That property would be "every clause produced ¿from $T+F$ is implied by $T+F$ *but not by* T *alone*".

- Even if the algorithm presented here has many advantages regarding efficiency, computing circumscription still remains NP-complete, as it is equivalent to the decision problem in propositional logic. A very important point which is often neglected is to find heuristics to make NP-complete algorithms work efficiently in non-trivial problems. Siegel includes a study of such heuristics. Although this point is out of the scope of this paper, let us mention the following one as an example. First, the current clauses are written using a different data structure. For example, the framed clause

$$\{a,[b],c,[d],e\}$$

is written

$$(abd)(cd)(e).$$

So the first literal of each list represent unframed literals of OL- resolution, and the rest of each list represents the framed literals at the right. Now, this permits reordering the clause at each step: for example, this clause can be rewritten

$$(e)(abd)(cd)$$

without altering any of the properties of the algorithm.

Reordering the clause so that the literal whose opposite appears in the least number of clauses of T is put first, is a heuristic that has proved to substantially improve the complexity of the algorithm.

6 Reiter and de Kleer

Reiter and de Kleer [5] propose a generalization of the basic ATMS called the CMS (clause management system) and show its application to abductive reasoning and efficient search. A CMS is intended to work together with a reasoner, that issues queries that take the form of clauses. The CMS is then responsible for finding supports, which are clauses whose negations (called *explanations*) are the most general contexts in which those queries (called *observations*) hold.

The key concept is thus that of *support* for a query C with respect to a set of clauses Σ. Such a support S must verify the following:

1. $\Sigma \models S \cup C$
2. $\Sigma \not\models S$

Moreover, a support is *minimal* if no proper subset of it is a support.

Reiter and de Kleer then point out that, if we want to maximize the efficiency of the CMS once a query is issued, the set Σ should be saturated (i.e. containing every clause implied by it and with no redundant clauses), as in such a case, a clause logically follows from the set iff it is subsumed by a clause in the saturation. They call such a set the *prime implicants* and show some relationships between prime implicants and minimal supports.

Reexpressing the first condition as $\Sigma \cup \neg C \models S$, and setting the production field to be the set of all clauses, we see that

$$Minimal\_Supports(\Sigma, C) = Newcarc(\Sigma, \neg C).$$

However, $\neg C$ is now a conjunction of literals, not a clause. But this is no problem, as the following result holds:

Proposition: Let $H = C_1 \wedge \cdots \wedge C_m$ be a conjunction of clauses. Then

$$Newcarc(\Sigma, H) = \bigcup_{i=1}^{m} Newcarc(\Sigma_i, C_i),$$

where $\Sigma_1 = \Sigma$ and $\Sigma_{i+1} = \Sigma_i + C_i$, for $i = 1, \ldots, m-1$

if however we eliminate subsumed clauses from this last set.

In other words, we just need to run the algorithm with each clause, and then clean up the resulting set to eliminate redundant clauses.

In the case of de Kleer's ATMS, there is a distinguished subset of propositional symbols called the *assumptions*. The important point is that in this case, the output of CMS are clauses that only contain such literals. So the production field is that set of clauses containing only *negations* of assumptions. (Note that an explanation of a formula is a negation of a support for it.)

Finally, Reiter and de Kleer consider the tradeoffs between a compiled and an interpreted approach. In the former, the prime implicants of Σ are computed. This is very expensive, but

retrieval is then efficient. In the latter, CMS's database is kept as it is; the price we have to pay is a high retrieval cost.

We thus see that Siegel's algorithm will compute prime implicants and minimal supports. But as Reiter and de Kleer point out, that can also be done with a brute-force algorithm, that makes all possible resolutions until only subsumed clauses are produced. So let's consider efficiency problems compared with such an approach.

Computing supports using a compiled theory is not expensive. The serious problems are either computing supports with an uncompiled theory or compiling a theory. Regarding complexity, these problems are equivalent. The prime implicants can be computed using every clause as origin; for the first of such clauses, the original theory will probably be highly redundant; for the second one, a little less, and so on. Thus the interpreted approach is a particular case of the compiled one: the question is to know how efficient will be the algorithm working with a non-compiled theory.

In both situations, two cases have to be considered.

1. In the general case of the CMS, no distinguished subset of the language has a priority. Thus the relative efficiency of the algorithm as compared with a brute-force algorithm comes from the additional conditions needed to make a resolution: those involving framed literals. Among them, the last two remarks at the end of last section are particularly relevant.

2. In the case of the ATMS, the algorithm must produce clauses all whose literals are from a distinguished subset of the language. Of course in this situation, in addition to the above remarks that still hold, the algorithm performs much better as the search focuses on this restricted vocabulary.

But there is another interesting production field that offers an intermediate alternative to the compiled/interpreted disjunctive. If the production field is the set of *subclauses of clauses of* Σ, (note this is a stable production field) then Σ is reduced to a high extent. Computation from this new set will thus be much more efficient than from the former, and yet not all the theorems of the original set need to be computed.

For example, the set $\{\{a\}, \{\neg a, b\}\}$ will be transformed in $\{\{a\}, \{b\}\}$ (in this case producing the sub-clauses is equivalent to saturation). However, $\{\{a, c\}, \{\neg a, b\}\}$ contains all the sub-clauses implied by it and thus will not be simplified, while the saturation would have added $\{b, c\}$.

Before concluding this section, we should note that in the original setting of [5], the CMS and the ATMS are used for computing all minimal supports for the query. This is just the case of not skipping literals from the production field, which is described in the Siegel's condition 2 in the last section. But if we need the minimal precursor of the characteristic clauses instead of computing all the characteristic clauses, we can extend the notion of the CMS as described in section 3.2. We show in the next section how the concept of minimal precursors is useful for computing circumscription if we use an ATMS to answer whether or not a formula holds in all preferred models.

7 Ginsberg

Ginsberg [2] presents an another algorithm for computing circumscription. For the algorithm to work, assumptions as in Przymusinski need to be done: clausal logic, domain closure and uniqueness of names. The algorithm however works only in the case Q, the set of fixed predicates, is empty.

First, a result is presented which transforms the problem of circumscription into the problem of finding a confirming formula for the query. These confirming formulas are then computed using an ATMS.

We show here that Ginsberg's result is exactly the same as Przymusinski's second theorem, and leads to a straightforward generalization to the case in which not all predicates vary. These results were however expected, and Ginsberg mentioned both of them. But more interesting, Ginsberg uses an ATMS to compute confirming clauses, and claims this is the main difference between his work and [4]. Now, we have seen that Siegel's generalization of MILO-resolution does the same job that of an ATMS, namely computing minimal supports. Thus this algorithm subsumes both Przymusinski's and Ginsberg's and helps to understand the close connection between them.

We turn to each of these points now.

7.1 Results

Let's transform Ginsberg's definitions and results to ours.

Definition 3.1 (Ginsberg): D is a set of sentences. p is in dnf form wrt D if it can be written as a disjunction of conjunctions of elements of D. T is a set. Then q is confirmed by p wrt T and D if the following hold:

1. $T \cup \{p\}$ is satisfiable
2. $T \cup \{p\} \models q$
3. p is dnf wrt D.

In our terms, q is confirmed by p if

1. $T \not\models \neg p$
2. $T \cup \{\neg q\} \models \neg p$
3. $\neg p$ is a set of clauses (i.e., a conjunction of disjunctions) of the production field $\neg D$.

Or, in other words,

Definition 3.1 (New Version): q is confirmed by p if $\neg p$ is in $Newcarc(T, \neg q)$.
Moreover, q is unconfirmed, if no p confirms q: i.e., if $Newcarc(T, \neg q) = \phi$.

Next is the main result:

Proposition 3.2 (Ginsberg): $CIRC(T;P;Z) \models q$ if and only if there is some p confirming q so that $\neg p$ is unconfirmed.

which we can rewrite

Proposition 3.2 (New Version): Set the production field to $\neg D$. $CIRC(T;P;Z) \models q$ if and only if $Newcarc(T, \neg q)$ is a formula p such that $Newcarc(T, \neg p) = \phi$.

Ginsberg briefly mentions connections with Przymusinski's work and the possibility of relaxing the assumption of all non-minimized predicates being variable. Our above results show that:

1. This last proposition is *exactly* Przymusinski's theorem 2.6.
2. All results can be extended to the case $Q \neq \phi$ (i.e., not varying all predicates) just by setting $D = P^- + Q$, that is, $\mathcal{P} = \neg D = P^+ + Q$.

7.2 Algorithm

Let's now compare Ginsberg's algorithm with Przymusinki's. Both start with exactly the same input and produce exactly the same output, so let's see where the computational differences lie.

Let us remark that this comparison is somewhat difficult to make, as Ginsberg does not give an implementation algorithm, but uses an ATMS based on his multivalued logic [3], whose applicability is much wider to this specific problem. But still enough information is present to draw many conclusions.

Ginsberg's algorithm only works for the case $Q = \phi$; as we showed that it can be easily extended to the case $Q \neq \phi$, this is no problem: the following discussion is valid in either case. But of course it would be of no use to compare Ginsberg's original setting where $Q = \phi$ with the general problem in which $Q \neq \phi$: in the former case, the production field is simply P^+, which is *much* smaller than $P^+ + Q$, and as the efficiency of the algorithm depends critically on the size of the production field, the comparison will not be very interesting.

Ginsberg, referring Przymusinski, says that "the algorithm he develops appears to be substantially less efficient than the one we propose"[2, page 100]. We believe this is wrong. In fact the *only* advantage of Ginsberg's algorithm appears to be the following: as it uses an ATMS, it records some inferences so that they are made only once, while Przymusinski's might do some of those inferences unnecessarily over and over again.

We already have pointed out this problem in Section 5. Again, that is the motivation behind the definition of characteristic clauses, and this solution, of course more efficient than Przymusinski's, appears to be also better than Ginsberg's: we don't need to wait for a query to be issued and, while answering to it, record the inferences. We make all possible inferences before the query is issued, and so minimize the answering process.

Now, to make the competition fair, suppose that Ginsberg starts with the characteristic clauses too, or with whatever set he thinks might the best one to answer the query faster.

We believe MILO-resolution will be much more efficient, and if we add Siegel's improvements, the difference will be still greater. Our argument will be as follows: first, we defined the minimal set of formulae that need to be generated in order to correctly answer a query; we show Ginsberg generates more than these formulae. More precisely, Ginsberg generates all the characteristic clauses, while Przymusinki-Siegel (subsequently denoted PS) generates only their minimal precursor. Second, even if we fix a set of formulae to be generated, PS will generate it much more efficiently, because it has available information concerning (a) the production field, and (b) the framed literals, while Ginsberg uses a conventional theorem prover [3, Procedure 10.5].

Let's develop on this.

1. Ginsberg generates all the characteristic clauses, thus more formulae than those needed.

   Note that this has two disadvantages: the cost of generating those additional formulae, and the cost of testing if the negation of these produces no new characteristic clause (i.e., what Ginsberg calls testing if the negation of the confirmation is unconfirmed).

   The example is the following: T is

   $$\{ P_1 \vee \neg P_2, \quad P_2 \vee \neg P_3, \quad P_3 \vee Z \}$$

   The production field is $\mathcal{P} = P^+$, i.e., positive occurrences of P_1, P_2, and P_3. The query is Z.

   By adding $\neg Z$ to T, PS generates only P_3, the only new theorem that belongs to $\mathcal{P}$. This literal belongs to $\mathcal{P}$ and thus the algorithm skips it and stops. It then adds $\neg P_3$ to T which generates no characteristic clause, showing that $CIRC(T; P; Z) \models Z$.

   Let's see what Ginsberg does. The set of assumptions is $D = \neg\mathcal{P} = P^-$. Then, the confirmation of Z is dnf:

   $$\neg P_3 \vee \neg P_2 \vee \neg P_1.$$

   The negations of the confirmation is

   $$P_3 \wedge P_2 \wedge P_1,$$

   which is unconfirmed.

   So the ATMS has produced two additional contexts, $\{\neg P_1\}$ and $\{\neg P_2\}$, in which Z holds (the three are produced as neither is a subset of another). PS did not need to generate them. As explained above, the reason is that $\{P_3\}$ is the minimal precursor of the others, as

   $$T \cup \{P_3\} \models P_1 \wedge P_2.$$

2. Even if PS generated such a larger set, it would do it in a more efficient way, and for two independent reasons:

   (a) It uses the information on the production field not at all present in Ginsberg, who uses a conventional theorem prover. Thus, it generates far less clauses.

For a very simple example, suppose the query is $Z \vee Q$, where Z is not in the production field, while Q is. If Z cannot be resolved against clauses of the theory in such a way that the result of the resolution produces a clause in the production field, PS will never try to resolve on Q. Of course the conventional theorem prover will give no priority to Z over Q and thus try all the resolutions on Q as well.

This example, although trivial, is representative of what will happen in more realistic situations. In general, maybe Z will be resolved against literals belonging to the production field, thus producing characteristic clauses. But all the failed branches will produce backtracking, while a conventional theorem prover will examine the next literal, Q, making unnecessary computation.

(b) PS will generate less redundant clauses, and yet make less redundancy tests.

This has extensively been discussed above. The reason here is the information in the framed literals, not present in Ginsberg.

In particular, the additional test Siegel introduces, not present in MILO-resolution, is a fundamental one for this purpose. For this problem, however, Ginsberg has some answer: his bilattice records some clauses produced by the algorithm in order to avoid producing redundant ones. In his terms, this means only keeping justifications which are less general than others in [3, Procedure 10.9]. It is difficult to evaluate the behavior of the algorithm at this point as it is not clear when those subsumption tests are done. But again, doing the subsumption tests is less efficient than just avoiding in the deduction the resolution with clauses that will necessarily produce redundant ones. To be convinced, read again Section 5.

8 Conclusion

We have tried to show that many different problems concerning nonmonotonic proof procedures follow exactly the same pattern:

- The key problem is to know if the query has altered the preferred models of the theory.
- A subset of the theorems of such a theory is representative of the minimal models, in the sense that a change in this set denotes a change in the minimal models as well.
- These representative formulae are the ones constructed on sub-vocabulary of the language, the one on which the (nonmonotonic) inference rule behaves monotonically.
- As an implication test is necessary after generating such representative formulae, only a subset of these is necessary: the minimal precursor, which is thus the smallest set that needs to be generated to correctly answer a query.

We then compared Siegel's and Przymusinski's algorithm that solve this problem, and showed many connections with Reiter & de Kleer's and Ginsberg's work.

We believe many improvements concerning efficiency can still be done. We are currently investigating this at ICOT.

References

[1] Genevieve Bossu and Pierre Siegel: Saturation, Nonmonotonic Reasoning, and the Closed-World Assumption. *Artificial Intelligence* 25:23–67, 1985.

[2] Matthew Ginberg: A Circumcriptive Theorem Prover. *Proc. 2nd Workshop on Nonmonotonic Reasoning.* LNCS 346, Springer-Verlag, 1989. Also to appear in *Artificial Intelligence.*

[3] Matthew Ginberg: Multivalued Logics: A Uniform Approach to Reasoning in Artificial Intelligence. *Computational Intelligence* 4:265–316, 1988.

[4] Teodor Przymusinski: An Algorithm to Compute Circumscription. *Artificial Intelligence* 38:49–73, 1989.

[5] Raymond Reiter and Johan de Kleer: Foundations of Assumption-Based Truth Maintenance Systems: Preliminary Report. *Proc. AAAI-87.*

[6] Yoav Shoham: Reasoning About Change: Time and Causation From the Standpoint of Artificial Intelligence. MIT Press. 1987.

[7] Pierre Siegel: Représentation et Utilisation de la Connaissance en Calcul Propositionnel. Thèse d'État. Université d'Aix-Marseille II. 1987.