ICOT Technical Report: TR-611

TR-611

Query Answering in Circumscription

by
N. Helft, K. Inoue & D. Poole

January, 1991

ICOT
Mita Kokusai Bldg. 21F
4-28 Mita 1-Chome
Minato-ku Tokyo 108 Japan
(03)3456-3191~5
Telex ICOT J32964

Institute for New Generation Computer Technology

Query Answering in Circumscription *

Nicolas Helft†
Artificial Intelligence Center
SRI International
333 Ravenswood Ave.
Menlo Park, CA 94025
U. S. A.

Katsumi Inoue
ICOT Research Center
Institute for New Generation Computer Technology
Mita Kokusai Bldg. 21F.
1-4-28 Mita, Minato-ku, Tokyo 108,
Japan

David Poole‡
Department of Computer Science
University of British Columbia
Vancouver, B.C.
Canada V6T 1W5

April 13, 1991

*This paper is also to appear in *The Twelfth International Joint Conference on Artificial Intelligence* (IJCAI-91), Sydney, Australia, August 1991.

†Supported by the NTT Corporation, and by the Office of Naval Research under Contract No. N00014–89–C–0095.

‡This research was supported under NSERC grant OGPOO44121.

Abstract

We address the problem of answering queries in circumscription and related nonmonotonic formalisms. The answering process we describe uses resolution-based theorem provers recently developed for circumscription. In a way analogous to query answering techniques in classical predicate logic, the process extracts information from a proof of the query. Circumscriptive theorem provers consist of two processes, generating explanations for the theorem to be proved and showing that these explanations cannot be refuted. In general, many explanations compete in supporting the theorem. We show that queries can be answered by finding certain combinations of explanations, and present results to search the space of explanations, while carrying out significant pruning on this space. The results are relevant to other nonmonotonic formalisms having explanation-based proof procedures.

Keywords: circumscription, answer extraction

1 Introduction

For the first-order predicate logic, techniques developed by Green [1969a] are the basis for query-answering systems extensively used in deductive databases, logic programming and synthesis problems such as planning. These techniques rely on resolution-based theorem provers that attempt to prove the query while keeping track of the information generated during the proof. Theorem provers can decide whether a query follows from a given theory, and thus answer questions such as "Is there a coffee cup?"; the corresponding query-answering procedure computes the instance for which the query holds and can provide answers to questions such as "Where is the coffee cup?".

This paper addresses the query answering problem for logic databases augmented with a circumscription axiom [McCarthy, 1986] and related nonmonotonic formalisms. As in the first-order case, the answering procedure we present extracts information from a proof of the query. We build on existing proof procedures for circumscription that have recently been developed [Przymusinski, 1989; Ginsberg, 1989; Inoue and Helft, 1990].

While existing theorem provers for circumscription can correctly answer whether or not a formula follows from a circumscription, Green's techniques, although necessary, are not sufficient to provide answer extraction. The reason is the following. Circumscriptive theorem provers are based on finding explanations, or arguments, for the theorem to be proved, and showing that these explanations cannot be refuted. In general, many explanations compete in supporting the theorem, and a certain combination of these has to be found. We show that to an informative answer corresponds a particular combination of explanations, and present a procedure to find these combinations, together with results to search this space and carry out significant pruning.

Although we focus on circumscription, the results we present obviously apply to its restrictions, as for example logic databases using different types of closed-world assumptions, and similar default reasoning systems having explanation-based proof theories [Geffner, 1990; Poole, 1989].

The next section is a summary of results concerning circumscription and its theorem provers. Section 3 illustrates the problem through an example, and Section 4 provides the main results on extracting answers from a proof.

2 Background

This section gives a very brief survey on circumscription and its proof procedures. Additional background can be found in [McCarthy, 1986; Lifschitz, 1985].

2.1 Circumscription

The circumscription of a first-order theory T is its augmentation with a second-order axiom $CIRC(T; P; Z)$, where P and Z denote sets of predicate symbols of T, whose model-theoretic characterization is based on the following definition and result.

Definition 2.1 Let M_1 and M_2 be models of T. Then $M_1 \leq_{P,Z} M_2$ if M_1 and M_2 differ only in the way they interpret predicates from P and Z, and the extension of every predicate from P in M_1 is a subset of its extension in M_2. A model M of T is (P, Z)-*minimal* if for no other model M' of T it is the case that $M' \leq_{P,Z} M$.[1]

The predicates in P are said to be *minimized* and those in Z to be *variables*; Q denotes the rest of the predicates, called *parameters*.

Theorem 2.2 $CIRC(T; P; Z) \models F$ if and only if $M \models F$, for every (P, Z)-minimal model M of T.[2]

2.2 Theorem Proving Results

General circumscription is highly uncomputable [Schlipf, 1986], and existing theorem proving results apply to restrictions of circumscriptive theories. From now on, T is a first-order theory without equality, consisting of finitely many *clauses*, each of which is a disjunction of possibly negated atoms called *literals*, augmented with the Unique Names Assumptions, that is, different ground terms denote different elements of the domain. We also assume that the Domain Closure Assumption is satisfied since this is necessary to guarantee the soundness of the query answering procedure described in this paper.

[1] We will often just say minimal models, P and Z being clear from the context.

[2] $\models$ is classical first-order.

Queries to be answered are restricted to existentially quantified formulas (note that this includes ground formulas).

Theorem proving techniques for circumscription are based on the following results [Gelfond *et al.*, 1989; Przymusinski, 1989; Ginsberg, 1989].

Definition 2.3 Let T be a theory, $CIRC(T; P; Z)$ its circumscription, F a formula, and let P^+ (P^-) denote the set of positive (negative) literals whose predicate symbol belongs to P.

1. $P^- + Q$[3] is called the *explanation vocabulary.*
2. A finite conjunction E of literals from the explanation vocabulary is an *elementary explanation for* F relative to T if
   (a) $T + E \models F$, and
   (b) $T + E$ is consistent.

   A disjunction of elementary explanations is called an *explanation.*
3. Let E be an explanation. An elementary explanation for $\neg E$ is called a *counter* to E.
4. If an explanation has no counters it is *valid.*
5. A valid explanation E is *minimal* if there is no other valid explanation E' such that $E' \models E$.

Theorem 2.4 $CIRC(T; P; Z) \models F$ if and only if there exist a valid explanation for F relative to the theory T.

Example 2.5 Consider the theory

$$T = \{ \quad \forall x \; bird(x) \wedge \neg ab(x) \supset flies(x), \quad bird(tweety) \quad \},$$

where $P = \{ab\}$, $Q = \{bird\}$ and $Z = \{flies\}$, so that the explanation vocabulary is $\{ab\}^- + \{bird\}^+ + \{bird\}^-$. Let us consider the query

$$F = flies(tweety).$$

[3] We identify Q with the set of all literals whose predicate symbols are parameters.

Now, $\neg ab(tweety)$ is an explanation for F, as it implies F together with T and belongs to the explanation vocabulary. It has no counters, as no formula from the explanation vocabulary can be consistently added to T to deduce $ab(tweety)$. F is thus a theorem of $CIRC\,(T; \{ab\}; \{flies\})$.

Next, let

$$T' = T \cup \{\, ab(tweety) \vee ab(sam) \,\}.$$

Then $\neg ab(sam)$ is a counter to $\neg ab(tweety)$ relative to T'. As no other valid explanation for F exists, F is not a theorem of the circumscription of T'.

2.3 Query Answering Procedure

In line with the above results, the task of a query answering procedure for circumscription is to search for explanations of the query and test their validity.

To do so, the answering procedure can rely on an *explanation-finding algorithm*. Such an algorithm is provided with a set of clauses T, a clause F, and a vocabulary, and returns an explanation E, that is, a conjunction of literals from the vocabulary, consistent with T, such that $T + E \models F$. The computation of explanations is based on the observation that such T, F and E verify

(a) $T + \neg F \models \neg E$, and

(b) $T \not\models \neg E$.

Explanations can thus be obtained by computing the set

$$New(T, \neg F) = Th(T + \neg F) - Th(T)$$

that belong to the vocabulary $P^+ + Q$. We call these the *new theorems* of $\neg F$ relative to T.[4] The negation of each of such clauses (a conjunction of literals from the explanation vocabulary) is an elementary explanation and any disjunction of these is an explanation.

Testing the validity of an explanation represents the same computational problem: if an explanation E has no counter, then there is no new theorem

[4]Note that $New(T, \neg F)$ does not include clauses implied by T alone because their negations are inconsistent with T and they cannot be counters. The predicates of P have their sign changed because we look for the negation of E.

of E relative to T, belonging to the vocabulary $P^+ + Q$. In symbols, given an algorithm to compute the set $New(T, F)$, we are interested in

$$Explanations(T, F) = \neg New(T, \neg F)$$

and

$$Valid(E, T) \Leftrightarrow New(T, E) = \emptyset.$$

Algorithms based on ordered-linear resolution [Chang and Lee, 1973] are known to perform this computation [Przymusinski, 1989; Oxusoff and Rauzy, 1989; Siegel, 1987; Inoue, 1991], and are used in many abductive procedures [de Kleer, 1986; Poole, 1989]. The explanation-finding algorithm is not a concern of this paper. The results we present concern how to *combine* explanations in order to extract answers from a proof. We thus assume such an algorithm exist and return the correct explanations, and concentrate on the query answering procedure. The following one has been shown to correctly return yes/no answers, and is used in [Ginsberg, 1989; Przymusinski, 1989].

Algorithm 2.6 (Yes/No Answering Procedure)

Step 1. (Generate Elementary Explanations)
Compute elementary explanations of F relative to T.

Step 2. (Combine Elementary Explanations)
Set the explanation E to the disjunction of all elementary explanations, and represent it in conjunctive normal form (i.e., as a set of clauses).

Step 3. (Test Validity)
Test if E has no counter, in which case answer "Yes"; otherwise answer "No".

This query answering procedure is not an exact implementation of Theorem 2.4 in one respect. The Theorem stipulates the need for an arbitrary valid explanation, while the answering procedure, in Step 2, only tests one for validity, namely the disjunction of all the elementary ones generated in Step 1. This is enough to return yes/no answers. The reason is that if a certain disjunction of valid explanations exist, then the maximal disjunction is valid. This maximal disjunction is then tested for validity. The example we present next illustrates the inability of this procedure to provide answer-extraction.

3 Example

I have to do some Prolog and Lisp programming this morning, and I need the manuals. Asking people around, I collect information about who has recently been using them. I know the office number of my colleagues, and I also know that normally people leave books in their offices. However, there are exceptions to this rule: for example, some of my colleagues work at home and don't bring back the books to the office. The information I have can be expressed with the following theory T, where predicate symbols and constants have obvious intended interpretations:

$$\begin{gathered}
\forall x\, \forall y\, \forall z\ \mathit{had}(x,y) \wedge \mathit{office}(x,z) \wedge \neg \mathit{ab}(x) \supset \mathit{at}(y,z), \\
\mathit{had}(\mathit{fred},\mathit{prolog\text{-}manual}) \vee \mathit{had}(\mathit{mary},\mathit{prolog\text{-}manual}), \\
\mathit{had}(\mathit{harold},\mathit{prolog\text{-}manual}), \\
\mathit{had}(\mathit{kurt},\mathit{lisp\text{-}manual}), \\
\mathit{office}(\mathit{fred},\mathit{EJ225}), \\
\mathit{office}(\mathit{mary},\mathit{EJ230}), \\
\mathit{office}(\mathit{harold},\mathit{EJ235}), \\
\mathit{office}(\mathit{kurt},\mathit{EJ240}), \\
\forall x\, \forall y\, \forall z\ \mathit{different}(y,z) \supset \neg \mathit{at}(x,y) \vee \neg \mathit{at}(x,z), \\
\mathit{different}(\mathit{EJ225},\mathit{EJ230}) \wedge \cdots \\
\cdots \wedge \mathit{different}(\mathit{EJ235},\mathit{EJ240}).
\end{gathered}$$

Where should I look for the manuals? Suppose I submit to the theorem prover the query

$$F = \exists x\, \exists y\ \mathit{at}(x,y)\,.$$

I am not really interested in knowing whether F is true or not. I would like to know how to get the manuals back, and do so without inspecting all the offices around.

If we set $P = \{\mathit{ab}\}$ and let the rest of the predicates vary, the minimal models of T can be divided in three groups, in each of which exactly one of $\neg \mathit{ab}(\mathit{harold})$, $\neg \mathit{ab}(\mathit{mary})$ or $\neg \mathit{ab}(\mathit{fred})$ is true. In the first group of these

$$\mathit{at}(\mathit{prolog\text{-}manual},\mathit{EJ235})$$

holds. In the second and the third groups,

$$at(prolog\text{-}manual, EJ225) \vee at(prolog\text{-}manual, EJ230)$$

holds. Thus

$$at(prolog\text{-}manual, EJ225) \vee at(prolog\text{-}manual, EJ230) \vee at(prolog\text{-}manual, EJ235)$$

holds in all minimal models, and no subdisjunction does. Moreover, in all minimal models $\neg ab(kurt)$ is true, which means that

$$at(lisp\text{-}manual, EJ240)$$

is another theorem of the circumscription. These two smallest disjunction of answers provide me with information about where the manuals are.

However, Algorithm 2.6 produces the following.

Step 1. Three elementary explanations are computed:

$$\begin{aligned} E1 &= \neg ab(fred) \wedge \neg ab(mary) \\ E2 &= \neg ab(harold) \\ E3 &= \neg ab(kurt) \end{aligned}$$

Step 2. The disjunction $E1 \vee E2 \vee E3$ is transformed to conjunctive normal form. This is the conjunction of the following clauses:

$$\neg ab(fred) \vee \neg ab(harold) \vee \neg ab(kurt)$$
$$\neg ab(mary) \vee \neg ab(harold) \vee \neg ab(kurt)$$

Step 3. These clauses, when added to T, produce no new theorem in the vocabulary of positive ab predicates, showing that the disjunction of explanations is valid, as it has no counters. The procedure correctly answers "Yes".

There is no way instances of the query can be returned with this procedure. The reason is that the actual substitution for the variable in the query is lost in step 2, when the explanation is converted from disjunctive to conjunctive normal form. The rest of the paper describes a methodology and results on how to combine elementary explanation in a more careful way in order to produce the informative answers.

4 Answer Extraction

We consider informative answers to a query F relative to a circumscription $CIRC\,(T;P;Z)$ to be the most specific instances of F entailed by the circumscription; in the sequel we simply call them *answers* to the query.

Definition 4.1 Let $CIRC(T;P;Z)$ be a circumscriptive theory, F an existentially quantified query. An *answer* to F is a formula A such that

1. $CIRC(T;P;Z) \models A$,
2. $A \models F$, and
3. No A' different from A satisfies (1), (2) and $A' \models A$.

We now show how to produce such answers.

4.1 Obtaining Instances of the Query

As we said before, we assume that an explanation-finding algorithm returns explanations for the query. The problem we address is that of finding answers, that is, most specific disjunction of instances of the query entailed by the circumscription. To compute such instances of the query, we use Green's techniques for first-order logic [Green, 1969a; Green, 1969b], that consist of associating with F the clause

$$F' = \neg F \vee Ans(\mathbf{x})$$

where $\mathbf{x} = x_1, \ldots, x_n$ stands for the variables appearing in F. During the proof of F, Ans keeps track of the substitutions for which F holds at no extra cost. The explanation-finding algorithm is thus the same resolution-based algorithm used by existing theorem provers for circumscription [Ginsberg, 1989; Przymusinski, 1989], but instead of supplying the negation of the query F to compute $\neg New(T, \neg F)$, we supply F' and compute $New(T, F')$.[5] Accordingly, instead of obtaining explanations

$$E_1\,,\ \ldots\ ,\ E_n$$

[5]The answer predicate Ans is added to the vocabulary $P^+ + Q$.

for F, we obtain

$$E_1 \supset Ans_1\,,\ \ldots\ ,\ E_n \supset Ans_n\,,$$

where each Ans_i keeps the substitutions for a disjunction F_i of instances of F explained by E_i, that is,

$$T + E_i \models F_i\ (i = 1, \ldots, n).$$

4.2 Computing Answers

Given a query, the explanation-finding algorithm can compute the candidate answers $F_1, F_2, \ldots$, the explanations for each of these $E_{11}, E_{12}, \ldots$, and their counters $C_{111}, C_{112}, \ldots$. An answer is a shortest disjunction of F_i that has a valid explanation, that is, for which at least one explanation exists with no counter.

Suppose two instances of the query F_1 and F_2 have elementary explanations E_1 and E_2, neither of which is valid. E_1 and E_2 thus have counters C_1 and C_2. Now $C = C_1 \wedge C_2$ is a candidate counter for $E = E_1 \vee E_2$ in the sense that as $T + C_i \models \neg E_i$, obviously $T + C_1 \wedge C_2 \models \neg E$. But still C might not be a counter to E because, although each C_i is consistent with T, C might not be. If C is not consistent with T, and no other counter exist, E will be a valid explanation for $F_1 \vee F_2$.

Thus while certain explanations may have counters, their disjunction might not, as the corresponding conjunction of potential counters is inconsistent. Thus, compared with theorem provers that return yes/no answers, *the only additional computation needed is the consistency check on combinations of counters.*

So the search problem consists of finding the disjunctions of instances of the query such that the potential counters of their explanations are inconsistent. A counter C is inconsistent with T if and only if $T \models \neg C$; and such counters belong to a particular vocabulary. It is thus rewarding to compute the following set of clauses.

Definition 4.2 Let $CIRC(T; P; Z)$ be a circumscriptive theory. A *characteristic clause* of $CIRC(T; P; Z)$ is a clause C that satisfies the following.

1. Every literal of C belongs to $P^+ + Q$.
2. $T \models C$.

3. No other clause C' satisfies (1), (2) and $C' \models C$.

These are the restriction of the *prime implicates* [Reiter and de Kleer, 1987] of T to a particular vocabulary. They can be computed with the same linear resolution algorithm used by the explanation-finding procedure, as shown by [Inoue, 1991].

We can take advantage of this set in at least the following ways.

1. Let E be an elementary explanation, and C a counter to E. Then $T + C \models \neg E$, and thus $T \models \neg C \vee \neg E$. As the explanation-finding algorithm returns the shortest of such clauses, *$\neg C \vee \neg E$ is a characteristic clause of the circumscription.* In other words, to compute counters to an explanation, we consider the negation of its elementary components. The counters are the negation of the complement within a characteristic clause.

2. If $T + C$ is inconsistent, and thus $T \models \neg C$, then *$\neg C$ is implied by a characteristic clause of the circumscription.* This means that the consistency test on a combination of counters can be performed by entailment tests on the characteristic clauses.

The above ideas can be implemented using different search strategies, which are not our concern. The following is a possible implementation of an algorithm[6] to return informative answers to a query in a circumscriptive theory.

Algorithm 4.3 (Query Answering Procedure)

Step 1. (Compilation)
Compute the characteristic clauses of the circumscription.

Step 2. (Generate Elementary Explanations)
Compute elementary explanations of F relative to T.

Step 3. (Compute Counters)
A counter to an explanation is the complement of its elementary components within the characteristic clauses. If an explanation has no counters, output the instance of the query explained by it.

[6]Strictly speaking, this is not an "algorithm". The reason is, in Step 1 or 2, it may produce an infinite number of characteristic clauses or elementary explanations.

Step 4. (Combinations of Counters)
Compute the conjunctions of counters whose negation is entailed by a characteristic clause. Such a conjunction of counters is inconsistent with T, and the corresponding disjunction of elementary explanations is valid. If such an explanation has no other counters, the corresponding disjunction of instances of the query is an answer.

4.3 Example

We consider again the example of section 3.

Step 1. Since ab is the only minimized predicate, the only characteristic clause of the circumscription is:

$$ab(harold) \vee ab(fred) \vee ab(mary).$$

Step 2. We provide the explanation-finding algorithm with the clause

$$\neg at(x, y) \vee Ans(x, y),$$

and the answer predicate Ans.
Then we obtain the new clause

$$\begin{aligned} & ab(fred) \vee ab(mary) \\ & \vee Ans(\textit{prolog-manual}, EJ225) \\ & \vee Ans(\textit{prolog-manual}, EJ230), \end{aligned}$$

indicating that

$$E_1 = \neg ab(fred) \wedge \neg ab(mary)$$

explains

$$\begin{aligned} A_1 \quad = \quad & at(\textit{prolog-manual}, EJ225) \\ & \vee at(\textit{prolog-manual}, EJ230). \end{aligned}$$

In a similar way,

$$E_2 = \neg ab(harold)$$

explains

$$A_2 = at(\textit{prolog-manual}, EJ235),$$

and

$$E_3 = \neg ab(kurt)$$

explains

$$A_3 = at(lisp\text{-}manual, EJ240).$$

Step 3. $C_1 = \neg ab(harold)$, is a counter to E_1.
$C_2 = \neg ab(fred) \wedge \neg ab(mary)$, is a counter to E_2.
E_3 has no counters, thus A_3 is output.

Step 4. The only characteristic clause subsumes (in fact, is equivalent to) the disjunction of the negation of the two counters C_1 and C_2. This indicates that $T + C_1 \wedge C_2$ is inconsistent, and as $E_1 \vee E_2$ has no other counter, it is a valid explanation for $A_1 \vee A_2$.

5 Conclusion

Nonmonotonic theorem provers often consist in a two-step classical deduction — making a default proof in which explanations are collected and checking validity of the explanations. We showed that the substitutions needed for query answering are lost in this process, and a combination of theses need to be found to produce the required answers.

We presented a procedure for combining explanations in order to obtain informative answers, and results that enable an answering procedure to return the interesting answers with minimal search.

The importance of the results presented lies in their applicability to a wide class of systems that are either a restriction of circumscription, for example, databases using different types of closed-world assumptions (see [Przymusinski, 1989; Gelfond *et al.*, 1989]), or similar default reasoning systems having explanation-based proof theories [Poole, 1989; Geffner, 1990].

Acknowledgment

We are grateful to Martin Abadi and Vladimir Lifschitz for their comments on earlier drafts.

References

[Chang and Lee, 1973] Chin-Liang Chang and Richard Char-Tung Lee. *Symbolic Logic and Mechanical Theorem Proving.* Academic Press, New York, 1973.

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

[Etherington, 1988] David W. Etherington. *Reasoning with Incomplete Information.* Pitman, London, 1988.

[Geffner, 1990] Hector Geffner. Causal theories for nonmonotonic reasoning. In *Proceedings of the 8th AAAI*, pages 524–530, Boston, MA, 1990.

[Gelfond *et al.*, 1989] Michael Gelfond, Halina Przymusinska, and Teodor Przymusinski. On the relationship between circumscription and negation as failure, *Artificial Intelligence*, 38: 75–94, 1989.

[Ginsberg, 1989] Matthew L. Ginsberg. A circumscriptive theorem prover. *Artificial Intelligence*, 39: 209–230, 1989.

[Green, 1969a] Cordell Green. Theorem-proving by resolution as a basis for question-answering systems. In: B. Meltzer and D. Michie, editors, *Machine Intelligence 4*, pages 183–205, Edinburgh University Press, Edinburgh, 1969.

[Green, 1969b] Cordell Green. Application of theorem proving to problem solving. In *Proceedings of the IJCAI*, pages 219–239, Washington D.C., 1969.

[Inoue and Helft, 1990] Katsumi Inoue and Nicolas Helft. On theorem provers for circumscription. In *Proceedings of the 8th Biennial Conference of the Canadian Society for Computational Studies of Intelligence*, pages 212–219, Ottawa, Ontario, 1990.

[Inoue, 1991] Katsumi Inoue. Consequence-finding based on ordered linear resolution. In *Proceedings of the 12th IJCAI*, Sydney, Australia, 1991.

[Lifschitz, 1985] Vladimir Lifschitz. Computing circumscription. In *Proceedings of the 9th IJCAI*, pages 121–127, Los Angeles, CA, 1985.

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

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

[Oxusoff and Rauzy, 1989] Laurent Oxusoff and Antoine Rauzy. L'evaluation sémantique en calcul propositionnel. Thèse de Doctorat, Université d'Aix-Marseille II, Luminy, France, 1989.

[Poole, 1989] David Poole. Explanation and prediction: an architecture for default and abductive reasoning. *Computational Intelligence*, 5: 97–110, 1989.

[Przymusinski, 1989] Teodor C. Przymusinski. An algorithm to compute circumscription. *Artificial Intelligence*, 38: 49–73, 1989.

[Reiter and de Kleer, 1987] Raymond Reiter and Johan de Kleer. Foundations of assumption-based truth maintenance systems: preliminary report. In *Proceedings of the 6th AAAI*, pages 183–187, Seattle, WA, 1987.

[Schlipf, 1986] John S. Schlipf. How uncomputable is general circumscription? In *Proceeings of the 1st IEEE Symposium on Logic in Computer Science*, pages 92–95, Cambridge, MA, 1986.

[Siegel, 1987] Pierre Siegel. Représentation et utilisation de la connaissance en calcul propositionnel. Thèse d'État. Université d'Aix-Marseille II, Luminy, France, 1987.