ICOT Technical Memorandum: TM-0877

TM-0877

Saturation and Abdution for an Extended ATMS

by
K. Inoue

April, 1990

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

Saturation and Abduction for an Extended ATMS

Katsumi Inoue
ICOT Research Center
1-4-28 Mita, Minato-ku, Tokyo 108, Japan
inoue@icot.jp

Abstract

This paper concerns deductive procedures for abductive reasoning and a variety of ATMSs. Reiter & de Kleer [21] view an ATMS as a kind of abduction in which the best explanation of a formula is defined as a minimal conjunction of hypotheses that explain the formula. However, they do not give any algorithm to compute such minimal explanations of a formula in their CMS that is a generalization of de Kleer's basic ATMS [3]. In this paper, we use the notion of characteristic clauses [24] to explain clearly the computational aspects of the CMS and the ATMS and to produce an efficient abductive procedure based on linear resolution. By means of this abductive procedure, we give the CMS algorithms for computing minimal explanations in the interpreted approach and for updating them in the compiled approach. We then present algorithms for generating and updating labels of nodes in an extended ATMS that accepts any formula justifications and literal assumptions.

Keywords: ATMS, CMS, Abduction, Saturation, Linear Resolution

1 Introduction

An assumption-based truth maintenance system (ATMS) [3] has been widely used when problems require reasoning in multiple contexts. Recent investigations such as those of Reiter & de Kleer [21] and Levesque [15] show that there are strong connections between the ATMS and a logical account of *abduction* or *hypothesis generation* [19, 8, 18].

Definition 1.1 Let W be a set of formulas, A a set of ground literals (called the *assumptions*), and G a closed formula (called *observation*). A formula H is an *explanation of G with respect to W and A* if:

(1) $W \cup \{H\} \models G$,
(2) $W \cup \{H\}$ is satisfiable, and
(3) H is a conjunction of literals in A.

An explanation H of G with respect to W and A is *minimal* if:

(4) No proper sub-conjunct of H is an explanation of G.

From the viewpoint of abductive reasoning, condition (3) says that an explanation H is limited to consisting of a conjunction of literals that are expressed in terms of a prespecified subset of the predicate symbols, assumable literals. Pople [19] does not allow for such a distinguished set of literals. Poole [18] and Finger [8] do not require the minimality condition (4). Here we claim that minimal explanations can reasonably be accepted because we do not need unnecessary hypotheses (the maxim of Occam's Razor [19]). Moreover, the minimality is essential for efficiency in some application domains such as diagnosis and design.

The ATMS is precisely intended to generate *all and only* minimal explanations. In the ATMS terminology, the set of explanations of an observation G (called *node*) with respect to the sets W (called *justifications*) and A that satisfy the above four conditions is called the *label* of G, which is *consistent, sound, complete* and *minimal*. The *basic* ATMS [3] is restricted to accepting only Horn clause justifications and atomic assumptions, and each observation to be explained is only an atom. In the above four conditions for an ATMS, justifications and observations can contain any formulas, and assumptions are allowed to be literals. We call this generalization an *extended* ATMS, because it covers de Kleer's various extended versions of the ATMS [4, 5, 6], Dressler's extended ATMS [7], and Reiter & de Kleer's clause management system (CMS) [21] if every ground literal is regarded as an assumption. Although the CMS is well defined, [21] does not give any algorithm for computing such minimal explanations of a formula in it.

In the remaining sections, we describe abduction as the problem of finding the *characteristic clauses* [1, 24] that are theorems of a given set of clauses and that belong to a distinguished sub-vocabulary of the language. We will give a propositional linear resolution procedure with a *production field* for abduction and saturation, then show ways in which to implement the CMS and the extended ATMS described above for both label generating (the interpreted approach) and label updating (the compiled approach). Since our extended ATMS can accept literal assumptions and general formulas, the methods described in this paper can also be applied to better implementations of theorem provers for closed world assumptions [1] and circumscription [20, 9], based on abductive procedures [11, 10]. Unless otherwise specified, proofs for theorems and propositions are given in [12].

2 Background

We begin with some definitions and notations that will be used throughout this paper. We shall assume a propositional language with finitely many propositional symbols $\mathcal{A}$ and with logical connectives. The set of *literals* is defined as: $\mathcal{A}^{\pm} = \mathcal{A} \cup \neg \cdot \mathcal{A}$, where $\neg \cdot S$ means the set formed by taking the negation of each element in S. A *clause* is a finite set of literals, understood disjunctively; the empty clause is denoted by $\Box$. A *conjunctive normal form (CNF) formula* is a conjunction of clauses. Let C and C' be two clauses. $C - C'$ denotes a clause whose literals are those in the difference of C and C'. C is said to *subsume* C' if every literal in C occurs in C' ($C \subseteq C'$). Let Σ be a set of clauses. By $\mu\Sigma$ or $\mu\,[\Sigma]$ we mean the set of clauses of Σ not subsumed by any other clause of Σ. A clause C is an *implicate* of Σ if $\Sigma \models C$. The set of implicates of Σ is denoted by $Th(\Sigma)$. The *prime implicates* of Σ are: $PI(\Sigma) = \mu\, Th(\Sigma)$.

2.1 Characteristic Clauses

We use the notion of *characteristic clauses*, which helps to analyze the computational aspect of ATMSs. While the idea of characteristic clauses was introduced by Bossu & Siegel [1] to evaluate a form of closed-world reasoning and was later generalized by Siegel [24], neither research focused on abductive reasoning or the ATMS. Informally speaking, characteristic clauses are intended to represent "interesting" clauses to solve a certain problem, and are constructed over a sub-vocabulary of the representation language called a *production field*.

Definition 2.1 (1) A *production field* $\mathcal{P}$ is a pair, $\langle L_{\mathcal{P}}, Cond \rangle$, where $L_{\mathcal{P}}$ (called the *characteristic literals*) is a subset of $\mathcal{A}^{\pm}$, and $Cond$ is some conditions to be satisfied. When $Cond$ is not specified, $\mathcal{P}$ is just denoted as $\langle L_{\mathcal{P}} \rangle$. A production field $\langle \mathcal{A}^{\pm} \rangle$ is denoted $\mathcal{P}_{\pi}$.

(2) A clause C *belongs to a production field* $\mathcal{P} = \langle L_{\mathcal{P}}, Cond \rangle$ if every literal in C belongs to $L_{\mathcal{P}}$ and C satisfies $Cond$. The set of implicates of Σ belonging to $\mathcal{P}$ is denoted by $Th_{\mathcal{P}}(\Sigma)$.

(3) A production field $\mathcal{P}$ is *stable* if $\mathcal{P}$ satisfies the condition: for two clauses C and C' where C subsumes C', if C' belongs to $\mathcal{P}$, then C also belongs to $\mathcal{P}$.

Example 2.2 The following are examples of implicates belonging to stable production fields.

(1) $\mathcal{P} = \mathcal{P}_{\pi}$: $Th_{\mathcal{P}}(\Sigma)$ is equivalent to $Th(\Sigma)$.

(2) $\mathcal{P} = \langle \mathcal{A} \rangle$: $Th_{\mathcal{P}}(\Sigma)$ is the set of positive clauses implied by Σ.

(3) $\mathcal{P} = \langle \neg \cdot A,$ *below size* $k \rangle$ where $A \subseteq \mathcal{A}$: $Th_{\mathcal{P}}(\Sigma)$ is the set of negative clauses implied by Σ containing less than k literals all of which belong to $\neg \cdot A$.

Definition 2.3 Let Σ be a set of clauses.

(1) The *characteristic clauses of* Σ *with respect to* $\mathcal{P}$ are:

$$Carc(\Sigma, \mathcal{P}) = \mu\, Th_{\mathcal{P}}(\Sigma)\,.$$

In other words, a characteristic clause of Σ is a prime implicate of Σ belonging to $\mathcal{P}$.

(2) Let F be a formula. The *new characteristic clauses of* F *with respect to* Σ *and* $\mathcal{P}$ are:

$$Newcarc(\Sigma, F, \mathcal{P}) = Carc(\Sigma \cup \{F\}, \mathcal{P}) - Carc(\Sigma, \mathcal{P})\,,$$

that is, those characteristic clauses of $\Sigma \cup \{F\}$ that are not characteristic clauses of Σ.

$Carc(\Sigma, \mathcal{P})$ represents *saturation*: all the unsubsumed implicates of Σ that belong to a production field $\mathcal{P}$ must be contained in it. For example, $Carc(\Sigma, \mathcal{P}_*) = PI(\Sigma)$. Note that if Σ is unsatisfiable, then $Carc(\Sigma, \mathcal{P})$ only contains $\square$. On the contrary, the next theorem shows that $Newcarc(\Sigma, \mathcal{P}, F)$ represents *abduction*.

Theorem 2.4 Let Σ be a set of clauses, $A \subseteq \mathcal{A}^{\pm}$, G a formula. The set of all minimal explanations of G with respect to Σ and A is $\neg \cdot Newcarc(\Sigma, \neg G, \mathcal{P})$, where $\mathcal{P} = \langle \neg \cdot A \rangle$.

2.2 About Abductive Procedures

In this section, we show that given a set of clauses Σ, a stable production field $\mathcal{P}$ and a formula F, the characteristic clauses $Carc(\Sigma, \mathcal{P})$ and the new characteristic clauses $Newcarc(\Sigma, F, \mathcal{P})$ can be computed by using resolution. Before describing this matter in detail, it is worth noting that, since we are dealing with abduction, the proof procedure has the following difficulties:

1. It should be complete for *consequence-finding*, that is, every relevant theorem can be produced, instead of just *refutation-complete* (producing $\square$ if the theory is unsatisfiable).
2. It should focus on producing only those theorems that belong to $\mathcal{P}$.
3. It should be able to check produced clauses from $\Sigma \cup \{F\}$ and $\mathcal{P}$ with the condition "not belonging to $Th_{\mathcal{P}}(\Sigma)$", which corresponds to consistency checking in abduction.

The completeness for consequence-finding was investigated by Slagle, Chang & Lee [25] and Minicozzi & Reiter [17]. The second property requires that such consequences belong to $\mathcal{P}$. A promising approach to overcome these difficulties is to use an *incremental* resolution procedure, which should first deduce all the $Carc(\Sigma, \mathcal{P})$ prior to giving $Carc(\Sigma \cup \{F\}, \mathcal{P})$. Bossu & Siegel's [1] saturation procedure is an example of incremental resolution methods.

A better approach to compute $Newcarc(\Sigma, C, \mathcal{P})$ does not construct the whole of each saturated set. It is possible by using a linear resolution procedure, given Σ, $\mathcal{P}$, and a newly added single clause C as the *top clause* of a deduction. Siegel [24] proposes such a resolution method by extending SL-resolution [14]. In this paper, we use the basic idea of [24] but introduce a more simplified procedure which is enough to explain our goals. The resolution method, which we call *m.c.l.s. resolution*, is based on *m.c.l.* (merge, C-ordered, linear) *resolution* [17] [1], and is augmented by the *skipping* operation [2]. The following procedure is based on the description of OL-deduction in [2], but the result is not restricted to it. An *ordered* clause is a sequence of literals possibly containing *framed literals* which represents literals that have been resolved upon: from a clause C an ordered clause $\vec{C}$ is obtained just by ordering the elements of C; conversely, from an ordered clause $\vec{C}$ a clause C is obtained by removing the framed literals and converting the remainder to the set. A *structured* clause $\langle P, \vec{Q} \rangle$ is a pair of a clause P and an ordered clause $\vec{Q}$, whose clausal meaning is $P \cup Q$.

[1] By the term m.c.l. resolution, we mean the family of linear resolution using ordered clauses and the information of literals resolved upon. Examples of m.c.l. resolution are OL-resolution [2], SL-resolution [14], the model elimination procedure [16], and the graph construction procedure [23].

[2] Roughly speaking, the skipping operation corresponds to Pople's [19] *synthesis* operation.

Definition 2.5 Given a set of clauses Σ, a clause C, and a production field $\mathcal{P} = \langle L_{\mathcal{P}}, Cond \rangle$, an *m.c.l.s. deduction of a clause S from $\Sigma + C$ and $\mathcal{P}$* consists of a sequence of structured clauses $D_0, D_1, \ldots, D_n$, such that:

1. $D_0 = \langle \Box, \vec{C} \rangle$.
2. $D_n = \langle S, \Box \rangle$.
3. For each $D_i = \langle P_i, \vec{Q}_i \rangle$, $P_i \cup Q_i$ is not a tautology.
4. For each $D_i = \langle P_i, \vec{Q}_i \rangle$, $P_i \cup Q_i$ is not subsumed by any $P_j \cup Q_j$, where $D_j = \langle P_j, \vec{Q}_j \rangle$ is a previous structured clause, $j < i$.
5. $D_{i+1} = \langle P_{i+1}, \vec{Q_{i+1}} \rangle$ is generated from $D_i = \langle P_i, \vec{Q}_i \rangle$ according to the following steps:
   (a) Let l be the first literal of $\vec{Q}_i$. P_{i+1} and $\vec{R_{i+1}}$ are obtained by applying either of the rules:
      i. **(Skip)** If $l \in L_{\mathcal{P}}$ and $P_i \cup \{l\}$ satisfies $Cond$, then $P_{i+1} = P_i \cup \{l\}$ and $\vec{R_{i+1}}$ is the ordered clause obtained by removing l from $\vec{Q}_i$.
      ii. **(Resolve)** $P_{i+1} = P_i$ and $\vec{R_{i+1}}$ is an ordered resolvent of $\vec{Q}_i$ with a clause B_i in Σ, where the literal resolved upon in $\vec{Q}_i$ is l.
   (b) $\vec{Q_{i+1}}$ is the reduced ordered clause of the ordered factor of $\vec{R_{i+1}}$.

Remarks. (1) Rules 1, 3, 5(a)ii and 5b form an OL-deduction for the non-production part (the right side) of structured clauses. By the *ordered factor* of $\vec{R}_i$, it implies the ordered clause obtained by merging right for any identical literals in $\vec{R}_i$ and by deleting every framed literal not followed by an unframed literal in the remainder (truncation). The *reduction* (or ancestry) of $\vec{R}_i$ deletes any unframed literal k in $\vec{R}_i$ for which there exists a framed literal $\boxed{\neg k}$ in $\vec{R}_i$.
(2) Rule 4 is included for efficiency. It does not affect the completeness described below [3].
(3) Rules 5(a)i and 5(a)ii are not exclusive; for $l \in L_{\mathcal{P}}$ either rule can be applied.

The **Skip** rule (5(a)i) reflects the following operational interpretation of a *stable* production field $\mathcal{P}$: by Definition 2.1 (3), if a clause C does not belong to $\mathcal{P}$ and a clause C' is subsumed by C, then C' does not belong to $\mathcal{P}$ either. Thus we can prune a deduction sequence if no rule can be applied for a structured clause D_i; if **Skip** was applied nevertheless, any resultant sequence would not succeed, thus making unnecessary computation.

Theorem 2.6 (1) *Soundness*: If a clause S is derived using an m.c.l.s. deduction from $\Sigma + C$ and $\mathcal{P}$, then S belongs to $Th_{\mathcal{P}}(\Sigma \cup \{C\})$.
(2) *Completeness*: If a clause T *does not belong to* $Th_{\mathcal{P}}(\Sigma)$, *but* belongs to $Th_{\mathcal{P}}(\Sigma \cup \{C\})$, then there is an m.c.l.s. deduction of a clause S from $\Sigma + C$ and $\mathcal{P}$ such that S subsumes T.

[3] In fact, in Chang & Lee's version of OL-deduction [2] this rule is overlooked. The deletion rule is clearly present in the model elimination procedure [16]. These two observations were pointed out by Mark Stickel.

The proof for the completeness in Theorem 2.6 can be seen as an extension of the result for linear resolution by Minicozzi & Reiter [17]. Note that m.c.l. resolution is refutation-complete [16, 14, 2], but is incomplete for consequence-finding [17]. The procedure of m.c.l.s. resolution is complete for characteristic-clause-finding, because it includes the additional skipping operation.

Example 2.7 Suppose that $\Sigma = \{ a \vee b,\ \neg c \vee \neg b,\ \neg c \vee \neg a \}$. There is no m.c.l. deduction of $\neg c$ from Σ, but $\neg c$ is derived using an m.c.l.s. deduction from Σ and $\mathcal{P}_\pi$ as:
$\langle \Box,\ \underline{a} \vee b \rangle,\ \langle \Box,\ \underline{\neg c} \vee \boxed{a} \vee b \rangle,\ \langle \neg c,\ \boxed{\not a} \vee \underline{b} \rangle,\ \langle \neg c,\ \underline{\neg c} \vee \boxed{b} \rangle,\ \langle \neg c,\ \boxed{\not b} \rangle$.

Definition 2.8 Given a set of clauses Σ, a clause C, and a stable production field $\mathcal{P}$, the *production from* $\Sigma + C$ *and* $\mathcal{P}$ is defined as:
$Prod(\Sigma, C, \mathcal{P}) = \mu \{ S \mid S$ is a clause derived using an m.c.l.s. deduction from $\Sigma + C$ and $\mathcal{P} \}$.

In [24], there is no precise statement about computing $Newcarc(\Sigma, C, \mathcal{P})$ and $Carc(\Sigma, \mathcal{P})$ by using $Prod(\Sigma, C, \mathcal{P})$. Here we show the connections between them. Firstly, the next theorem shows that we can compute $Newcarc(\Sigma, C, \mathcal{P})$ for a single clause C, without a naive implementation of Definition 2.3 (2) that computes the saturated sets, $Carc(\Sigma, \mathcal{P})$ and $Carc(\Sigma \cup \{C\}, \mathcal{P})$, and that we need check for each clause $S \in Prod(\Sigma, C, \mathcal{P})$, only whether $\Sigma \models S$ or not.

Theorem 2.9 Let C be a clause. $Newcarc(\Sigma, C, \mathcal{P}) = Prod(\Sigma, C, \mathcal{P}) - Th_{\mathcal{P}}(\Sigma)$.

For a CNF formula G, $Newcarc(\Sigma, G, \mathcal{P})$ can be computed incrementally as follows:

Theorem 2.10 Let $G = C_1 \wedge \cdots \wedge C_m$ be a CNF formula. Then

$$\begin{aligned} Newcarc(\Sigma, G, \mathcal{P}) &= \mu [\bigcup_{i=1}^{m} Newcarc(\Sigma_i, C_i, \mathcal{P})] \\ &= \mu [\bigcup_{i=1}^{m} Prod(\Sigma_i, C_i, \mathcal{P})] - Th_{\mathcal{P}}(\Sigma), \\ \text{where } \Sigma_1 = \Sigma, &\text{ and } \Sigma_{i+1} = \Sigma_i \cup \{C_i\}, \text{ for } i = 1, \ldots, m-1. \end{aligned}$$

Finally, the characteristic clauses $Carc(\Sigma, \mathcal{P})$ can be generated by the following incremental method. This will be used for the compiled approaches to the CMS and an ATMS in sections 3.1 and 4.2. Notice that for some propositional symbol p, if $\Sigma \not\models p$, $\Sigma \not\models \neg p$, and $p \vee \neg p$ belongs to $\mathcal{P}$, then $p \vee \neg p$ belongs to $Carc(\Sigma, \mathcal{P})$.

Theorem 2.11 The characteristic clauses with respect to $\mathcal{P}$ can be generated incrementally [4]:

$$\begin{aligned} Carc(\phi, \mathcal{P}) &= \{ p \vee \neg p \mid p \in \mathcal{A} \text{ and } p \vee \neg p \text{ belongs to } \mathcal{P} \}, \text{ and} \\ Carc(\Sigma \cup \{C\}, \mathcal{P}) &= \mu [Carc(\Sigma, \mathcal{P}) \cup Prod(\Sigma, C, \mathcal{P})]. \end{aligned}$$

[4] In practice, no tautology will take part in any deduction; tautologies decrease monotonically.

3 The Clause Management System

Reiter & de Kleer [21] propose a generalization of the basic ATMS [3] called the *clause management system* (CMS) and show its applications to abductive reasoning. A CMS is intended to work together with a reasoner, which issues queries that take the form of clauses. The CMS is then responsible for finding *minimal supports* for the queries:

Definition 3.1 [21] Let Σ be a set of clauses and C a clause. A clause S is a *support for C with respect to* Σ if: $\Sigma \models S \cup C$, and $\Sigma \not\models S$.
A support for C with respect to Σ is *minimal* if there is no other support S' for C which subsumes S. The set of minimal supports for C with respect to Σ is written $MS(\Sigma, C)$.

Comparing minimal supports with minimal explanations described in Definition 1.1, a minimal support S for C with respect to Σ is exactly a minimal explanation $\neg S$ of C with respect to Σ and $\mathcal{A}^{\pm}$. Therefore, the above definition can be easily extended to handle any formula instead of a clause as a query. Setting the production field to $\mathcal{P}_\pi = \langle\, \mathcal{A}^{\pm} \,\rangle$, we see that:

Proposition 3.2 Let F be any formula. $MS(\Sigma, F) = Newcarc(\Sigma, \neg F, \mathcal{P}_\pi)$.

This formulation can solve one of the limitations of the CMS. In [21], the CMS is defined to handle only the observations of the clause form, so that it cannot compute minimal explanations of a conjunctive observation. For example, $\mu\{\, \neg e \mid \Sigma \models e \supset g_1 \wedge g_2 \text{ and } \Sigma \not\models \neg e\}$ can be computed straightforwardly in our formulation as $Newcarc(\Sigma, \neg g_1 \vee \neg g_2, \mathcal{P}_\pi)$. And for a disjunctive normal form observation F, we can compute $MS(\Sigma, \neg F)$ by using Theorem 2.10.

We thus see that our algorithm can compute minimal supports. However, Reiter & de Kleer [21] consider the two ways the CMS manages the knowledge base: keeping the set of clauses Σ transmitted by the reasoner as it is (the *interpreted* approach), or computing $PI(\Sigma)$ (the *compiled* approach). Theorem 2.9 shows that we can generate the new characteristic clauses $Newcarc(\Sigma, C, \mathcal{P}_\pi)$ without knowing the saturated sets, $PI(\Sigma)$ and $PI(\Sigma \cup \{C\})$. Therefore, computation using Theorem 2.9 and Proposition 3.2 represents the interpreted approach [5].

3.1 Compiling the Knowledge Base

When we are faced with a situation in abduction where we want to know explanations for many different queries, we must run the algorithm each time a query is issued. Instead of keeping the initial theory Σ as it is and doing the same deductions over and over for different top clauses, some of these inferences can be made once and for all. That is the motivation for the compiled approach: the set Σ is compiled into the saturated set, $PI(\Sigma) = Carc(\Sigma, \mathcal{P}_\pi)$.

Given $PI(\Sigma)$, to find $MS(\Sigma, G)$ for each query G in the compiled approach, again we do not need to compute the saturated set $PI(\Sigma \cup \{\neg G\})$, as Reiter & de Kleer show some relationships between prime implicates and minimal supports.

Proposition 3.3 [21] Let C be a clause. $MS(\Sigma, C) = \mu\{\, P - C \mid P \in PI(\Sigma) \text{ and } P \cap C \neq \phi\}$.

[5] Note that in [21] there is no description of an algorithm for the interpreted approach.

Corollary 3.4 [21] Let $n \in \mathcal{A}^{\pm}$ be a literal. $MS(\Sigma, \{n\}) = \{ P - \{n\} \mid P \in PI(\Sigma) \text{ and } n \in P \}$.

One of the disadvantages of the compiled approach is the high cost of updating the knowledge base. When the reasoner adds a clause C to Σ, we must compute all the $PI(\Sigma \cup \{C\})$. However, for both purposes, that is, constructing the prime implicates and updating them, Theorem 2.11 can be used by setting the production field to $\mathcal{P}_\pi$.

Proposition 3.5 Given $PI(\Sigma)$ and a clause C, $PI(\Sigma \cup \{C\})$ can be found incrementally:

$$\begin{aligned} PI(\phi) &= \{ p \vee \neg p \mid p \in \mathcal{A} \}, \text{ and} \\ PI(\Sigma \cup \{C\}) &= \mu [PI(\Sigma) \cup Prod(PI(\Sigma), C, \mathcal{P}_\pi)]. \end{aligned}$$

By Proposition 3.5, the prime implicates can be incrementally constructed using every clause as a top clause. Thus the transmitted clauses Σ can be substituted for $PI(\Sigma)$. When a clause C is newly added, we just need to add the theorems deduced from $PI(\Sigma)$ with top clause C and to remove the subsumed clauses. The computation of all prime implicates of Σ by Proposition 3.5 is much more efficient than the brute-force way of resolution proposed briefly by Reiter & de Kleer, which makes every possible resolution until no more unsubsumed clauses are produced [6].

4 An Extended ATMS

In de Kleer's versions of ATMSs [3, 4, 5, 6], there is a distinguished set of assumptions $A \subseteq \mathcal{A}^{\pm}$. One of the most generalized versions of the ATMS can be considered as a CMS with assumptions as described in Definition 1.1. Therefore, based on Theorem 2.4, an ATMS can be defined as a responsible system for finding all the minimal explanations (called the label) for the queries:

Definition 4.1 An *ATMS* is a triple $\langle N, A, \Sigma \rangle$, where $N \subseteq \mathcal{A}^{\pm}$ is a set of literals, *nodes*, $A \subseteq N$ is a set of literals, *assumptions*, and Σ is a set of clauses all of whose literals belong to $N \cup \neg \cdot N$, *justifications*. The *label of* $n \in N$ *with respect to* $\langle N, A, \Sigma \rangle$ is defined as:

$$L(n, A, \Sigma) = \neg \cdot Newcarc(\Sigma, \neg n, \mathcal{P}), \text{ where } \mathcal{P} = \langle \neg \cdot A \rangle.$$

The following properties [3, 5] hold for the label of each node $n \in N$ with respect to an ATMS $\langle N, A, \Sigma \rangle$ given by Definition 4.1:

Proposition 4.2 Let $\langle N, A, \Sigma \rangle$ be an ATMS, $n \in N$ a literal, $\mathcal{P} = \langle \neg \cdot A \rangle$.

(1) *Label consistency*: for each $E_i \in L(n, A, \Sigma)$, $\Sigma \cup \{E_i\}$ is satisfiable.

(2) *Label soundness*: for each $E_i \in L(n, A, \Sigma)$, $\Sigma \cup \{E_i\} \models n$.

(3) *Label completeness*: for every conjunct E of assumptions in A, if $\Sigma \cup \{E\} \models n$, then there exists $E_i \in L(n, A, \Sigma)$ such that E_i is a sub-conjunct of E.

(4) *Label minimality*: every $E_i \in L(n, A, \Sigma)$ is not a super-conjunct of any other element.

[6] Reiter & de Kleer [21] also briefly alluded to more disciplined ways for computing prime implicates and announced that they would be considered in the full paper, which has not been published yet.

In the same way as the CMS, we will consider the following two problems, that is, abduction and saturation, concerning the computation of the labels of the nodes with respect to an ATMS:

1. *Generating labels.* Given an ATMS $\langle N, A, \Sigma \rangle$, compute $L(n, A, \Sigma)$ for some node $n \in N$ from the original set Σ. This corresponds to the interpreted approach of the CMS.

2. *Updating labels.* Given an ATMS $\langle N, A, \Sigma \rangle$ and the current label $L(n, A, \Sigma)$ of each $n \in N$, compute the new label $L(n, A, \Sigma \cup \{C\})$ of every $n \in N$ with respect to $\langle N, A, \Sigma \cup \{C\} \rangle$. This corresponds to the compiled approach of the CMS.

4.1 Label Generation

Generating the label $L(n, A, \Sigma)$ of a node n is straightforward by Theorem 2.9 and Definition 4.1. Moreover, a query is not restricted to being a literal of N in this case: for a general formula, Theorem 2.10 can be applied by converting it to CNF.

Example 4.3 Let an ATMS be $\langle \{a, b, c, x, y\}, \{x, \neg y\}, \{\neg a \vee \neg b \vee c,\ \neg x \vee \neg b \vee a,\ y \vee b \vee c\} \rangle$. Then the following deduction finds c's label $\{x \wedge \neg y\}$:
$\langle \Box,\ \underline{\neg c} \rangle$, $\langle \Box,\ \underline{\neg a} \vee \neg b \vee \boxed{\neg c} \rangle$, $\langle \Box,\ \underline{\neg x} \vee \not\neg b \vee \boxed{\neg a} \vee \neg b \vee \boxed{\neg c} \rangle$, $\langle \neg x,\ \boxed{\not\neg c} \vee \underline{\neg b} \vee \boxed{\neg c} \rangle$, $\langle \neg x,\ \underline{y} \vee \not\neg \vee \boxed{\neg b} \vee \boxed{\neg c} \rangle$, $\langle \neg x \vee y,\ \boxed{\not\neg b} \vee \boxed{\not\neg c} \rangle$.

The question is how effectively consistency can be checked, that is, by testing whether a clause S produced from $\Sigma + \neg n$ and $\mathcal{P} = \langle \neg \cdot A \rangle$ belongs to $Th_{\mathcal{P}}(\Sigma)$ or not. A direct implementation is to use a theorem prover, as we already know that S belongs to $\mathcal{P}$, but theorem proving is also possible in m.c.l.s. resolution: there is an m.c.l. refutation from $\Sigma \cup \{\neg S\}$ iff there is an m.c.l.s. deduction from $\Sigma + \neg S$ and $\langle \phi \rangle$.

However, there is an another way for consistency checking. Unlike with the CMS, the computation of $Carc(\Sigma, \mathcal{P})$ can be performed better as the search focuses on the restricted vocabulary $\mathcal{P}$ if it is small compared with the whole literals $\mathcal{A}^{\pm}$. Having $Carc(\Sigma, \mathcal{P})$, consistency checking is much easier; $S \in Th_{\mathcal{P}}(\Sigma)$ iff there is a clause $T \in Carc(\Sigma, \mathcal{P})$ such that T subsumes S. The characteristic clauses $Carc(\Sigma, \langle \neg \cdot A \rangle)$ are called unsubsumed *nogoods* in the ATMS terminology. This checking can be embedded into an m.c.l.s. deduction: $Prod(\Sigma, C, \mathcal{P}) = Newcarc(\Sigma, C, \mathcal{P})$ if **Skip** (Rule 5(a)i) of Definition 2.5 is replaced with the following rule:

> 5(a)i'. **(Skip & Check)** If $P_i \cup \{l\}$ belongs to $\mathcal{P}$ and is not subsumed by any clause of $Carc(\Sigma, \mathcal{P})$, then the same as **Skip**.

4.2 Label Updating

In the compiled approach to an ATMS, the following result corresponding to Corollary 3.4 for the CMS and to a generalization of [21, Theorem 7] holds:

Theorem 4.4 Let $\langle N, A, \Sigma \rangle$ be an ATMS, $n \in N$ a literal, and $\mathcal{P} = \langle \neg \cdot A \rangle$.

$$Newcarc(\Sigma, \neg n, \mathcal{P}) = \{\ P - \{n\} \mid P \in PI(\Sigma),\ n \in P \text{ and } P - \{n\} \text{ belongs to } \mathcal{P}\ \}.$$

Theorem 4.4 shows that we can compute the label of a node from the prime implicates of Σ. Therefore an approach may keep $PI(\Sigma)$ and when a new clause C is added we compute $PI(\Sigma \cup \{C\})$ by Proposition 3.5 for updating labels of nodes. However, compared with the CMS many of the prime implicates are not significant for the task of an ATMS when the assumptions A are relatively small, although their computation is extremely high. Fortunately, we can compute a subset of $PI(\Sigma)$ enough to give labels by using the following stable production field:

Definition 4.5 Given an ATMS $\langle N, A, \Sigma \rangle$ and a production field $\mathcal{P} = \langle \neg \cdot A \rangle$, a production field $\mathcal{P}^*$ is defined as:

$$\mathcal{P}^* = \langle \neg \cdot A \cup N, \textit{ the number of literals in } N - \neg \cdot A \textit{ is at most one} \rangle.$$

Since $\mathcal{P}^*$ is stable, $Carc(\Sigma, \mathcal{P}^*)$ can be constructed incrementally by using Theorem 2.11:

$$Carc(\Sigma \cup \{C\}, \mathcal{P}^*) = \mu\,[\,Carc(\Sigma, \mathcal{P}^*) \cup Prod(\Sigma, C, \mathcal{P}^*)\,].$$

Here we only need to keep Σ and $Carc(\Sigma, \mathcal{P}^*)$. Looking further at Definition 4.5, we can show:

$$Carc(\Sigma, \mathcal{P}^*) = Carc(\Sigma, \mathcal{P}) \cup \{\, S \cup \{n\} \mid n \in N - \neg \cdot A \text{ and } S \in Newcarc(\Sigma, \neg n, \mathcal{P}) \,\}.$$

Therefore, the knowledge base can consist of the justifications Σ, unsubsumed nogoods $Carc(\Sigma, \mathcal{P})$, and prime implicates mentioning one node with the negation of an element of its label. No other prime implicates are necessary. Having $Carc(\Sigma, \mathcal{P}^*)$, we can find the label of each node $n \in N$ easily as follows:

Theorem 4.6 Let $\langle N, A, \Sigma \rangle$ be an ATMS, $n \in N$, $\mathcal{P} = \langle \neg \cdot A \rangle$, and $\mathcal{P}^*$ the same as Definition 4.5.

$$Newcarc(\Sigma, \neg n, \mathcal{P}) = \begin{cases} \{\, S - \{n\} \mid S \in Carc(\Sigma, \mathcal{P}^*), \text{ and } n \in S \,\} & \text{if } n \in N - \neg \cdot A \\ \{\, S - \{n\} \mid S \subset Carc(\Sigma, \mathcal{P}), \text{ and } n \in S \,\} & \text{if } n \in N \cap \neg \cdot A \end{cases}$$

For updating the knowledge base when a new clause C is added, again we just compute $Carc(\Sigma \cup \{C\}, \mathcal{P}^*)$ from the previous $Carc(\Sigma, \mathcal{P}^*)$ incrementally by using Theorem 2.11. Since this computation guarantees the completeness of characteristic-clause-finding, the four properties of the ATMS labels in Proposition 4.2 are also satisfied in this case. Note that the μ operation removes all the previous prime implicates that are subsumed by some newly added prime implicates. This operation is also crucial to guarantee the label consistency because implicates subsumed by some nogood must be removed.

Example 4.7 Suppose that an ATMS is $\langle \{a, b, x, y\}, \{x, y\}, \Sigma \rangle$ where $\Sigma = \{\, a \vee b,\ \neg y \vee a \,\}$. Now suppose that a new clause $\neg x \vee \neg a$ is added to Σ. Then the updating algorithm will find b's new label x, as well as a new unsubsumed nogood $\neg x \vee \neg y$:

$$\langle \square,\ \underline{\neg x} \vee \neg a \rangle,\ \ \langle \neg x,\ \underline{\neg a} \rangle,\ \ \langle \neg x,\ \underline{b} \vee \boxed{\neg a} \rangle,\ \ \langle \neg x \vee b,\ \boxed{\neg q} \rangle.$$
$$\hookrightarrow\ \ \langle \neg x,\ \underline{\neg y} \vee \boxed{\neg a} \rangle,\ \ \langle \neg x \vee \neg y,\ \boxed{\neg q} \rangle.$$

We thus see that $Carc(\Sigma, \mathcal{P}^*)$ can be used for giving labels for nodes. To maximize efficiency, however, it can be used also for caching the result of the production to be utilized later as the bypass of resolution. In [12], we describe how the updating algorithm can be modified for this purpose and still establish the label completeness for various ATMSs [3, 4, 5, 7], and the correspondence of the modified algorithm with de Kleer's label updating algorithms [3, 5].

5 Related Works

In this section, we compare our characteristic-clause-finding procedure to proof procedures of various abductive and nonmonotonic reasoning systems. The notions of production fields and (new) characteristic clauses are very helpful in understanding the relationships between them and in reconstructing them in our simple and general formalism.

5.1 Saturation

Bossu & Siegel [1] define a closed-world reasoning called sub-implication, in which all ground atoms are to be minimized. Their saturation procedure finds $Carc(\Sigma, \mathcal{P})$ where the characteristic literals $L_{\mathcal{P}}$ are fixed to positive ground literals. However, it does not use C-ordering.

Kean & Tsiknis [13] extend Tison's [26] consensus method of producing prime implicates to generate them incrementally. In our framework, the corresponding result is illustrated in Proposition 3.5. The difference is that their method is based on a set-of-support strategy, while ours uses linear resolution and thus naturally has more restriction strategies.

De Kleer [6] introduces hyperresolution rules to pre-compile a set of clauses Σ, all of which are either positive or negative. The negative clauses of the resulting set closed under these rules and subsumption are the characteristic clauses $Carc(\Sigma, \mathcal{P})$ where $\mathcal{P} = \langle \neg A,\ \textit{below size } k \rangle$ (see Example 2.2 (3)). In our formulation, instead of using hyperresolution, linear resolution can be used to produce such characteristic clauses for any clause set Σ and any characteristic literals $L_{\mathcal{P}} \subseteq \mathcal{A}^{\pm}$. In practice, this size-restriction is very useful for minimizing the computational effort, because it causes earlier pruning in m.c.l.s. deduction sequences.

5.2 Abduction via Deduction

There are many systems for logic-based abductive reasoning. However, many systems [18, 8, 20] other than [19] do not require minimality of explanation. Pople [19] proposed the mechanization of abduction via deduction based on SL-resolution [14]. However, his system does not distinguish literals, that is, the production field is fixed to $\mathcal{P}_{\pi}$, and "hypothesizes whatever cannot be proven". This criterion can be implemented if **Skip** (Rule 5(a)i) is preceded by **Resolve** (Rule 5(a)ii) and is applied only if **Resolve** cannot be applied in Step 5a of an m.c.l.s. deduction (Definition 2.5). Finger [8] gives residue procedures for abductive reasoning where assumptions are restricted to only atoms, but his "resolution residue" uses set-of-support resolution.

5.3 Query Answering for Circumscription

Przymusinski [20] defines MILO-resolution, a variant of OL-resolution [2], which is used in his circumscriptive theorem prover. MILO-resolution can be seen as m.c.l.s. resolution where the characteristic literals $L_{\mathcal{P}}$ are fixed to the positive occurrence of minimized predicates and any occurrence of fixed predicates in circumscription policies (see Inoue & Helft [11]).

Proposition 5.1 [20, 9, 11] Suppose that $L_{\mathcal{P}}$ is the same as in the above description and that $\mathcal{P} = \langle L_{\mathcal{P}} \rangle$. Every circumscriptive minimal model satisfies a formula F if and only if there is a conjunct G of clauses of $[Th_{\mathcal{P}}(\Sigma \cup \{\neg F\}) - Th_{\mathcal{P}}(\Sigma)]$ such that $[Th_{\mathcal{P}}(\Sigma \cup \{\neg G\}) - Th_{\mathcal{P}}(\Sigma)] = \phi$.

There is a big difference between MILO-resolution and the ATMS. In Proposition 5.1, to get theorems in $[Th_{\mathcal{P}}(\Sigma \cup \{C\}) - Th_{\mathcal{P}}(\Sigma)]$ for some clause C, MILO-resolution does not actually compute $Newcarc(\Sigma, C, \mathcal{P})$, while the ATMS does. Let us divide the produced clauses from $\Sigma + C$ and $\mathcal{P}$ possibly containing subsumed clauses into two sets, say $S1$ and $S2$, such that $\Sigma \cup S1 \models S2$. Then adding $S2$ to $S1$ does not change the models of the production. Thus only $S1$ needs to be computed model-theoretically[7]. We call a set $S1$ verifying this condition a *precursor* of the production. MILO-resolution computes such a precursor, because when the first literal belongs to $L_{\mathcal{P}}$ in Step 5a of an m.c.l.s. deduction (Definition 2.5), only **Skip** (Rule 5(a)i) is applied. On the contrary, since the CMS and the ATMS are used for computing *all and only minimal* supports for a query, if the literal resolved upon belongs to $L_{\mathcal{P}}$, they apply either **Skip** or **Resolve**[8]. Thus a precursor-finding algorithm can be written by ordering two rules as:

5(a)i'. **(Skip & Cut)** If $P_i \cup \{l\}$ belongs to $\mathcal{P}$, then the same as **Skip** (Rule 5(a)i).

5(a)ii'. **(Resolve')** Otherwise, the same as **Resolve** (Rule 5(a)ii).

Theorem 5.2 If a clause T is derived by an m.c.l.s. deduction from $\Sigma + C$ and $\mathcal{P}$, then there is a deduction with the **Skip & Cut** rule of a clause S from $\Sigma + C$ and $\mathcal{P}$ such that $\Sigma \cup \{S\} \models T$.

In [12], based on the above modification of m.c.l.s. resolution, we show a proof procedure for skeptical inference in an extended ATMS $\langle N, A_1 \cup \neg \cdot A_2, \Sigma \rangle$ where $A_1 \subseteq \mathcal{A}$ and $A_2 \subseteq \mathcal{A}$, which answers whether or not a formula is satisfied by every preferred model [22] of Σ.

6 Conclusion

We have shown a logical basis of procedural interpretation of abduction, the CMS and the ATMS based on linear resolution. The **Skip** rule can be safely embedded in linear resolution strategies making characteristic-clause-finding complete, due to the stability of production fields. While we used the description of OL-resolution as the definition of our linear resolution procedure, **Skip** can be applied to other, superior versions of propositional linear resolution, such as Shostak's graph construction procedure [23], and further improvements on these methods can be used to improve efficiency still more. We should also note that the control of inference can be made to the production in various ways as breadth-first or best-first search [2], integration of top-down and bottom-up strategies [8], reordering subgoal trees [24], and others.

Using the methods described in this paper, many AI techniques such as preferential-models approaches to nonmonotonic reasoning and constraint satisfaction problems, as well as direct applications of abduction or the ATMS, may be helped on the way to better implementation.

Acknowledgment

I am grateful to Nicolas Helft, a visiting researcher of ICOT in 1989, for introducing me Pierre Siegel's [24] work. I would like to thank David Poole, Mark Stickel, Kurt Konolige, Wolfgang Bibel and Koichi Furukawa for helpful discussions on this topic.

[7]However, not all of $S1$ are still the relevant parts needed to determine that F is in the circumscribed theory. The detailed discussion is given by Helft, Inoue & Poole [10].

[8]Since Ginsberg's circumscriptive theorem prover [9] is based on a backward-chaining "plain ATMS", it may produce more clauses than MILO-resolution. For more detailed discussion, see Inoue & Helft [11].

References

[1] Bossu, G. and Siegel, P., "Saturation, Nonmonotonic Reasoning, and the Closed-World Assumption", *Artificial Intelligence* **25** (1985), pp.23–67.

[2] Chang, C. L., and Lee, R. C. T., *Symbolic Logic and Mechanical Theorem Proving* (Academic Press, 1973).

[3] de Kleer, J., "An Assumption-based TMS", *Artificial Intelligence* **28** (1986), pp.127–162.

[4] de Kleer, J., "Extending the ATMS", *Artificial Intelligence* **28** (1986), pp.163–196.

[5] de Kleer, J., "A General Labeling Algorithm for Assumption-based Truth Maintenance", *Proc. AAAI-88* (1988), pp.188–192.

[6] de Kleer, J., *Propositional Inference in CSP and ATMS Techniques*, SSL Paper P89-00023, Xerox Palo Alto Research Center, 1989.

[7] Dressler, O., "An Extended Basic ATMS", *Proc. 2nd Int'l Workshop on Non-Monotonic Reasoning*, Lecture Notes in Artificial Intelligence 346, Springer-Verlag (1989), pp.143–163.

[8] Finger, J. J., *Exploiting Constraints in Design Synthesis*, Department of Computer Science, STAN-CS-88-1204, Stanford University, 1987.

[9] Ginsberg, M. L., "A Circumscriptive Theorem Prover", *Artificial Intelligence* **39** (1989), pp.209–230.

[10] Helft, N., Inoue, K. and Poole, D., *Extracting Answers in Circumscription*, ICOT Technical Memorandum TM-855, ICOT, 1989.

[11] Inoue, K. and Helft, N., "On Theorem Provers for Circumscription", *Proc. CSCSI-90*, Ottawa (1990), to appear.

[12] Inoue, K., *Procedural Interpretation for an Extended ATMS*, ICOT Technical Report TR-547, ICOT, 1990.

[13] Kean, A. and Tsiknis, G., *An Incremental Method for Generating Prime Implicants/Implicates*, Technical Report 88-16, Department of Computer Science, The University of British Columbia, 1988.

[14] Kowalski, R. A. and Kuhner, D. G., "Linear Resolution with Selection Function", *Artificial Intelligence* **2** (1971), pp.227–260.

[15] Levesque, H. J., "A Knowledge-level Account of Abduction (preliminary version)", *Proc. IJCAI-89* (1989), pp.1061–1067.

[16] Loveland, D., *Automated Theorem Proving: A Logical Basis*, (North-Holland, 1978).

[17] Minicozzi, E. and Reiter, R., "A Note on Linear Resolution Strategies in Consequence-Finding", *Artificial Intelligence* **3** (1972), pp.175–180.

[18] Poole, D., "A Logical Framework for Default Reasoning", *Artificial Intelligence* **36** (1988), pp.27–47.

[19] Pople, H. E., "On the Mechanization of Abductive Logic", *Proc. IJCAI-73* (1973), pp.147–152.

[20] Przymusinski, T. C., "An Algorithm to Compute Circumscription", *Artificial Intelligence* **38** (1989), pp.49–73.

[21] Reiter, R. and de Kleer, J., "Foundations of Assumption-based Truth Maintenance Systems: Preliminary Report", *Proc. AAAI-87* (1987), pp.183–188.

[22] Shoham, Y., "A Semantical Approach to Nonmonotonic Logics", *Proc. IJCAI-87* (1987), pp.388–392.

[23] Shostak, R., "Refutation Graphs", *Artificial Intelligence* **7** (1976), pp.51–64.

[24] Siegel, P., *Représentation et Utilisation de la Connaissance en Calcul Propositionnel*, PhD thesis, University of Aix-Marseille II, 1987.

[25] Slagle, J. R., Chang, C. L., and Lee, R. C. T., "Completeness Theorems for Semantic Resolution in Consequence Finding", *Proc. IJCAI-69* (1969), pp.281–285.

[26] Tison, P., "Generalized Consensus Theory and Application to the Minimization of Boolean Functions", *IEEE transactions on electronic computers* **16** (1967), pp.446–456.