ICOT Technical Report: TR-547

TR-547

Procedural Interpretation 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

Procedural Interpretation for an Extended ATMS

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

March 7, 1990

Abstract

This paper concerns deductive procedures for abductive reasoning and a variety of ATMSs. Although de Kleer's basic ATMS [4] has been widely used in AI, the algorithm has not yet been given any formal procedural semantics. Reiter & de Kleer [26] 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 the basic ATMS. In this paper, we use the notion of characteristic clauses [29] 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. Additionally, proof procedures for a class of nonmonotonic reasoning based on variations of the abductive procedure are also mentioned.

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

1 Introduction

An assumption-based truth maintenance system (ATMS) [4] has been widely used when problems require reasoning in multiple contexts. However, this basic ATMS can only handle the restricted form of formulas, and is described algorithmically rather than declaratively or model-theoretically, and no proof of its correctness is given, so it is not obvious how to generalize or refine it. The motivation for this research was the desire to formalize generalizations of the ATMS within simple model and proof theories.

Recent investigations such as those of Reiter & de Kleer [26] and Levesque [18] show that there are strong connections between the ATMS and a logical account of *abduction* or *hypothesis generation* [24, 3, 10, 22]. An ATMS can be characterized by the following type of abduction:

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) and (4) work as restrictions to accept appropriate explanations from the large number of explanations that satisfy (1) and (2). 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 [24] and Cox & Pietrzykowski [3] do not allow for such a distinguished set of literals but adopt another criterion. Poole [22] and Finger [10] compute an explanation satisfying (1), (2) and (3), if assumptions are restricted to being ground atoms, but they 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 [24]). 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 [4] 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 [5, 6, 7], Dressler's extended ATMS [9], and Reiter & de Kleer's clause management system (CMS) [26] if every ground literal is regarded as an assumption. Although the CMS is well defined, [26] 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, 29] 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). Two algorithms for ATMS label updating will be given: full and less-complete versions. The full algorithm computes all minimal explanations for a node; and the less-complete algorithm violates the label completeness, but is much more efficient. This extended ATMS can also capture a kind of nonmonotonic reasoning because it introduces the notion of negation [9]. 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 [25, 11], and membership in all extensions [23], based on abductive procedures [13, 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' ($C \subseteq C'$). In logical notation, C subsumes C' if $\models C \supset C'$. For a set of clauses Σ, by $\mu\Sigma$ or $\mu[\Sigma]$ we mean the set of clauses of Σ not subsumed by any other clause of Σ.

Definition 2.1 Let Σ be a set of clauses.
(1) A clause C is an *implicate* of Σ if $\Sigma \models C$. The set of implicates of Σ is denoted by $Th(\Sigma)$.
(2) 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 [29], 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 2.2 (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 $\mathcal{P}$.

Consequent properties. The empty clause $\square$ belongs to every stable production field.

Example 2.3 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 2.4 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.

Consequent properties. (1) If Σ is unsatisfiable, then $Carc(\Sigma, \mathcal{P})$ only contains $\square$.
(2) $PI(\Sigma) = Carc(\Sigma, \mathcal{P}_\pi)$.

$Carc(\Sigma, \mathcal{P})$ represents *saturation*: all the unsubsumed implicates of Σ that belong to a production field $\mathcal{P}$ must be contained in it. On the contrary, the next theorem shows that $Newcarc(\Sigma, \mathcal{P}, F)$ represents *abduction*.

Proposition 2.5 $Newcarc(\Sigma, F, \mathcal{P}) = \mu\,[Th_{\mathcal{P}}(\Sigma \cup \{F\}) - Th_{\mathcal{P}}(\Sigma)]$.

> **Proof:** Let $A = Th_{\mathcal{P}}(\Sigma \cup \{F\})$ and $B = Th_{\mathcal{P}}(\Sigma)$. Notice that $B \subseteq A$. We will prove that $\mu[A - B] = \mu A - \mu B$.
>
> Let $c \in \mu[A - B]$. Then obviously $c \in A - B$ and thus $c \in A$. Now assume that $c \notin \mu A$. Then $\exists d \in \mu A$ such that $d \subset c$. By the minimality of $c \in A - B$, $d \in B$. Since $d \subset c$, $c \in B$, contradiction. Therefore $c \in \mu A$. Clearly, by $c \notin B$, $c \notin \mu B$. Hence, $c \in \mu A - \mu B$.
>
> Conversely, assume that $c \in \mu A - \mu B$. Firstly we must prove that $c \in A - B$. Suppose to the contrary that $c \in B$. Since $c \notin \mu B$, $\exists d \in \mu B$ such that $d \subset c$. However, as $B \subseteq A$, $d \in A$, contradicting the minimality of $c \in A$. Therefore, $c \in A - B$. Now assume that c is not minimal in $A - B$. Then, $\exists e \in A - B$ such that $e \subset c$, again contradicting the minimality of $c \in A$. Hence, $c \in \mu[A - B]$. $\square$

Theorem 2.6 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 $\mathcal{P} = \langle \neg \cdot A \rangle$.

Proof: Suppose that H is an explanation of G with respect to Σ and A. By Definition 1.1, it is observed that (1) $\Sigma \cup \{H\} \models G$ can be written as $\Sigma \cup \{\neg G\} \models \neg H$, (2) the fact that $\Sigma \cup \{H\}$ is satisfiable means $\Sigma \not\models \neg H$, and (3) $\neg H$ is a clause all of whose literals belong to $\neg \cdot A$. Thus $\neg H \in [Th_{\mathcal{P}}(\Sigma \cup \{\neg G\}) - Th_{\mathcal{P}}(\Sigma)]$. By Proposition 2.5, H is a minimal explanation of G with respect to Σ and A if and only if $\neg H \in Newcarc(\Sigma, \neg G, \mathcal{P})$. □

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 □ 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 [30] and Minicozzi & Reiter [20]. 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 [29] proposes such a resolution method by extending SL-resolution [17]. In this paper, we use the basic idea of [29] 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* [20] [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 [17], the model elimination procedure [19], and the graph construction procedure [28].

[2] Roughly speaking, the skipping operation corresponds to Pople's [24] *synthesis* operation.

Definition 2.7 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 2.2 (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.

For m.c.l.s. resolution, the following theorem can be shown to hold.

Theorem 2.8 (1) *Decidability*: The algorithm stops for a set of propositional clauses Σ.
(2) *Soundness*: Every clause produced from Σ and $\mathcal{P}$ belongs to $Th_{\mathcal{P}}(\Sigma)$. In other words, if a clause S is derived using an m.c.l.s. deduction from $\Sigma + C$ and $\mathcal{P}$, then S belongs to $Th_{\mathcal{P}}(\Sigma \cup \{C\})$.
(3) *Completeness*: If a clause T belongs to $Th_{\mathcal{P}}(\Sigma)$, then there is an m.c.l.s. deduction of a clause S from Σ 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 [19]. These two observations were pointed out by Mark Stickel.

(4) *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 $\Sigma + C$ (that is, *with top clause* C) and $\mathcal{P}$ such that S subsumes T.

> **Proof:** The proof for the completeness can be seen as an extension of the result for linear resolution by Minicozzi & Reiter [20]. And these results follow easily using the same method as in the proofs for Siegel's procedure described in [29]. □

Note that m.c.l. resolution is refutation-complete [19, 17, 2], but is incomplete for consequence finding [20]. The procedure of m.c.l.s. resolution is complete for characteristic-clause-finding, because it includes the additional skipping operation.

Example 2.9 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 2.10 Given a set of clauses Σ, a clause C, and a stable production field $\mathcal{F}$, 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 [29], 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, by the soundness and the completeness' in Theorem 2.8, the following lemma can be shown to hold.

Lemma 2.11 Let C be a clause. $Newcarc(\Sigma, C, \mathcal{P}) \subseteq Prod(\Sigma, C, \mathcal{P}) \subseteq Th_{\mathcal{P}}(\Sigma \cup \{C\})$.

> **Proof:** By Theorem 2.8 (2), the second set-inclusion relationship easily follows. Let $T \in Newcarc(\Sigma, C, \mathcal{P})$. By Proposition 2.5, $T \in \mu[Th_{\mathcal{P}}(\Sigma \cup \{C\}) - Th_{\mathcal{P}}(\Sigma)]$. By Theorem 2.8 (4), $\exists S \in Prod(\Sigma, C, \mathcal{P})$ such that $S \subseteq T$. By the minimality of T, $T = S$. Hence, $Newcarc(\Sigma, C, \mathcal{P}) \subseteq Prod(\Sigma, C, \mathcal{P})$. □

The next theorem shows that we can compute $Newcarc(\Sigma, C, \mathcal{P})$ for a single clause C, without a naive implementation of Definition 2.4 (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 2.12 Let C be a clause. $Newcarc(\Sigma, C, \mathcal{P}) = Prod(\Sigma, C, \mathcal{P}) - Th_{\mathcal{P}}(\Sigma)$.

> **Proof:** By Lemma 2.11, $Newcarc(\Sigma, C, \mathcal{P}) \subseteq Prod(\Sigma, C, \mathcal{P})$. It remains to show that $Prod(\Sigma, C, \mathcal{P}) - Newcarc(\Sigma, C, \mathcal{P}) \subseteq Th_{\mathcal{P}}(\Sigma)$. Suppose to the contrary, for $S \in Prod(\Sigma, C, \mathcal{P}) - Th_{\mathcal{P}}(\Sigma)$, that $S \notin Newcarc(\Sigma, C, \mathcal{P})$. Since $S \notin Th_{\mathcal{P}}(\Sigma)$, $S \notin Carc(\Sigma, \mathcal{P})$. Because $S \in Prod(\Sigma, C, \mathcal{P})$, $S \in Th_{\mathcal{P}}(\Sigma \cup \{C\})$ by Lemma 2.11. Since S is not minimal by the supposition, $\exists S' \in Carc(\Sigma \cup \{C\})$ such that $S' \subset S$. Then, clearly $S' \notin Th_{\mathcal{P}}(\Sigma)$ as $S' \subset S$. Thus, $S' \in Th_{\mathcal{P}}(\Sigma \cup \{C\}) - Th_{\mathcal{P}}(\Sigma)$. By Theorem 2.8 (4), $\exists S'' \in Prod(\Sigma, C, \mathcal{P})$ such that $S'' \subseteq S' \subset S$. However, by Definition 2.10, $Prod(\Sigma, C, \mathcal{P})$ is μ-closed, that is, does not contain any redundant clauses, contradiction. Hence, $S \in Newcarc(\Sigma, C, \mathcal{P})$. □

For a CNF formula G, $Newcarc(\Sigma, G, \mathcal{P})$ can be computed by applying Theorem 2.12 incrementally. By using Proposition 2.5, we get the following theorem:

Theorem 2.13 Let $G = C_1 \wedge \cdots \wedge C_m$ be a CNF formula. Then

$$\begin{aligned} Newcarc(\Sigma, G, \mathcal{P}) &= \mu[\bigcup_{i=1}^{m} Newcarc(\Sigma_i, C_i, \mathcal{P})] \\ &= \mu[\bigcup_{i=1}^{m} Prod(\Sigma_i, C_i, \mathcal{P})] - Th_{\mathcal{P}}(\Sigma), \end{aligned}$$

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

Proof: Notice that in the following proof, for sets, A, B, and C, such that $C \subseteq B \subseteq A$, $A - C = (A - B) \cup (B - C)$ holds.

$$\begin{aligned} & Newcarc(\Sigma, G, \mathcal{P}) \\ = \; & \mu[\, Th_{\mathcal{P}}(\Sigma \cup \{C_1, \cdots, C_m\}) - Th_{\mathcal{P}}(\Sigma)\,] \text{ (by Proposition 2.5)} \\ = \; & \mu[\, (Th_{\mathcal{P}}(\Sigma_m \cup \{C_m\}) - Th_{\mathcal{P}}(\Sigma_m)) \cup \cdots \cup (Th_{\mathcal{P}}(\Sigma \cup \{C_1\}) - Th_{\mathcal{P}}(\Sigma))\,] \\ = \; & \mu[\, \mu[Th_{\mathcal{P}}(\Sigma_m \cup \{C_m\}) - Th_{\mathcal{P}}(\Sigma_m)] \cup \cdots \cup \mu[Th_{\mathcal{P}}(\Sigma_2) - Th_{\mathcal{P}}(\Sigma_1)]\,] \\ = \; & \mu[\, Newcarc(\Sigma_m, C_m, \mathcal{P}) \cup \cdots \cup Newcarc(\Sigma_1, C_1, \mathcal{P})\,] \\ = \; & \mu[\bigcup_{i=1}^{m} Newcarc(\Sigma_i, C_i, \mathcal{P})\,]. \end{aligned}$$

Now, the last equality can be shown to hold by using the following equation successively:

$$\begin{aligned} & Newcarc(\Sigma_{k+1}, C_{k+1}, \mathcal{P}) \cup Newcarc(\Sigma_k, C_k, \mathcal{P}) \\ = \; & (\, Prod(\Sigma_k \cup \{C_k\}, C_{k+1}, \mathcal{P}) - Th_{\mathcal{P}}(\Sigma_k \cup \{C_k\})\,) \cup (\, Prod(\Sigma_k, C_k, \mathcal{P}) - Th_{\mathcal{P}}(\Sigma_k)\,) \\ = \; & (\, Prod(\Sigma_k \cup \{C_k\}, C_{k+1}, \mathcal{P}) \cup Prod(\Sigma_k, C_k, \mathcal{P})\,) - (\, Th_{\mathcal{P}}(\Sigma_k \cup \{C_k\}) - Prod(\Sigma_k, C_k, \mathcal{P})\,) \\ & - (\, Th_{\mathcal{P}}(\Sigma_k) - Prod(\Sigma_k \cup \{C_k\}, C_{k+1}, \mathcal{P})\,) - (\, Th_{\mathcal{P}}(\Sigma_k \cup \{C_k\}) \cap Th_{\mathcal{P}}(\Sigma_k)\,) \\ = \; & (\, Prod(\Sigma_{k+1}, C_{k+1}, \mathcal{P}) \cup Prod(\Sigma_k, C_k, \mathcal{P})\,) - \phi \\ & - (\, Th_{\mathcal{P}}(\Sigma_k) - Prod(\Sigma_k \cup \{C_k\}, C_{k+1}, \mathcal{P})\,) - Th_{\mathcal{P}}(\Sigma_k) \\ = \; & (\, Prod(\Sigma_{k+1}, C_{k+1}, \mathcal{P}) \cup Prod(\Sigma_k, C_k, \mathcal{P})\,) - Th_{\mathcal{P}}(\Sigma_k). \end{aligned}$$

Hence, $Newcarc(\Sigma, G, \mathcal{P}) = \mu[\bigcup_{i=1}^{m} Prod(\Sigma_i, C_i, \mathcal{P})] - Th_{\mathcal{P}}(\Sigma)$. □

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 some stable production field $\mathcal{P}$, then $p \vee \neg p$ belongs to $Carc(\Sigma, \mathcal{P})$.

Theorem 2.14 The characteristic clauses with respect to $\mathcal{P}$ can be generated incrementally [4]:

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

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

Proof: The first equation follows immediately from Definition 2.4 (1). Now,

$$\begin{aligned}
Carc(\Sigma \cup \{C\}, \mathcal{P}) &= \mu Th_{\mathcal{P}}(\Sigma \cup \{C\}) \\
&= \mu [\, Th_{\mathcal{P}}(\Sigma \cup \{C\}) \cup Th_{\mathcal{P}}(\Sigma) \,] \\
&= \mu [\, \mu Th_{\mathcal{P}}(\Sigma \cup \{C\}) \cup \mu Th_{\mathcal{P}}(\Sigma) \,] \quad (*) \\
&= \mu [\, Carc(\Sigma, \mathcal{P}) \cup Carc(\Sigma \cup \{C\}, \mathcal{P}) \,] \\
&= \mu [\, Carc(\Sigma, \mathcal{P}) \cup (Carc(\Sigma \cup \{C\}, \mathcal{P}) - Carc(\Sigma, \mathcal{P})) \,] \\
&= \mu [\, Carc(\Sigma, \mathcal{P}) \cup Newcarc(\Sigma, C, \mathcal{P}) \,] \\
&= \mu [\, Carc(\Sigma, \mathcal{P}) \cup (Prod(\Sigma, C, \mathcal{P}) - Th_{\mathcal{P}}(\Sigma)) \,] \quad \text{(by Theorem 2.12)} \\
&= \mu [\, Carc(\Sigma, \mathcal{P}) \cup Prod(\Sigma, C, \mathcal{P}) \,] .
\end{aligned}$$

Notice that at $(*)$, for two sets, A and B, $\mu[\, A \cup B \,] = \mu[\, \mu A \cup \mu B \,]$ holds. □

3 The Clause Management System

Reiter & de Kleer [26] propose a generalization of the basic ATMS [4] 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 [26] 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. $MS(\Sigma, F) = Newcarc(\Sigma, \neg F, \mathcal{P}_{\pi})$.

Proof: A clause S is a support for F with respect to Σ

$\iff \Sigma \models S \cup F$, and $\Sigma \not\models S$

$\iff \Sigma \cup \{\neg F\} \models S$, and $\Sigma \not\models S$

$\iff S \in [\, Th(\Sigma \cup \{\neg F\}) - Th(\Sigma) \,]$.

Therefore, $S \in MS(\Sigma, F) \iff S \in Newcarc(\Sigma, \neg F, \mathcal{P}_{\pi})$ (by Proposition 2.5). □

This formulation can solve one of the limitations of the CMS. In [26], 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 2.13.

We thus see that our algorithm can compute minimal supports. However, Reiter & de Kleer [26] 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 2.12 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 2.12 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 [26, Theorem 5] Let C be a clause. Then

$$MS(\Sigma, C) = \mu\{\, P - C \mid P \in PI(\Sigma) \text{ and } P \cap C \neq \phi \,\}.$$

Corollary 3.4 [26, Corollary 4] Let $n \in \mathcal{A}^{\pm}$ be a literal. Then

$$MS(\Sigma, \{n\}) = \{\, P - \{n\} \mid P \in PI(\Sigma) \text{ and } n \in P \,\}.$$

Proposition 3.3 works only for a single clause query C. Levesque [18] generalizes it to handle a CNF formula query as follows [6]:

Proposition 3.5 [18, Theorem 3] Let $G = C_1 \wedge \cdots \wedge C_m$ be a CNF formula. Then

$$MS(\Sigma, G) = \{\, S \mid S \in \mu\nabla(\Sigma, G) \text{ and } S \text{ is not subsumed by any other clause in } PI(\Sigma) \},$$

$$\text{where } \nabla(\Sigma, G) = \{\, \bigcup_{i=1}^{m} (P - C_i) \mid P \in PI(\Sigma) \text{ and } P \cap C_i \neq \phi \,\}.$$

[5] Note that in [26] there is no description of an algorithm for the interpreted approach.

[6] We change the original expression in [18] slightly, so that the correspondence with Proposition 3.3 is clearer. Note that while Proposition 3.5 handles a CNF formula query directly, Theorem 2.13 handles a *DNF* formula query easily because of the duality of MS and $Newcarc$ in Proposition 3.2.

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 2.14 and the next lemma can be used by setting the production field to $\mathcal{P}_\pi$.

Lemma 3.6 For any stable production field $\mathcal{P}$, $Newcarc(\Sigma, C, \mathcal{P}) = Newcarc(PI(\Sigma), C, \mathcal{P})$.

Proof: Notice that $Carc(PI(\Sigma), \mathcal{P}) = Carc(Carc(\Sigma, \mathcal{P}_\pi), (\, L_\mathcal{P} \,)) = Carc(\Sigma, (\, \mathcal{L}^\pm \cap L_\mathcal{P} \,)) = Carc(\Sigma, \mathcal{P})$.

Now, $Carc(PI(\Sigma) \cup \{C\}, \mathcal{P}) = \mu Th_\mathcal{P}(\mu Th_{\mathcal{P}_\pi}(\Sigma) \cup \{C\}) = \mu Th_\mathcal{P}(Th(\Sigma) \cup \{C\}) = Carc(\Sigma \cup \{C\}, \mathcal{P})$.

The lemma follows immediately by Definition 2.4 (1). □

Proposition 3.7 Given $PI(\Sigma)$ and a clause C, $PI(\Sigma \cup \{C\})$ can be found incrementally:

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

Proof: The proposition follows by setting $\mathcal{P}$ to $\mathcal{P}_\pi$ in Theorem 2.14 and by using Theorem 2.12 and Lemma 3.6. □

By Proposition 3.7, 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 3.7 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 [7].

3.2 Comparing the Interpreted/Compiled Approaches

Reiter & de Kleer [26] also consider the trade-off between the compiled and the interpreted approaches. In the former, the computation of $PI(\Sigma)$ is very expensive, but retrieval is then efficient. In the latter, the CMS's database is kept as it is; the price we have to pay is a high retrieval cost. Now let us consider efficiency problems compared with such approaches. The serious problems are either computing supports with an uncompiled theory or compiling a theory. Each computational complexity is exponential. Proposition 3.7 shows that the interpreted approach is a particular case of the compiled one. The question is how efficiently will the algorithm work with a non-compiled theory.

In the CMS, the relative efficiency of the algorithm as compared with a brute-force algorithm comes from the restriction of resolution, as the key problem here is to generate as few as possible subsumed clauses together with making as few as possible subsumption tests.

But there is another interesting production field that offers an intermediate alternative to the compiled/interpreted disjunctive. The original theory Σ can yet be simplified, computing its reduction by replacing it with the set of *sub-clauses* implied by Σ.

[7] Reiter & de Kleer [26] 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.

Definition 3.8 The *reduction* of a set of clauses Σ is: $Reduced(\Sigma) = Carc(\Sigma, \mathcal{P}_R)$, where $\mathcal{P}_R = \langle \mathcal{A}^{\pm}, \textit{subclause of clauses of } \Sigma \rangle$.

Proposition 3.9 For any stable production field $\mathcal{P}$,

$$Newcarc(\Sigma, C, \mathcal{P}) = Newcarc(Reduced(\Sigma), C, \mathcal{P}).$$

Proof: It is easy to show that $Carc(Reduced(\Sigma), \mathcal{P}) = Carc(\Sigma, \mathcal{P})$. The proposition can be shown to hold in the similar manner to Lemma 3.6. □

Note that $\mathcal{P}_R$ is a stable production field. The original set Σ can be reduced to a great extent; computation from $Reduced(\Sigma)$ will be much more efficient than from Σ. For example, the set $\{ a, \neg a \vee b \}$ will be reduced to $\{ a, b \}$ (in this case producing the sub-clause is equivalent to saturation). However, $\{ a \vee c, \neg a \vee b \}$ contains all the sub-clauses implied by it and thus cannot be simplified, while the saturation would have added $b \vee c$.

4 An Extended ATMS

In de Kleer's versions of ATMSs [4, 5, 6, 7], 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 2.6, 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:

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

The following properties [4, 6] 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.

Proof: By Definition 4.1 and Theorem 2.6, $E_i \in L(n, A, \Sigma)$ is a minimal explanation of n with respect to Σ and A. Therefore, these four properties obviously hold by Definition 1.1. □

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 2.12 and Definition 4.1. Moreover, a query is not restricted to being a literal of N in this case: for a general formula, Theorem 2.13 can be applied by converting it to CNF.

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

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: $\Sigma \models S$ iff $Prod(\Sigma, \neg S, \mathcal{P}) = \{\square\}$. In this case, since we are not interested in any produced clause from $\Sigma + \neg S$ other than $\square$, the production field $\mathcal{P}$ can be replaced with $\langle \phi \rangle$ and **Skip** (Rule 5(a)i) will never be applied. Thus, 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 that offers an intermediate approach between the interpreted and compiled approaches. 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 [8]. The characteristic clauses $Carc(\Sigma, \langle \neg A \rangle)$ are called unsubsumed ***nogoods*** in the ATMS terminology. This checking can be embedded into an m.c.l.s. deduction: **Skip** (Rule 5(a)i) of Definition 2.7 can be 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**.

Proposition 4.4 If **Skip & Check** is used as Rule 5(a)i of an m.c.l.s. deduction instead of the original **Skip** rule, then $Prod(\Sigma, C, \mathcal{P}) = Newcarc(\Sigma, C, \mathcal{P})$.

[8] Note that even if we have $Carc(\Sigma, \mathcal{P})$, we cannot compute $Newcarc(\Sigma, \neg n, \mathcal{P})$ from it in the same way as Corollary 3.4 for the CMS, because in general for a stable production field $\mathcal{P}$, $Newcarc(\Sigma, C, \mathcal{P}) \neq Newcarc(Carc(\Sigma, \mathcal{P}), C, \mathcal{P})$. That is why for the compiled approach to an ATMS an another technique is anticipated (see Section 4.2).

Proof: Because every clause belonging to $Th_{\mathcal{P}}(\Sigma \cup \{C\})$ that is subsumed by some clause in $Carc(\Sigma, \mathcal{P})$ must be pruned in a deduction sequence, every clause produced from $\Sigma + C$ is not a super-clause of any clause in $Carc(\Sigma, \mathcal{P})$ and thus does not belong to $Th_{\mathcal{P}}(\Sigma)$. Hence, by Theorem 2.12, the proposition follows. □

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 [26, Theorem 7] holds:

Theorem 4.5 Let $\langle N, A, \Sigma \rangle$ be an ATMS, $n \in N$ a literal, and $\mathcal{P} = \langle \neg \cdot A \rangle$.

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

Proof: ($\supseteq$) Let $P \in PI(\Sigma)$ such that $n \in P$ and $P - \{n\}$ belongs to $\mathcal{P}$. Then, since $P - \{n\} \subset P$, $\Sigma \not\models P - \{n\}$. Since $n \in P$ and $\Sigma \models P$, $\Sigma \cup \{\neg n\} \models P - \{n\}$. Therefore, $P - \{n\} \in Th_{\mathcal{P}}(\Sigma \cup \{\neg n\}) - Th_{\mathcal{P}}(\Sigma)$. As $P \in PI(\Sigma)$ and $n \in P$, for any clause $S \subset P - \{n\}$, $\Sigma \not\models S \cup \{n\}$, and thus $\Sigma \cup \{\neg n\} \not\models S$ holds. This implies that $P - \{n\} \in Carc(\Sigma \cup \{\neg n\}, \mathcal{P})$, and thus $P - \{n\} \in Newcarc(\Sigma, \neg n, \mathcal{P})$ [9].

($\subseteq$) Let $S \in Newcarc(\Sigma, \neg n, \mathcal{P})$. As $\Sigma \cup \{\neg n\} \models S$, $\Sigma \models S \cup \{n\}$ holds. Suppose that $\exists T \in Th(\Sigma)$ such that $T \subset S \cup \{n\}$ and that $T - \{n\}$ belongs to $\mathcal{P}$. Clearly, $\Sigma \models T \cup \{n\}$. Now for any clause S' such that $S' \subset S$, since $\Sigma \cup \{\neg n\} \not\models S'$, $\Sigma \not\models S' \cup \{n\}$ holds. Therefore, $S \subseteq T \subset S \cup \{n\}$. As n is a literal, $T = S$. However, $S \notin Carc(\Sigma, \mathcal{P})$, contradiction. Hence, $S \cup \{n\} \in PI(\Sigma)$. Replacing $S \cup \{n\}$ with P, we get the theorem. □

Theorem 4.5 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 3.7 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. In such a case, we do not want to compute all the prime implicates. Fortunately, we can compute a subset of $PI(\Sigma)$ enough to give labels by using the following stable production field:

Definition 4.6 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 2.14:

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

Here we only need to keep Σ and $Carc(\Sigma, \mathcal{P}^*)$. Looking further at Definition 4.6, the relationship between $Carc(\Sigma, \mathcal{P}^*)$ and $Carc(\Sigma, \mathcal{P})$ can be shown exactly in the next lemma:

[9] Note that in this direction n need not be a literal in N; the relation holds for a clause C: if $P \in PI(\Sigma)$, $C \subseteq P$, and $P - C$ belongs to $\mathcal{P}$, then $\neg \cdot (P - C)$ is a minimal explanation of C with respect to Σ and A. This result corresponds to a generalization of [26, Theorem 3] for a general $\mathcal{P}$.

Lemma 4.7 $Carc(\Sigma, \mathcal{P}^*) = Carc(\Sigma, \mathcal{P}) \cup \{ S \cup \{n\} \mid n \in N - \neg \cdot A \text{ and } S \in Newcarc(\Sigma, \neg n, \mathcal{P}) \}$.

Proof: $Carc(\Sigma, \mathcal{P}^*)$ can be divided into two disjoint sets of clauses: (1) containing no literal in $N - \neg \cdot A$, and (2) containing exactly one literal in $N - \neg \cdot A$. The former is exactly $Carc(\Sigma, \mathcal{P})$. Assume that C belongs to the latter set, and that $n \in C$ is a literal in $N - \neg \cdot A$. Then, $C - \{n\}$ only contains literals in $\neg \cdot A$ and thus belongs to $\mathcal{P}$. We must show that $C - \{n\} \in Carc(\Sigma \cup \{\neg n\}, \mathcal{P})$. Since $C \in Carc(\Sigma, \mathcal{P})$, $\Sigma \models C$ and thus $\Sigma \cup \{\neg n\} \models C - \{n\}$. Suppose to the contrary that $\exists S \in Th_{\mathcal{P}}(\Sigma \cup \{\neg n\})$ such that $S \subset C - \{n\}$. Then, $S \cup \{n\} \subset C$ and $S \cup \{n\} \notin Th_{\mathcal{P}}(\Sigma)$ by the minimality of $C \in Th_{\mathcal{P}}(\Sigma)$. Since $\Sigma \not\models S \cup \{n\}$, $\Sigma \cup \{\neg n\} \not\models S$, contradiction. Therefore, $C - \{n\} \in Carc(\Sigma \cup \{\neg n\}, \mathcal{P})$. Since $C \in Carc(\Sigma, \mathcal{P}^*)$, obviously $C - \{n\} \notin Carc(\Sigma, \mathcal{P})$ holds. Hence, the lemma. □

Therefore, the knowledge base can consist of the justifications Σ, unsubsumed nogoods $Carc(\Sigma, \mathcal{P})$, and prime implicates mentioning one node $n \in N - \neg \cdot A$ 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 4.8 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 4.6.

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

Proof: ($\supseteq$) Obvious from Theorem 4.5 and Lemma 4.7.

($\subseteq$) Let $T \in Newcarc(\Sigma, \neg n, \mathcal{P})$. By Theorem 4.5, $T \cup \{n\} \in PI(\Sigma)$. (1) If $n \in N - \neg \cdot A$, then $T \cup \{n\}$ belongs to $\mathcal{P}^*$ because T belongs to $\mathcal{P}$. Therefore, $T \cup \{n\} \in Carc(\Sigma, \mathcal{P}^*)$. (2) If $n \in N \cap \neg \cdot A$, then $T \cup \{n\}$ belongs to $\mathcal{P}$. Therefore, $T \cup \{n\} \in Carc(\Sigma, \mathcal{P})$. □

Note that in the second case of Theorem 4.8, $Carc(\Sigma, \mathcal{P}) \subseteq Carc(\Sigma, \mathcal{P}^*)$ holds. This means that for $n \in \neg \cdot A$ the label of n is computed from unsubsumed nogoods containing n if $n \in N$. 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 2.14. 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. We call the procedure using m.c.l.s. resolution based on Theorem 2.14 and Theorem 4.8 the *full* procedure for label updating.

Example 4.9 Suppose that an ATMS is $\langle \{a, b, x, y\}, \{x, y\}, \Sigma \rangle$ where $\Sigma = \{ a \vee b, \neg y \vee a \}$. In this case $Carc(\Sigma, \mathcal{P}^*) = \Sigma \cup \{ \neg x \vee x, \neg y \vee y \}$. 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 $\neg x \vee \neg y$:

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

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 for the next updating. In the next section, we describe how the updating algorithm can be modified for this purpose and still establish the label completeness for various ATMSs [4, 5, 6, 9], and the correspondence of the modified algorithm with de Kleer's label updating algorithms [4, 6].

4.3 Optimization by Bypassing Resolution

In Section 4.2 we showed that the full procedure can update the label for each node $n \in N$ from the previous $Carc(\Sigma, \mathcal{P}^*)$ and Σ when a new clause C is added to Σ. However in computing $Carc(\Sigma \cup \{C\}, \mathcal{P}^*)$ in Theorem 2.14, some of the resolution in $Prod(\Sigma, C, \mathcal{P}^*)$ will often be the same as ones in the previous computation of characteristic clauses. As we took grate pains to get $Carc(\Sigma, \mathcal{P}^*)$, that information should be used in new deductions. This means that $Carc(\Sigma, \mathcal{P}^*)$ is used not only for giving labels for nodes but also for caching the result of the production to be utilized for the next updating. This leads us to introduce the following additional rule into Step 5a of an m.c.l.s. deduction (Definition 2.7).

> 5(a)iii. **(Bypass)** If $l \in \neg \cdot N$, then $P_{i+1} = P_i$ and $\vec{R}_{i+1}$ is an ordered resolvent of $\vec{Q}_i$ with a clause $B_i \in Carc(\Sigma, \mathcal{P}^*)$, where $\neg l \in B_i$ and $B_i - \{\neg l\}$ belongs to $\mathcal{P}$.

The resolvent $\vec{R}_{i+1}$ obtained by **Bypass** satisfies $R_{i+1} = (B_i - \{\neg l\}) \cup (Q_i - \{l\})$. Since $B_i - \{\neg l\}$ belongs to $\mathcal{P}$, there is a deduction from $\Sigma + (B_i - \{\neg l\})$ and $\mathcal{P}$ where all of literals are skipped. Since $B_i \in Carc(\Sigma, \mathcal{P}^*)$, $B_i - \{\neg l\} \in Newcarc(\Sigma, l, \mathcal{P})$ by Theorem 4.8. This implies that the deduction is also obtained from $\Sigma + l$ and $\mathcal{P}$ by the completeness' of m.c.l.s. resolution (Theorem 2.8 (4)). Conversely, if a clause T is derived by an m.c.l.s. deduction from $\Sigma + l$ and $\mathcal{P}$, then there exists a clause S in $Carc(\Sigma, \mathcal{P}^*)$ such that $\neg l \in S$ and that $S - \{\neg l\}$ belongs to $\mathcal{P}$ and subsumes T. That is why **Bypass** works as the saving of computation.

To maximize efficiency, however, **Resolve** (Rule 5(a)ii) and **Bypass** (Rule 5(a)iii) should be exclusively applied. The easiest and the most efficient way is to apply **Resolve** only when **Bypass** cannot be applied. In this case, Step 5a in m.c.l.s. deduction (Definition 2.7) is changed to the following rules:

> 5(a)i'. **(Skip')** If $P_i \cup \{l\}$ belongs to $\mathcal{P}^*$, then the same as **Skip** (Rule 5(a)i).
>
> 5(a)ii'. **(Resolve')** If $l \notin \neg \cdot N$, then $P_{i+1} = P_i$ 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.
>
> 5(a)iii'. **(Bypass')** If $l \in \neg \cdot N$, then the same as the above **Bypass**.

Note that again **Skip** is not exclusive to other two. The problem of this method is the incompleteness. The reason is that **Bypass** is complete only for m.c.l.s. deduction from $\Sigma + C$ and $\mathcal{P}$, not $\mathcal{P}^*$. We call the procedure based on Theorem 2.14 and m.c.l.s. resolution with this modification the *less-complete* procedure for label updating.

Example 4.10 Suppose that an ATMS is $\langle \{a, b, x, y\}, \{x, y\}, \Sigma \rangle$ where $\Sigma = \{ a \vee b, \neg y \vee a \}$ (the same as Example 4.9). When a new clause $\neg x \vee \neg a$ is added to Σ, the less-complete

algorithm cannot find b's new label x, but the full algorithm finds it by resolving $\neg x \vee \neg a$ with $a \vee b$ as:

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

A new unsubsumed nogood $\neg x \vee \neg y$ can be found in this case by both algorithms:

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

However, if a next new clause $\neg b$ is added to $\Sigma \cup \{\neg x \vee \neg a\}$, the less-complete algorithm cannot find a new unsubsumed nogood $\neg x$.

In some cases, the less-complete algorithm is complete for label updating. The completeness depends on how to build an ATMS, that is, how to choose the elements of N and A from $\mathcal{A}^{\pm}$.

Example 4.11 Suppose that an ATMS be $\langle \{a, b, \neg b, x, y\}, \{x, y, b, \neg b\}, \Sigma \rangle$ where Σ is the same as the previous example. This time the less-complete algorithm is complete for label updating because $Newcarc(\Sigma, \neg a, \mathcal{P}) = \{b\}$ holds and thus **Bypass'** is applied to $\neg a$ making an deduction continue (but the result is the same as the application of **Resolve'** in this case).

The last example suggests that we should find a class in which the less-complete algorithm is complete for updating the label of every node. And more **Resolve** is applied, more complete the algorithm is. Therefore, we should change the condition of **Bypass** and apply **Resolve** to some literals in $\neg \cdot N$. We can show a sufficient condition for that.

Definition 4.12 Let N be a set of literals. A clause is *N-Horn* if it is either of the form $\neg\alpha_1 \vee \cdots \vee \neg\alpha_k \vee \beta$, or of the form $\neg\alpha_1 \vee \cdots \vee \neg\alpha_k$, where $\alpha_i \in N$, $k \geq 0$ and $\beta \in N$.

Lemma 4.13 Let $\langle N, A, \Sigma \rangle$ be an ATMS. If (1) Σ is a set of N-Horn clauses, (2) $N \cap \neg N \subseteq A$ holds, and (3) l is a literal belonging to $\neg \cdot N - N$, then $Prod(\Sigma, l, \mathcal{P}^*) = Prod(\Sigma, l, \mathcal{P})$.

Proof: ($\supseteq$) Obvious from Definition 4.6.

($\subseteq$) We will prove that if T is derived from $\Sigma + l$ and $\mathcal{P}^*$, then there is an m.c.l.s. deduction of S from $\Sigma + l$ and $\mathcal{P}$ such that $S \subseteq T$. Firstly, in case that **Skip** (Rule 5(a)i) is applied for $l \in \neg \cdot A$, l is deduced. As l belongs to $\mathcal{P}$, $l \in Prod(\Sigma, l, \mathcal{P})$ holds.

Otherwise, **Resolve** (Rule 5(a)ii) is applied. Let $B = \neg\alpha_1 \vee \cdots \vee \neg\alpha_k \vee \neg l$ be a side N-Horn clause from Σ, where $\alpha_i \in N$ $(1 \leq i \leq k)$. Notice that $\neg l \in N - \neg \cdot N$. Since $N \cap \neg \cdot N \subseteq A$, $N \cap (\neg \cdot N - A) = \phi$ holds, and thus $\alpha_i \notin \neg \cdot N - A$. Therefore, $\alpha_i \in N - \neg \cdot N$, and thus $\neg\alpha_i \in \neg \cdot N - N$.

Now, every first literal l_j of ordered resolvent $\vec{Q}_j$ in each step satisfies the condition $l_j \in \neg \cdot N - N$. Applying the above argument for l to l_j, we can show that every produced clause P_j belongs to $\mathcal{P}$. Hence, the lemma holds. $\Box$

The second condition $N \cap \neg \cdot N \subseteq A$ means that if both a propositional symbol p and its negation $\neg p$ are in N, then they must be put into assumptions A. Notice that $a \vee b$ is not N_1-Horn for $N_1 = \{a, b, x, y\}$ (Example 4.10) but is N_2-Horn for $N_2 = \{a, b, \neg b, x, y\}$ (Example 4.11).

Theorem 4.14 Let $\langle N, A, \Sigma \rangle$ be an ATMS. If (1) Σ is a set of N-Horn clauses, and (2) $N \cap \neg \cdot N \subseteq A$ holds, then the less-complete algorithm replaced the conditions of Rules 5(a)ii′ and 5(a)iii′ by the following is complete for updating the label of every node in N:

5(a)ii″. **(Resolve″)** If $l \notin \neg \cdot N - N$, then the same as **Resolve**.

5(a)iii″. **(Bypass″)** If $l \in \neg \cdot N - N$, then the same as **Bypass**.

Proof: We will firstly prove that every deduction by using the above algorithm can also be obtained by an m.c.l.s. deduction. The **Bypass″** rule tries to find a clause B_i in $Carc(\Sigma, \mathcal{P}^*)$ that contains $\neg l$, where $l \in \neg \cdot N - N$. Since in this case $\neg l \in N - \neg \cdot A$, $B_i - \{\neg l\} \in Newcarc(\Sigma, l, \mathcal{P})$ by Theorem 4.8. Therefore, $B_i - \{\neg l\}$ is m.c.l.s. derived from $\Sigma + l$ and $\mathcal{P}$ by the completeness′ of m.c.l.s. resolution (Theorem 2.8 (4)): in fact, since $(B_i - \{\neg l\})$ belongs to $\mathcal{P}$, there is a deduction from $\Sigma + (B_i - \{\neg l\})$ and $\mathcal{P}$ where all of literals are skipped. By Lemma 4.13, it can be also m.c.l.s. derived from $\Sigma + l$ and $\mathcal{P}^*$.

Conversely, let C be a clause newly added to Σ. We will now show that if a clause T is m.c.l.s. derived from $\Sigma + C$ and $\mathcal{P}^*$, then there is a deduction by using the above algorithm of a clause S from $\Sigma + C$ and $\mathcal{P}^*$ such that $S \subseteq T$. With these facts, the theorem holds due to the completeness result by Theorem 2.13 and the result of label finding by Theorem 4.8.

Now, let $T \subset Prod(\Sigma, C, \mathcal{P}^*)$ be a clause in $Carc(\Sigma \cup \{C\}, \mathcal{P}^*)$. By Lemma 4.7 and Theorem 4.8, if $\exists \neg l \in N - \neg \cdot A$ such that $\neg l \in T$, then $T - \{\neg l\} \in Newcarc(\Sigma, l, \mathcal{P})$. Note that in this case $l \in \neg \cdot N - N$ holds, because by the condition (2) $N \cap \neg \cdot N \subseteq A$, $(\neg \cdot N - A) - (\neg \cdot N - N) = (N \cap \neg \cdot N) - A = \phi$ holds. Then, there is an m.c.l.s. deduction of $T - \{\neg l\}$ from $\Sigma + l$ and $\mathcal{P}$ (by Theorem 2.8 (4)). By Lemma 4.13, $T - \{\neg l\} \in Prod(\Sigma, l, \mathcal{P})$ implies $T - \{\neg l\} \in Prod(\Sigma, l, \mathcal{P}^*)$. Such l must appear in a deduction using the above algorithm, and **Bypass″** can be applied. If $\exists \neg l \in N \cap \neg \cdot A$ and thus $\neg l \notin N - \neg \cdot N$, then l can be resolved upon by the modified algorithm. Therefore, T is also derived by the above modified algorithm. Hence, the theorem. □

We call the procedure used in Theorem 4.14 the *modified* less-complete algorithm for label updating. Now let us verify how our modified less-complete algorithm establish the label completeness for various ATMSs. The following is the reconstruction of various ATMSs:
Let $\langle N, A, \Sigma \rangle$ be an ATMS. Recall that $N \subseteq \mathcal{A}^{\pm}$ and $A \subseteq N$.

1. *The basic ATMS* [4]. In this case, $N = \mathcal{A}$ and Σ is a set of Horn ($\mathcal{A}$-Horn) clauses. Since $\mathcal{A} \cap \neg \cdot \mathcal{A} = \phi$, the algorithm is complete. The conditions of Rules 5(a)ii″ and 5(a)iii″ are: if $l \in \neg \cdot \mathcal{A}$, then **Bypass**; otherwise, **Resolve**.

2. *The negated assumption ATMS* (NATMS) [6]. In this case, $A \subseteq \mathcal{A}$, $N = \mathcal{A} \cup \neg \cdot A$ and Σ is a set of N-Horn clauses. Since $N \cap \neg \cdot N = A \cup \neg \cdot A \not\subseteq A$, the algorithm is incomplete.

3. *The extended basic ATMS* [9]. In this case, $A = A_1 \cup \neg \cdot A_2$ where $A_1 \subseteq \mathcal{A}$ and $A_2 \subseteq \mathcal{A}$, $N = \mathcal{A} \cup \neg \cdot A_2$, and Σ is a set of N-Horn clauses. Since $N \cap \neg \cdot N = A_2 \cup \neg \cdot A_2 \not\subseteq A_1 \cup \neg \cdot A_2$, the algorithm is incomplete.

4. *The CMS* [26]. In this case, $N = A = \mathcal{A}^{\pm}$, and Σ is a set of general ($\mathcal{A}^{\pm}$-Horn) clauses. Since $N \cap \neg \cdot N = N = \mathcal{A}^{\pm} = A$, the algorithm is complete. The conditions of Rules 5(a)ii'' and 5(a)iii'' coincide with one in the full algorithm, that is, the original one of m.c.l.s. resolution (Definition 2.7): for any l, apply **Resolve**.

Example 4.15 Σ is the same as Example 4.10.
(1) Let $N_3 = \{a, b, \neg b, x, \neg x, y, \neg y\}$ and $A = \{x, y, b\}$. This is an NATMS setting as $a \vee b$ is N_3-Horn, and is incomplete ($N_3 \cap \neg \cdot N_3 = \{b, \neg b, x, \neg x, y, \neg y\} \not\subseteq A$). In fact, $\neg a \in \neg \cdot N_3 - N_3$ cannot be resolved upon with $a \vee b$ in the algorithm.
(2) Let $N_4 = \{a, \neg a, b, \neg b, x, \neg x, y, \neg y\}$ and $A = \{x, y, a, b\}$. In this case an NATMS is complete even though $N_4 \cap \neg \cdot N_4 \not\subseteq A$. In fact, $\neg \cdot N_4 - N_4 = \phi$ holds, so $\neg a$ can be resolved upon.

The above example shows a way for an NATMS to be complete for label updating in our modified less-complete algorithm. If a literal $l \in \neg \cdot N$ is expected to be resolved upon, then (1) l should be put into N ($l \notin \neg \cdot N - N$), and (2) $\neg l \in \mathcal{A}$ should be put into A. This technique corresponds to de Kleer's "encoding trick" [6]. In the similar manner, Dressler [9] proposes a way for an ATMS to be complete. Instead of the above (2), Dressler requires for a literal $l \in \neg \cdot N$ to be resolved upon that (2') $l \in \neg \cdot \mathcal{A}$ should be put into $\neg \cdot A_2 \subseteq A$. Note that auxiliary assumptions introduced to obtain the label completeness are used only for intermediate computation and should be ignored in the result.

The correspondence of less-complete algorithms with de Kleer's label updating algorithms [4, 6] is the following:

- The knowledge base is the same: N-Horn justifications Σ, the complete unsubsumed nogoods $Carc(\Sigma, \mathcal{P})$, and prime implicates mentioning exactly one node with the negations of an element of its label.
- The **Bypass** and **Skip** operations correspond to de Kleer's incremental construction of a tentative new label for each node, which is a union of previous or propagated new labels of antecedent nodes in a justification.
- The **Resolve** operation corresponds to de Kleer's propagation of a new label of a node to other nodes through justifications which contain that node in their antecedents.
- As in the full algorithm in Section 4.2, the μ operation is crucial to guarantee the label consistency and the label minimality.

In spite of these similarities between our less-complete algorithms and de Kleer's, there are still differences: the NATMS algorithm [6] is still less complete. This is because de Kleer's algorithm does not precisely depends on Theorem 4.14 and the application of **Resolve** is more restricted (but **Bypass** is more applied). This is verified by the next example:

Example 4.16 Suppose that an NATMS is $\langle \{a, x, \neg x\}, \{x\}, \Sigma \rangle$ where $\Sigma = \{ x \vee a \}$. When a new clause $\neg x \vee a$ is added to Σ, the NATMS algorithm cannot find a's new label $\{\}$. Since $\neg x \in N$ and $x \in A$, $\neg x$ should be resolved upon with $x \vee a$. In our modified algorithm, it is surely resolved. However, de Kleer's algorithm does not resolve it, because it restricts the application of the propagation only to literals in $N - \neg \cdot A = \{a, x\}$.

5 Nonmonotonicity and Skepticism

This section gives some semantical considerations on an extended ATMS. The semantics of the ATMS is close to the notion of minimization [10]. Each assumption is to be maximized in an explanation process. Maximization of a proposition p is equivalent to minimization of $\neg p$. Given an ATMS $\langle N, A, \Sigma \rangle$, we can think of a minimal model of Σ with respect to A which satisfies a maximal consistent set of assumptions (called an *interpretation* in the ATMS terminology [5]). We can define the equivalence relation on such minimal models of Σ according to each interpretation, and call an equivalence class an *A-extension* of Σ. Obviously, the label of $n \in N$ is not empty iff n is satisfied in every model in an A-extension.

In an extended ATMS $\langle N, A_1 \cup \neg A_2, \Sigma \rangle$ where $A_1 \subseteq \mathcal{A}$ and $A_2 \subseteq \mathcal{A}$ (Dressler's setting [9]). A_1 is to be maximized and A_2 is to be minimized. Given some set E of assumptions, Dressler defines an A-extension whose every model satisfies all of E and as many negated assumptions from $\neg \cdot A_2$ as possible. He requires that a node n should be satisfied by every model in such an A-extension to conclude that n holds in E. This is a weak form of nonmonotonic or closed-world reasoning. However, it is even reasonable for us to predict a formula if it is satisfied by all minimal models of Σ with respect to A. This inference is called *skeptical*, and is a *preferential-models* approach to nonmonotonic reasoning [27]. We will consider an implementation of this skeptical inference in an ATMS later in circumscriptive theorem provers (Section 6.3).

Notice that in the above ATMS, the intersection of A_1 and A_2 represents the *fixed* propositional symbols and $\mathcal{A} - (A_1 \cup A_2)$ represents *variables*. It is interesting to note that in the CMS [26] all literals are fixed ($A_1 = A_2 = \mathcal{A}$), so that every model of Σ is a preferred model and preferential entailment is equivalent to ordinary entailment. That is why the CMS cannot handle nonmonotonic reasoning.

Example 5.1 This is an example where Dressler's ATMS can behave itself nonmonotonically. Let $\Sigma = \{ \neg x \supset a \ , \neg a \supset b \}$. If we regard negations in antecedents of justifications as out-lists, assuming x causes a's outness and thus makes b in. By Definition 3.1, $MS(\Sigma, \{b\}) = \{ \neg b, \ \neg a \}$. Hence, x cannot explain b in the CMS. However, let an ATMS be $\langle \{x, \neg x, a, \neg a, b\}, \{x, \neg x, \neg a\}, \Sigma \rangle$. Then, given the assumption $\{x\}$, we can find an only minimal model $\{x, \neg a, b\}$ of $\Sigma \cup \{x\}$ and thus conclude that b holds in $\{x\}$.

6 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.

[10] Dressler [9] and Junker [14] discuss the nonmonotonicity problem that requires each label to be grounded [8]. In our formulation, the groundedness cannot be handled, as we only implement negated assumptions which differ out-lists of Doyle's nonmonotonic justifications. It is impossible to express the outness in propositional logic. To obtain the groundedness, we must add some external mechanism to a deduction procedure such as Dressler's and Junker's systems. Therefore, instead of pursuing the groundedness, we relate a nonmonotonic ATMS with the notion of minimization.

6.1 Saturation

Bossu & Siegel [1] define a closed-world reasoning called sub-implication, whose notion is similar (but not equivalent) to Minker's generalized closed world assumption (GCWA) [21] for propositional theories, 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 (see Example 2.3 (2)). However, it does not use C-ordering, but uses an earlier version of resolution method based on A-ordering [16]. Moreover, their method to compute $Newcarc(\Sigma, C, \mathcal{P})$ is a naive implementation of Definition 2.4. Note that those versions of CWA can be generalized to allow for variable and fixed predicates as well as minimized predicates, that is, circumscription of ground theories.

Kean & Tsiknis [15] extend Tison's [31] consensus method of producing prime implicates to generate them incrementally. In our framework, the corresponding result is illustrated in Proposition 3.7. The difference is that their method is based on a set-of-support strategy, where subsumption checking is performed at each resolution step, while ours uses linear resolution and thus naturally has more restriction strategies that Kean & Tsiknis manage to incorporate for optimization.

De Kleer [7] introduces hyperresolution rules to pre-compile a set of clauses Σ, all of which are either positive or negative. This technique is also given in [5] in more general form. An interesting approach in [7] is to use a rule limiting the inference only to negative clauses below a size k. The negative clauses of the resulting set closed under these rules and subsumption are the characteristic clauses $Carc(\Sigma, \mathcal{P})$ where $\mathcal{P} = \langle \neg \cdot A,\ \textit{below size } k \rangle$ (see Example 2.3 (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 $L_{\mathcal{P}} \subseteq \mathcal{A}^{\perp}$. 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.

6.2 Abduction via Deduction

There are many systems for logic-based abductive reasoning. However, many systems [22, 10, 25] other than [24] do not require minimality of explanation. Here we shall compare our method with those systems on matters other than minimality.

Pople [24] proposed the mechanization of abduction via deduction based on SL-resolution [17], with "synthesis" operation which corresponds to our skipping operation. Pople also incorporated the notion of simplicity as a criterion for selecting appropriate hypotheses. 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 is also used by Cox & Pietrzykowski [3]. It 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 2.7).

Finger [10] gives residue procedures for abductive reasoning where assumptions are restricted to only atoms. His "ordered residue" is similar to our linear method but is restricted to handle only Horn clauses. His "resolution residue" uses set-of-support resolution. As Finger mentions, the completeness result is a variation of Minicozzi & Reiter [20], and therefore excludes m.c.l. resolution. Thus, our linear method, which introduces more restriction strategies, is more

efficient than Finger's procedure.

Poole's Theorist [23] performs an incremental consistency checking. Instead of failing to prove the negation of an explanation, Theorist tries to prove the negation of each hypothesis in the explanation from the facts and from the previously assumed hypotheses as long as they are consistent with the facts Σ. Thus consistency checking is being done in the process of deducing new theorems. In our system, if $Carc(\Sigma, \mathcal{P})$ is computed, this checking is reduced to subsumption tests so that it can be embedded into deduction too (see Proposition 4.4).

6.3 Query Answering for Circumscription

Przymusinski [25] 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. Inoue & Helft [13] discuss this relationship in detail and how the efficiency of MILO-resolution can be improved. Every query answering procedure for circumscriptive theories is based on the following proposition:

Proposition 6.1 [25, 11, 13] 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 6.1, 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, as there is a set smaller than the whole set that can be used to answer a query. 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[11]. We call a set $S1$ verifying this condition a *precursor* of the production. Note that a clause in a precursor is not always a prime implicate of Σ. 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 2.7), 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**[12]. Thus a precursor-finding algorithm can be written by ordering two rules in Step 5a in the reverse order of the criterion that "hypothesizes whatever cannot be proven", as follows:

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 6.2 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 $\Sigma \cup \{S\} \models T$.

[11] 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 [12].

[12] Since Ginsberg's circumscriptive theorem prover [11] is based on a backward-chaining "plain ATMS", it may produce more clauses than MILO-resolution. For more detailed discussion, see Inoue & Helft [13].

Proof: Let $D_0, D_1, \cdots, D_n$ be an m.c.l.s. deduction of T from $\Sigma + C$ and $\mathcal{P}$. Let l_i be the first literal of $\vec{Q}_i$, where $D_i = \langle P_i, \vec{Q}_i \rangle$ and $0 \leq i \leq n-1$. Firstly, if **Skip** is applied for every l_j $(0 \leq j \leq n-1)$ such that $l_j \in L_{\mathcal{P}}$, then T is actually derived from $\Sigma + C$ and $\mathcal{P}$ by using the **Skip & Cut** rule, and of course $\Sigma \cup \{T\} \models T$ holds.

Next, suppose that $\exists D_j$ in the m.c.l.s. deduction such that $l_j \in L_{\mathcal{P}}$ but that **Resolve** is applied upon l_j in $\vec{Q}_j$ with a clause $B_j \in \Sigma$. Let m $(1 \leq m \leq n)$ be the number of such clauses, and D_k be such a clause where k $(0 \leq k \leq n-1)$ is the largest number. In this case, $D_{k+1} = \langle P_{k+1}, \vec{Q}_{k+1} \rangle$, where $P_{k+1} = P_k$ and $R_{k+1} = (B_k - \{\neg l_k\}) \cup (Q_k - \{l_k\})$. In the following proof, to simplify the discussion, we assume that there are no identical, truncated, or reduced literals in R_{k+1}; if they exist, then we can modify the proof appropriately. Now, let U be a clause m.c.l.s. derived from $\Sigma + (B_k - \{\neg l_k\})$ and $\mathcal{P}$, V a clause m.c.l.s. derived from $\Sigma + (Q_k - \{l_k\})$ and $\mathcal{P}$. Here, we can choose such U and V to satisfy $T = P_k \cup U \cup V$, because T is m.c.l.s. derived from $\Sigma + (P_{k+1} \cup R_{k+1})$ and $\mathcal{P}$.

Now assume that instead of applying **Resolve**, **Skip & Cut** is applied to D_k deducing $D'_{k+1} = \langle P'_{k+1}, \vec{Q'_{k+1}} \rangle$, where $P'_{k+1} = P_k \cup \{l_k\}$ and $R'_{k+1} = Q_k - \{l_k\}$. Then, $P_k \cup \{l_k\} \cup V$ is m.c.l.s. derived from $\Sigma + (P'_{k+1} \cup R'_{k+1})$ and $\mathcal{P}$, and thus from $\Sigma + C$ and $\mathcal{P}$. Since $\Sigma \cup \{l_k\} \models B_k - \{\neg l_k\}$, $\Sigma \cup \{l_k\} \models U$ holds, and thus $\Sigma \cup \{(P_k \cup \{l_k\} \cup V)\} \models T$ holds.

Now let $T_0 = T$ and $T_1 = (P_k \cup \{l_k\} \cup V)$. In the similar way, we can find an m.c.l.s. deduction of T_2 from $\Sigma + C$ and $\mathcal{P}$ such that $\Sigma \cup \{T_2\} \models T_1$, by resetting k to the second largest number. By using the bottom-up manner, we can successively find clauses T_j $(1 \leq j \leq m)$ m.c.l.s. derived from $\Sigma + C$ and $\mathcal{P}$ such that $\Sigma \cup \{T_j\} \models T_{j-1}$. Therefore, $\Sigma \cup \{T_m\} \models T_{m-1}$, $\Sigma \cup \{T_{m-1}\} \models T_{m-2}$, $\cdots$, $\Sigma \cup \{T_1\} \models T_0$. Hence, $\Sigma \cup \{T_m\} \models T_0$, and we get the theorem. □

Therefore, if we need only a precursor, instead of computing all the *Newcare*, we can use the above modification of m.c.l.s. resolution. This modified procedure can be used in 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 [27] of Σ (see Section 5).

7 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 [28], 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 [10], reordering subgoal trees [29], and others.

We have also analyzed both the interpreted and compiled approaches to the CMS and an extended ATMS. To find conditions in which the **Resolve** rule is replaced by the **Bypass** rule in an extended ATMS with maximum efficiency but without loss of completeness for general form of theories is one of the goals for future discussion.

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 [29] work. I would like to thank David Poole, Mark Stickel, Kurt Konolige, Wolfgang Bibel and Koichi Furukawa for helpful discussions on this topic.

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] Cox, P. T. and Pietrzykowski, T., "Causes for Events: Their Computation and Applications", *Proc. 8th Conf. on Automated Deduction*, Lecture Notes in Computer Science 230, Springer-Verlag (1986), pp.608–621.

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

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

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

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

[8] Doyle, J., "The Ins and Outs of Reason Maintenance", *Proc. IJCAI-83* (1983), pp.349–351.

[9] 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.

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

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

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

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

[14] Junker, U., "A Correct Non-Monotonic ATMS", *Proc. IJCAI-89* (1989), pp.1049–1054.

[15] 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.

[16] Kowalski, R. A. and Hayes, P. J., "Semantic Trees in Automatic Theorem-proving", in: B. Meltzer and D. Michie (Eds.), *Machine Intelligence* **4** (Edinburgh University Press, 1969), pp.87 101.

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

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

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

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

[21] Minker, J., "On Indefinite Databases and the Closed World Assumption", *Proc. 6th Conf. on Automated Deduction*, Lecture Notes in Computer Science 138, Springer-Verlag (1986), pp.292–308.

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

[23] Poole, D., "Explanation and Prediction: An Architecture for Default and Abductive Reasoning", *Computational Intelligence* **5** (1989), pp.97–110.

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

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

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

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

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

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

[30] 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.

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