A Circumscriptive Theorem Prover: Preliminary Report*

Matthew L. Ginsberg
Computer Science Department
Stanford University
Stanford, California 94305

Abstract

We discuss the application of an assumption-based truth maintenance system to the construction of a circumscriptive theorem prover, showing that the connection discovered by Reiter and de Kleer between assumption-based truth maintenance and prime implicants relates to the notions of minimality appearing in nonmonotonic reasoning.

The ideas we present have been implemented, and the resulting system is applied to the canonical birds flying example and to the Yale shooting problem. In both cases, the implementation returns the circumscriptively correct answer.

1 Introduction

Circumscription [McCarthy, 1980; McCarthy, 1986] is probably the most thoroughly investigated of all of the approaches to nonmonotonic reasoning. Unfortunately, these investigations have not led to the development of effective algorithms for determining whether or not some potential conclusion follows from the circumscription axiom. There have been several attempts at this [Lifschitz, 1985; Przymusinski, 1986], but none has been completely satisfactory.

This paper begins to address these difficulties. The approach we will be taking is the result of combining a variety of ideas relating to nonmonotonic inference:

1. The formalization of the connection between circumscription and minimal models. This appears to be due to Lifschitz [Lifschitz, 1985], and is also present in McCarthy's original paper [McCarthy, 1980] and recent work of Shoham's [Shoham, 1987].
2. A description of minimal models in terms of specific formulas describing them. This idea appears in Gelfond et. al.'s idea of a *free for negation* sentence [Gelfond *et al.*, 1986], and in the concept of a *prime implicant* discussed by Reiter and de Kleer [Reiter and de Kleer, 1987].
3. The observation that an assumption-based truth maintenance system (ATMS) [de Kleer, 1986] can be used to generate the formulae mentioned above. This observation appears in both [Reiter and de Kleer, 1987] and [Ginsberg, 1988].
4. The construction of a backward-chaining ATMS and the implementation of a backward-chaining circumscriptive theorem prover using it. Construction of a backward-chaining ATMS is discussed in [Ginsberg, 1988], and examples are presented there as well.

The subsequent sections of this paper consider each of these points in turn. In the next section, Section 2, we discuss the connection between the circumscription axiom and constructions involving minimal models.

In Section 3, we go on to show that these minimal models can be described in terms of a notion we will refer to as *confirmation*. Loosely speaking, a sentence is confirmed if it would follow from the closed world assumption [Reiter, 1978] applied to some predicate. We will discuss circumscription in terms of sentences whose negations are *not* confirmed.

Next, in Section 4, we describe the relationship between ATMS's and confirmation. The description we will give of circumscription involves the idea of a "weakest" confirming sentence; this is closely related to the notion of a *prime implicant* appearing in [Reiter and de Kleer, 1987].

The implementation itself is discussed in Section 5. Here, we argue that the notion of a *bilattice* [Ginsberg, 1988] can be used to construct a backward-chaining ATMS of the sort needed by a backward-chaining circumscriptive theorem prover. Finally, we present examples of the prover in operation in Section 6.

2 Circumscription and models

As remarked in the introduction, the description of circumscription that we will be using is one based on models, as opposed to the usual circumscription axiom appearing in [McCarthy, 1986].

Suppose, then, that D is some finite set of defaults; one might, for example, take D to be the set of all propositions of the form $\neg p(x)$, where p is a predicate being circumscribed and x is an instantiation of p's argument. Given two models M_1 and M_2, we will write $M_1 \leq_D M_2$ just in case the set of elements of D that hold in M_1 is a subset of the set of elements of D that hold in M_2. Given a collection of models, we will refer to the ones that are maximal under the partial order $\leq_D$ as *D-maximal* elements of the collection. The following result is now an easy consequence of Proposition 1 of [Lifschitz, 1985]:

Proposition 2.1 *Let T be a set of sentences without function symbols, such that T includes domain closure and uniqueness of names assumptions. Let p be a predicate, and let D be the set of all propositions of the form $\neg p(x)$, where x is a ground instantiation of p's argument. Now*

*This work has been supported by DARPA under grant number N00039-86-C-0033, by ONR under grant number N00014-81-K-0004, by NSF under grant number DCR-8620059, by the Rockwell Palo Alto Laboratory, and by General Dynamics.

for any sentence q, q is a consequence of circumscribing p in T while allowing all other predicates to vary if and only if q holds in all D-maximal models of T.

The approach we will take will be to work directly from the sets T and D, as opposed to working from the circumscription axiom itself.

Before proceeding, however, we should spend some time discussing Proposition 2.1. The assumptions made regarding T are essentially those needed to ensure both that we can enumerate all possible instantiations of p using constant symbols appearing in the original theory, and that these instantiations will be nonequivalent; these are quite strong assumptions.[1] It appears that both Proposition 2.1 and our implementation based on it will be valid in somewhat wider circumstances; this issue is currently under investigation by Gelfond and Lifschitz [Gelfond and Lifschitz].

Even in the general case where the proposition may fail, however, the argument can be made that an algorithm to determine whether or not some sentence q holds in all D-maximal models of T is of comparable interest to one that determines whether or not q holds after circumscribing p in T: Recall that the original motivation for the circumscription axiom was to help characterize precisely the notion of predicate minimization appearing in the proposition.

3 Confirmation

Technically, the reason we will work with Proposition 2.1 instead of the circumscription axiom itself is that it is possible to recast the proposition in a somewhat more useful form. We need the following definition:

Definition 3.1 *Let D be a set of sentences. We will say that a sentence p is* dnf with respect to D *if p is of the form*

$$\bigvee_i \bigwedge_j d_{ij}$$

for some collection of $d_{ij} \in D$. In other words, p is the disjunction of conjunctions of elements of D.

Now also fix a set T. Then we will say that some sentence q is confirmed by p for D and T *if the following conditions hold:*

1. *$T \cup \{p\}$ is satisfiable,*
2. *$T \cup \{p\} \models q$, and*
3. *p is dnf with respect to D.*

If no ambiguity is possible, we will simply say that q is confirmed by p. *If there is no p that confirms q, we will say that q is* unconfirmed. *If p confirms $\neg q$, we will say that p* disconfirms q.

The reason the above definitions are useful is because of the following result:[2]

Proposition 3.2 *Let T and D be sets of sentences, and q a sentence. Then q holds in all D-maximal models of T if and only if there is some p that confirms q such that $\neg p$ is unconfirmed.*

[1] I am indebted to Vladimir Lifschitz to providing me with a sharp formulation of them (personal communication).

[2] Proofs can be found in the full paper.

It is clear from this result and Proposition 2.1 that, given information about confirmation, we can determine whether or not some query q follows from a given circumscription.

4 Confirmation and truth maintenance

In order to use Proposition 3.2 effectively, we need some way to determine the various sentences that confirm some query q.

Suppose we consider the collection of all sentences that confirm q. Now it is fairly clear that the disjunction of all of these sentences also confirms q, and that the negation of this disjunction will be unconfirmed if and only if q satisfies the conditions of Proposition 3.2. Thus we can determine whether or not q holds in all D-maximal models of T by considering only the *weakest* of the sentences that confirm it.

This problem has in fact already been addressed in the AI literature. In [Reiter and de Kleer, 1987], Reiter and de Kleer show that assumption-based truth maintenance systems (ATMS's) perform just this sort of calculation; similar remarks can be found in [Ginsberg, 1988].

Essentially, an ATMS takes a proposition such as q and determines the 'environment' in which q holds. One possible representation for this environment is as a list of contexts, where each context is described by a list of assumptions that must hold in it. If we take the elements of D as our possible assumptions, we see that the ATMS environments correspond to our dnf formulae. Since the ATMS is looking for a minimally specific environment in which q holds, we can think of it as looking for a weakest sentence p that confirms q. Similarly, the ATMS label assigned to $\neg p$ will tell us whether or not $\neg p$ is unconfirmed.

5 Implementation

There are some subtleties involved in actually implementing these ideas. Firstly, a (presumably backward-chaining) circumscriptive theorem prover will rely on a backward-chaining ATMS; de Kleer's published work [de Kleer, 1986] is based on forward-chaining methods. In addition, de Kleer's work only describes an ATMS for propositional calculus; a circumscriptive theorem prover will need to work with a fully first-order version. We now turn to the problem of constructing an ATMS with these properties.

5.1 Backward chaining

Construction of a backward chaining ATMS is discussed in [Ginsberg, 1988].[3] The essential idea is to construct a bilattice that corresponds to deKleer's construction, and to then use the general-purpose algorithms described in [Ginsberg, 1988] to produce a backward chainer that uses this bilattice.

A description of the ATMS bilattice can also be found in [Ginsberg, 1988]. This bilattice is constructed using the fact that the environments described in the last section can be partially ordered by generality.

[3] Reiter and de Kleer describe this as the "interpreted approach" to truth maintenance in [Reiter and de Kleer, 1987]. No algorithm is presented, however.

It is clear that the contexts themselves, viewed simply as subsets of the set of all possible assumptions, can be partially ordered by set inclusion. Thus if $c = c_1 \wedge \ldots \wedge c_m$ and $d = d_1 \wedge \ldots \wedge d_n$ are two contexts, we can take $c \leq d$ to mean that the set of c_i's contains the set of d_j's as a subset. In other words, $c \leq d$ if the context c is *less general* than the context d.

Using this partial order, we can construct a partial order on the environments themselves, saying that one environment e_1 is less general than another e_2 if *every* context in e_1 is less general than *some* context in e_2. If every context in e_1 contains some context in e_2 as a subset, then the environment e_1 is less general than the environment e_2.

The points in bilattices corresponding to ATMS's consist of pairs of environments $\langle e_1, e_2 \rangle$. If a sentence p has truth value $\langle e_1, e_2 \rangle$, this means that e_1 is the most general environment in which p is known to hold, and e_2 is the most general environment in which $\neg p$ is known to hold.

5.2 First-order ATMS's

Next, we discuss the construction of a first-order ATMS, as opposed to a propositional one. In general, we need to consider the possibility that the sentences appearing in the various ATMS contexts be quantified in some way.

We will assume that this quantification can be handled implicitly, as in PROLOG [Clocksin and Mellish, 1981]. Existential quantification will be handled via Skolemization, and universal quantification will be handled by assuming that any free variables appearing in the database are universally quantified.[4]

We will now construct contexts in a fashion quite similar to that of the last section; from these contexts, we construct environments exactly as before. It follows that in order to extend our propositional ATMS to a first-order one, we need to extend the partial order we previously defined for contexts. Recall that this partial order defined a conjunctive context c to be less general than a context d just in case every sentence in c was also in d.

In the first-order case, we need to modify the definition only slightly, saying that a context c is less general than another context d if and only if, for each sentence c_i appearing in c, there is some sentence d_j appearing in d such that d_j is an *instantiation* of c_i.

Thus, for example, the context consisting of the single sentence $p(x)$ is less general than the context consisting of the sentence $p(a)$, where x is a variable and a is a constant. This is as it should be, since $p(a)$ surely holds if $\forall x.p(x)$ does.

In what follows, we will represent contexts using propositions and binding lists, so that the context consisting of the sentence $p(a)$ might well be represented as

$$(p(x) \;.\; \{x = a\}).$$

The reason we do this is that we will frequently have assigned designators to the universally quantified propositions, and this representation is slightly more compact than restating the proposition involved. Thus an element of any particular context will generally have the form

$$(p \;.\; \sigma),$$

where p is an assumption (i.e., an element of D) and σ is a binding list indicating which variables in p are bound in the context. If D consists of the single sentence schema $\neg\mathrm{ab}(x)$, a context depending on $\neg\mathrm{ab}(a)$ and $\neg\mathrm{ab}(b)$ would be written as:

$$\{(\neg\mathrm{ab}(x) \;.\; \{x = a\}), (\neg\mathrm{ab}(x) \;.\; \{x = b\})\}.$$

In the LISP-like notation to be used in the next section, we would write:

$$((\neg\mathrm{ab}(x) \;.\; \{x = a\})(\neg\mathrm{ab}(x) \;.\; \{x = b\})).$$

Environments will be written as lists of contexts.

Using this notation, we see that if the value assigned to q after computing the bilattice closure is $\langle e, f \rangle$, where $e = \{c_1, \ldots, c_m\}$ and

$$c_i = \{(d_{ij} \;.\; \sigma_{ij})\},$$

then the weakest sentence confirming q is

$$\bigvee_i \bigwedge_j d_{ij}|_{\sigma_{ij}}.$$

6 Examples

In this section, we present three examples of the implemented system at work: the usual birds flying example, a simple example involving disjunction, and the Yale shooting problem. All of the 'output' shown is as actually produced by the program, except for trivial textual modifications. (For example, the database is maintained in disjunctive normal form, but is displayed below using a PROLOG-like representation.)

6.1 Birds flying

Here is the database for the usual birds flying example:

```
Bird(Tweety).
Ostrich(Fred).
Flies(x) :- Bird(x),Not(Ab(x)).
Bird(x) :- Ostrich(x).
Not(Flies(x)) :- Ostrich(x).
Not(Ab(x)).                                P4
```

Tweety is a bird, and Fred is an ostrich. Birds fly unless they are abnormal; ostriches are birds that don't fly. By default, nothing is abnormal. The P4 labelling the statement that nothing is abnormal serves both to indicate that this is a default rule, and to give the ATMS a label for this rule.

We now ask the inference engine to find something that flies by giving it the query `Flies(x)`. Here is the result:

```
Flies(x)?
  Invoking multivalued prover on Flies(x).
  Value returned is:
    binding list: {x = Tweety}
    truth value:  (((P4 . {x = Tweety})))
  Checking circumscriptive context for truth
    value (((P4 . {x = Tweety}))).
  Checking confirmation for Not(Ab(Tweety)).
  Negation is unconfirmed.
Success!  x = Tweety.
```

[4] Ray Reiter has pointed out to me that not all first-order theories can be written in this fashion. It is not clear how difficult it will be to generalize the following construction to handle these situations.

The prover begins by invoking a multivalued prover on the query Flies(x) [Ginsberg, 1988]. The multivalued prover succeeds, since it can find a proof of Flies(x) for x bound to Tweety. The truth value returned contains justification information. In this case, the proof that Tweety can fly used the proposition P4 with x bound to Tweety.

This means that Flies(Tweety) is confirmed by Not(Ab(Tweety)), so the prover next checks to see if it can find some confirmation for the negation of this statement. Since Not(Not(Ab(Tweety))) is unconfirmed, the prover succeeds, returning the answer x = Tweety.

Alternatively, we can look for something that *doesn't* fly:

```
Not(Flies(x))?
  Invoking multivalued prover on Not(Flies(x)).
  Value returned is:
    binding list: {x = Fred}
    truth value:  TRUE
  Checking circumscriptive context for truth
    value TRUE.
  Negation is unconfirmed.
Success!  x = Fred.
```

The prover is able to show that Fred cannot fly without using any assumptions at all (recall that only the default rule about abnormality is a possible assumption; the other database facts are accepted unconditionally). Since TRUE cannot be disconfirmed, the prover informs us that Fred cannot fly.

6.2 A disjunctive example

We next turn to an example from [McCarthy, 1980]:

```
Block(a) :- Not(Block(b)).
Not(Block(x)).                    P15
```

We are told that either a is a block or b is, and circumscribe the predicate block. The result of the circumscription should be that either a is the *only* block, or b is. It follows from this that

$$\neg[\texttt{block}(a) \wedge \texttt{block}(b)]$$

should be circumscriptively valid:

```
Not(And(Block(a),Block(b)))?
  Invoking multivalued prover on
    Not(And(Block(a),Block(b))).
  Value returned is:
    binding list: {}
    truth value:  (((P15 . {x = a}))
                   ((P15 . {x = b})))
  Checking circumscriptive context for truth
    value
    (((P15 . {x = a})) ((P15 . {x = b}))).
  Checking confirmation for
    Or(Not(Block(a)),Not(Block(b))).
  Negation is unconfirmed.
Success!
```

This example is slightly more difficult than the preceding one. The multivalued prover manages to prove the query by using either P15 with x bound to a, or with x bound to b. Thus the query is confirmed by:

$$p \equiv \neg\texttt{block}(a) \vee \neg\texttt{block}(b).$$

The negation of p is $\texttt{block}(a) \wedge \texttt{block}(b)$, and this appears to itself be confirmed by P15 applied to *both* of a and b (since applying P15 to a allows us to conclude that b is a block, and similarly for applying it to b). Thus the negation of p is apparently confirmed by:

$$\neg\texttt{block}(a) \wedge \neg\texttt{block}(b).$$

This sentence is inconsistent with our theory, however, since we are assuming that either a or b is a block. Thus $\neg p$ is in fact unconfirmed, and the prover returns with success.

6.3 The Yale shooting

Finally, we discuss the well known Yale shooting example from [Hanks and McDermott, 1987]:

```
Not(Holds(alive,Do(shoot,s))) :-
  Holds(loaded,s).
Holds(p,Do(a,s)) :-
  Holds(p,s), Not(Ab(a,p,s)).
Holds(loaded,s0).
Holds(alive,s0).
Not(Ab(a,p,s)).                   P23
```

The example involves a gun and an intended victim (generally named Fred). If the gun is loaded in a state s, then Fred will not be alive in the state resulting from firing the gun. The second axiom is a frame axiom, telling us that a proposition p will hold after performing an action a in a state s if p held before performing the action, unless the triple (a, p, s) is abnormal. The gun is loaded and Fred is alive in the initial state s_0. Finally, actions are (by default) not abnormal.

The question is this: If we wait and then fire the gun, do we kill Fred? Surprisingly, the answer is no, since there are two different ways in which we might apply the default rules. In the first (the intended one), waiting has no effect, and the action of firing the gun is abnormal in the state Do(wait,s0) because Fred becomes not alive. The second possibility is one in which the waiting action is abnormal and the gun becomes mysteriously unloaded.

If we ask the circumscriptive theorem prover whether Fred is alive after we wait and fire the gun, here is the result:

```
Holds(alive,Do(shoot,Do(wait,s0)))?
  Invoking multivalued prover on
    Holds(alive,Do(shoot,Do(wait,s0))).
  Value returned is:
    binding list: {}
    truth value:
      (((P23 . {a = wait, p = alive, s = s0})
        (P23 . {a = shoot, p = alive,
                           s = Do(wait,s0)})))
  Checking circumscriptive context for truth
    value
      (((P23 . {a = wait, p = alive, s = s0})
        (P23 . {a = shoot, p = alive,
                           s = Do(Wait,s0)}))).
  Checking confirmation for
    And(Not(Ab(wait,alive,s0)),
        Not(Ab(shoot,alive,Do(wait,s0)))).
  Negation confirmed based on truth value
    (((P23 . {a = wait, p = loaded, s=s0}))).
Fails.
```

The system first notes that it can prove that Fred is alive simply by applying the frame axiom twice, first to conclude that he is alive after the waiting action, and then to conclude that he remains alive after the shooting action. Thus his being alive is confirmed by:

$$\neg ab(\texttt{wait}, \texttt{alive}, s_0) \wedge \neg ab(\texttt{shoot}, \texttt{alive}, \texttt{do}(\texttt{wait}, s_0)).$$

The negation of this sentence is confirmed by

$$\neg ab(\texttt{wait}, \texttt{loaded}, s_0),$$

however. If wait is not abnormal in s_0, then the gun will be loaded after the waiting action, and Fred will necessarily be dead after the gun is fired. Since there is no proof of the negation of this sentence, the confirming sentence for the original query is itself disconfirmed, and the query does not follow from the circumscription.

Alternatively, we can try to show that Fred is *not* alive after the sequence of actions:

```
Not(Holds(alive,Do(shoot,Do(wait,s0))))?
  Invoking multivalued prover on
    Not(Holds(alive,Do(shoot,Do(wait,s0)))).
  Value returned is:
    binding list: {}
    truth value:
      (((P23 . {a = wait, p = loaded, s = s0})))
  Checking circumscriptive context for truth
    value
    (((P23 . {a = wait, p = loaded, s = s0}))).
  Checking confirmation for
    Not(Ab(wait,loaded,s0)).
  Negation confirmed based on truth value
    (((P23 . {a = wait, p = alive, s = s0})
      (P23 . {a = shoot, p = alive,
              s = Do(wait,s0)}))).
Fails.
```

Now a proof is found indicating that Fred will not be alive after the gun is fired, provided that waiting did not cause it to become unloaded. The negation of the confirming fact is $ab(\texttt{wait}, \texttt{loaded}, s_0)$, but this follows from the conjunction

$$\neg ab(\texttt{wait}, \texttt{alive}, s_0) \wedge \neg ab(\texttt{shoot}, \texttt{alive}, \texttt{do}(\texttt{wait}, s_0)).$$

Since the negation of the above sentence cannot be proven, the confirmation of

$$\neg \texttt{holds}(\texttt{alive}, \texttt{do}(\texttt{shoot}, \texttt{do}(\texttt{wait}, s_0)))$$

is disconfirmed, and the prover fails.

In all of these examples, the prover returns the correct circumscriptive answer. In addition, the computational procedure used is reasonably efficient. The Yale shooting problem, for example, is solved in approximately one second on a Symbolics 3600.

Acknowledgement

I would like to thank Johan de Kleer, Benjamin Grosof, John McCarthy, Karen Myers, Ray Reiter and David Smith for many useful discussions and suggestions. I am especially grateful to Vladimir Lifschitz for the assistance he has given me in sharpening the results of Section 2, and in directing me to Gelfond's paper [Gelfond *et al.*, 1986].

References

[Clocksin and Mellish, 1981] W. F. Clocksin and C. S. Mellish. *Programming in Prolog.* Springer-Verlag, Berlin, 1981.

[de Kleer, 1986] Johan de Kleer. An assumption-based truth maintenance system. *Artificial Intelligence*, 28:127–162, 1986.

[Gelfond and Lifschitz] Michael Gelfond and Vladimir Lifschitz. Compiling circumscriptive theories into logic programs. In preparation.

[Gelfond *et al.*, 1986] Michael Gelfond, Halina Przymusinska, and Teodor Przymusinski. The extended closed world assumption and its relationship to parallel circumscription. In *Proceedings of ACM SIGACT-SIGMOD Symposium on Principles of Database Systems*, pages 133–139, 1986.

[Ginsberg, 1988] Matthew L. Ginsberg. Multivalued logics: A uniform approach to reasoning in artificial intelligence. *Computational Intelligence*, 4, 1988.

[Hanks and McDermott, 1987] Steve Hanks and Drew McDermott. Nonmonotonic logics and temporal projection. *Artificial Intelligence*, 33:379–412, 1987.

[Lifschitz, 1985] Vladimir Lifschitz. Computing circumscription. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 121–127, 1985.

[McCarthy, 1980] John McCarthy. Circumscription – a form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[McCarthy, 1986] John McCarthy. Applications of circumscription to formalizing common sense knowledge. *Artificial Intelligence*, 28:89–116, 1986.

[Przymusinski, 1986] Teodor C. Przymusinski. Query-answering in circumscriptive and closed-world theories. In *Proceedings of the Fifth National Conference on Artificial Intelligence*, pages 186–190, 1986.

[Reiter, 1978] Ray Reiter. On closed world data bases. In H. Gallaire and J. Minker, editors, *Logic and Data Bases*, pages 119–140, Plenum, New York, 1978.

[Reiter and de Kleer, 1987] Raymond Reiter and Johan de Kleer. Foundations of assumption-based truth maintenance systems: Preliminary report. In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 183–188, 1987.

[Shoham, 1987] Yoav Shoham. A semantical approach to nonmonotonic logics. In *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, pages 388–393, 1987.