Consequence-Finding Based on Ordered Linear Resolution

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

Abstract

Since linear resolution with clause ordering is incomplete for consequence-finding, it has been used mainly for proof-finding. In this paper, we re-evaluate consequence-finding. Firstly, consequence-finding is generalized to the problem in which only interesting clauses having a certain property (called characteristic clauses) should be found. Then, we show how adding a skip rule to ordered linear resolution makes it complete for consequence-finding in this general sense. Compared with set-of-support resolution, the proposed method generates fewer clauses to find such a subset of consequences. In the propositional case, this is an elegant tool for computing the prime implicants/implicates. The importance of the results lies in their applicability to a wide class of AI problems including procedures for nonmonotonic and abductive reasoning and truth maintenance systems.

1 Introduction

It is well known in automated deduction that while resolution [Robinson, 1965] is complete for proof-finding (called *refutation* complete), that is, it can deduce *false* from every unsatisfiable set of formulas, it is not deductively complete for finding every logical consequence of a satisfiable set of formulas. Lee [1967] addresses himself to this problem and defines the *consequence-finding* problem, which is expressed in the following form:

> Given a set of formulas *T* and a resolution procedure *P,* for any logical consequence *D* of T, can *P* derive a logical consequence *C* of *T* such that *C* subsumes *D?*

If a resolution procedure is complete for consequence-finding, then it is useful in spite of lacking deductive completeness because in general the logical consequences not deducible from the theory are neither interesting nor useful. Namely, such a formula is subsumed by some formula deducible from the theory and thus it is weak.

Historically, consequence-finding had been investigated intensively since the resolution principle [Robinson, 1965] was invented for proof-finding. Lee's [1967] completeness theorem was proved for the original resolution principle. Slagle, Chang and Lee [1969] extended the result to various kinds of semantic resolution. However, after Minicozzi and Reiter [1972] extended these results to various linear resolution strategies in the early 70s, consequence-finding was once abandoned in research of automated theorem proving and attention has been directed towards only proof-finding [1]. It appears that there are three reasons for this discouragement:

1. The results presented by [Minicozzi and Reiter, 1972] are in some sense negative. Linear resolution involving C-ordering [Loveland, 1978; Reiter, 1971; Kowalski and Kuhner, 1971; Chang and Lee, 1973; Shostak, 1976] (literals are ordered in each clause in the theory), which is the most familiar and efficient class of resolution procedures, is incomplete for consequence-finding. Thus, the completeness result that we would most like to obtain does not hold.
2. It is neither practical nor useful to find all of the consequences in general. However, there has not been an intellectual method which directly searches interesting formulas, instead of getting all theorems and then filtering them by some criteria.
3. As opposed to proof-finding, consequence-finding has lacked useful applications in AI.

In this paper, we re-evaluate consequence-finding and give new perspectives. The proposals are motivated and justified by the following solutions to the above three problems:

1. Recently, Finger [1987] gave a complete procedure based on set-of-support deduction for generating formulas (called *ramification)* derivable from a theory and a newly added formula as an initial set of support. We provide a complete procedure for consequence-finding, which contains more restriction strategies than Finger's, by adding one rule called *skip* operation to C-ordered linear resolution.
2. Bos8u and Siegel [1985] give a complete algorithm for finding the set of positive clauses derivable from a groundable theory (called *characteristic clauses).*

[1] One can see that textbooks of resolution-based theorem proving, such as [Chang and Lee, 1973; Loveland, 1978], have no sections for consequence-finding.

Recently, Siegel [1987] redefined the notion of characteristic clauses for propositional theories and proposed a complete algorithm for finding them. We show how our results can both improve the efficiency of Bossu and Siegel's algorithm and lift Siegel's for the general case. Moreover, easy modifications of the proposed procedure can be shown to be applied to more efficient variations of consequence-finding.

3. Przymusinski [1989] defines MILO-resolution to be used in his query answering procedure for *circumscription* of ground theories. MILO-resolution can be characterized as C-ordered linear resolution with skip operation [inoue and Helft, 1990]. On the other hand, most procedures for *abduction* [Pople, 1973; Cox and Pietrzykowski, 1986; Finger, 1987; Poole, 1989; Stickel, 1990] can utilize consequence-finding procedures to generate explanations [inoue, 1990]. We show how the proposed procedure can be applied to generate such interesting formulas for nonmonotonic and abductive reasoning. In the propositional case, the technique can be viewed as an elegant algorithm to compute *prime implicants/implicates,* and thus can be utilized for the *clause management system* [Reiter and de Kleer, 1987] that is a generalization of the ATMS [de Kleer, 1986].

The importance of the results presented lies in their applicability to a wide class of AI problems. In other words, the methods shed some light on better understanding and implementation of many AI techniques.

The present paper is organized as follows. The next section characterizes consequence-finding in a general way, and shows how various AI problems can be well defined by using this notion of characteristic clauses. Section 3 presents the basic procedure, which is sound and complete for characteristic-clause-finding, based on C-ordered linear resolution. Efficient but incomplete variations of the basic procedure and their properties are provided in Section 4. Because of space limitation, proofs of propositions are given in the full paper.

2 Characterizing Consequence-Finding

We define a *theory* as a set of clauses, which can be identified with a *conjunctive normal form (CNF) formula.* A *clause* is a disjunction (possibly written as a set) of *literals,* each of which is a possibly negated atomic formula. Each variable in a clause is assumed to be universally quantified. For a method converting a formula to this form of theory, see [Loveland, 1978). If 5 is a set of clauses, by *S* we mean the set formed by taking the negation of each clause in *S*. The empty clause is denoted by $\square$. A clause C is said to ***subsume*** a clause D if there is a substitution θ such that $C\theta \subseteq D$ and C has no more literals than D [2]. 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. A clause C is a ***theorem***, or a ***(logical) consequence*** of Σ if $\Sigma \models C$. The set of theorems of Σ is denoted by $Th(\Sigma)$.

[2]This definition of subsumption is called *$-subaumption* in [Loveland, 1978]. Unlike in the propositional case, the second condition is necessary because a clause implies its factor.

2.1 Characteristic Clauses

We use the notion of *characteristic clauses,* which is a generalized notion of logical consequences and helps to analyze computational aspects of many of AI problems. The idea of characteristic clauses was introduced by Bossu and Siegel [1985] for evaluating a kind of closed-world reasoning and was later redefined by Siegel [1987] for propositional logic. The description below is more general than [Bossu and Siegel, 1985; Siegel, 1987; Inoue, 1990] in the sense that the notion is not limited to some special purposes and that it deals with the general case instead of just the propositional cases. Also, these notions are independent of implementation; we do not assume any particular resolution procedure in this section. 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.

Definition 2.1 (1) We denote by $\mathcal{R}$ the set of all predicate symbols in the language. For $R \subseteq \mathcal{R}$, we denote by R^+ (R^-) the positive (negative) occurrences of predicates from R in the language. The set of all atomic formulas is denoted as $\mathcal{A} = \mathcal{R}^+$, and the set of literals is denoted $\mathcal{L} = \mathcal{A} \cup \overline{\mathcal{A}} = \mathcal{R}^+ \cup \mathcal{R}^-$.
(2) 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 $\mathcal{L}$ and $Cond$ is a certain condition to be satisfied. When $Cond$ is not specified, $\mathcal{P}$ is just denoted as $\langle L_{\mathcal{P}} \rangle$. The production field $\langle \mathcal{L} \rangle$ is denoted $\mathcal{P}_{\mathcal{L}}$.
(3) 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 theorems of Σ belonging to $\mathcal{P}$ is denoted by $Th_{\mathcal{P}}(\Sigma)$.
(4) A production field $\mathcal{P}$ is *stable* if for any two clauses C and D such that C subsumes D, it holds that if D belongs to $\mathcal{P}$, then C also belongs to $\mathcal{P}$.

Example 2.2 Examples of stable production fields.
(1) $\mathcal{P}_1 = \mathcal{P}_{\mathcal{L}}$: $Th_{\mathcal{P}_1}(\Sigma)$ is equivalent to $Th(\Sigma)$.
(2) $\mathcal{P}_2 = \langle \mathcal{A} \rangle$:
$Th_{\mathcal{P}_2}(\Sigma)$ is the set of positive clauses implied by Σ.
(3) $\mathcal{P}_3 = \langle \overline{A},$ *size is less than* $k \rangle$ where $A \subseteq \mathcal{A}$:
$Th_{\mathcal{P}_3}(\Sigma)$ is the set of negative clauses implied by Σ containing less than k literals all of which belong to $\overline{A}$.

Example 2.3 $\mathcal{P}_4 = \langle \mathcal{A},$ *size is more than* $k \rangle$ is not stable. For example, if $k = 2$ and $p(a), q(b), r(c) \in \mathcal{A}$, then $D_1 = p(a) \vee q(b)$ subsumes $D_2 = p(a) \vee q(b) \vee r(c)$, and D_2 belongs to $\mathcal{P}_4$ while D_1 does not.

Definition 2.4 (Characteristic Clauses) Let Σ be a set of clauses, and $\mathcal{P}$ a production field. The *characteristic clauses of* Σ *with respect to* $\mathcal{P}$ are:

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

$Carc(\Sigma, \mathcal{P})$ contains all the unsubsumed theorems of Σ belonging to a production field $\mathcal{P}$. To see why this notion is a generalization of consequence-finding, let $\mathcal{P}$ be $\mathcal{P}_{\mathcal{L}}$. From the definition of consequence-finding, for any clause $D \in Th(\Sigma)$, a complete procedure P can derive a clause $C \in Th(\Sigma)$ such that C subsumes D. Therefore, P can derive every clause $C' \in \mu Th(\Sigma)$ because

C' is not subsumed by any other theorem of Σ. Hence, $Carc(\Sigma, \mathcal{P}_{\mathcal{L}}) = \mu Th(\Sigma)$ must be contained in the theorems derivable from Σ by using P. Note also that the empty clause belongs to every stable production field $\mathcal{P}$, and that if Σ is unsatisfiable, then $Carc(\Sigma, \mathcal{P})$ contains only $\square$. This means that proof-finding is a special case of consequence-finding. Next is a summarizing proposition.

Proposition 2.5 Let Σ be a theory, $\mathcal{P}$ a stable production field. A clause D is a theorem of Σ belonging to $\mathcal{P}$ if and only if there is a clause C in $Carc(\Sigma, \mathcal{P})$ such that C subsumes D. In particular, Σ is unsatisfiable if and only if $Carc(\Sigma, \mathcal{P}) = \{\square\}$.

As we will see later, when new information is added to the theory, it is often necessary to compute newly derivable consequences caused by this new information. For this purpose, consequence-finding is extended to look for such a ramification of new information.

Definition 2.6 (New Characteristic Clauses) Let Σ be a set of clauses, $\mathcal{P}$ a production field, and F a formula. The *new characteristic clauses of F with respect to Σ and $\mathcal{P}$* are:

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

In other words, $C \in Newcarc(\Sigma, F, \mathcal{P})$ if: (i) $\Sigma \cup \{F\} \models C$, (ii) C belongs to $\mathcal{P}$, (iii) $\Sigma \not\models C$, and (iv) no other clause subsuming C satisfies (i)–(iii).

The next three propositions show the connections between the characteristic clauses and the new characteristic clauses. Firstly, $Newcarc(\Sigma, F, \mathcal{P})$ can be represented by the set difference of two sets of characteristic clauses.

Proposition 2.7

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

When F is a CNF formula, $Newcarc(\Sigma, F, \mathcal{P})$ can be decomposed into a series of *primitive Newcarc operations* each of whose added formula is just a single clause.

Proposition 2.8 Let $F = C_1 \wedge \cdots \wedge C_m$ be a CNF formula. Then,

$$Newcarc(\Sigma, F, \mathcal{P}) = \mu\,[\,\bigcup_{i=1}^{m} Newcarc(\Sigma_i, C_i, \mathcal{P})\,]$$

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

Finally, the characteristic clauses $Carc(\Sigma, \mathcal{P})$ can be expressed by constructively using primitive $Newcarc$ operations. Notice that for any atomic formula 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 $Carc(\Sigma, \mathcal{P})$.

Proposition 2.9 Let $\Sigma_m = \{\,C_1, \cdots, C_m\,\}$. Then,

$Carc(\emptyset, \mathcal{P}) = \{\,p \vee \neg p \mid p \in \mathcal{A}$ and $p \vee \neg p$ belongs to $\mathcal{P}\,\}$,

$Carc(\Sigma_{i+1}, \mathcal{P}) = \mu\,[Carc(\Sigma_i, \mathcal{P}) \cup Newcarc(\Sigma_i, C_i, \mathcal{P})]$,

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

2.2 Applications

We illustrate how the use of the (new) characteristic clauses enables elegant definition and precise understanding of many AI problems.

2.2.1 Propositional Case

In the propositional case, $\mathcal{A}$ is reduced to the set of propositional symbols in the language. The subsumption relation is now very simple: a clause C subsumes D if $C \subseteq D$. A theorem of Σ is called an *implicate* of Σ, and the *prime implicates* [Kean and Tsiknis, 1990] of Σ are:

$$PI(\Sigma) = \mu Th(\Sigma)\,.$$

The characteristic clauses of Σ with respect to $\mathcal{P}$ are the prime implicates of Σ belonging to $\mathcal{P}$. When $\mathcal{P} = \mathcal{P}_{\mathcal{L}}$, it holds that $Carc(\Sigma, \mathcal{P}) = PI(\Sigma)$ [3].

Computing prime implicates is an essential task in the ATMS [de Kleer, 1986] and in its generalization called the *clause management system* (CMS) [Reiter and de Kleer, 1987]. The CMS is responsible for finding minimal supports for the queries:

Definition 2.10 [Reiter and de Kleer, 1987] Let Σ be a set of clauses and C a clause. A clause S is a *support for C with respect to* Σ if: (i) $\Sigma \models S \cup C$, and (ii) $\Sigma \not\models S$. A support S for C with respect to Σ is *minimal* if no other support S' for C subsumes S. The set of minimal supports for C with respect to Σ is written $MS(\Sigma, C)$.

The above definition can be easily extended to handle any formula instead of a clause as a query. Setting the production field to $\mathcal{P}_{\mathcal{L}}$ we see that:

Proposition 2.11 [Inoue, 1990] Let F be any formula.

$$MS(\Sigma, F) = Newcarc(\Sigma, \neg F, \mathcal{P}_{\mathcal{L}}).$$

When we choose the primitive $Newcarc$ operation as a basic computational task, the above proposition does not require computation of $PI(\Sigma)$. On the other hand, the *compiled* approach [Reiter and de Kleer, 1987] takes $PI(\Sigma)$ as input to find $MS(\Sigma, C)$ for any clause C easily:

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

In the ATMS [de Kleer, 1986], there is a distinguished set of *assumptions* $A \subseteq \mathcal{L}$. An ATMS can be defined as a system responsible for finding the negations of all minimal supports for the queries consisting of only literals from A [Reiter and de Kleer, 1987; Inoue, 1990]. Therefore, the ATMS *label* of a formula F relative to a given theory Σ and A is characterized as

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

Inoue [1990] gives various sound and complete methods for both generating and updating the labels of queries relative to a non-Horn theory and literal assumptions.

2.2.2 Abductive and Nonmonotonic Reasoning

The task of the CMS/ATMS can be viewed as propositional *abduction* [Reiter and de Kleer, 1987; Levesque, 1989; Inoue, 1990]. For general cases, there are many proposals for a logical account of abduction [Pople, 1973; Cox and Pietrzykowski, 1986; Finger, 1987; Poole, 1989; Stickel, 1990], whose task is defined as generation of explanations of a query.

[3] The *prime implicants* of a disjunctive normal form formula can be defined in the same manner if the duality of $\wedge$ and $\vee$ is taken into account.

Definition 2.12 Let Σ be a theory, $H \subseteq \mathcal{L}$ (called the *hypotheses*), and G a closed formula. A conjunction E of ground instances of H is an *explanation of* G *from* (Σ, H) if: (i) $\Sigma \cup \{E\} \models G$ and (ii) $\Sigma \cup \{E\}$ is satisfiable[4].
An explanation E of G is *minimal* if no proper subconjunction E' of E satisfies $\Sigma \cup \{E'\} \models G$.
An *extension of* (Σ, H) is the set of logical consequences of $T \cup \{M\}$ where M is a maximal conjunction of ground instances of H such that $T \cup \{M\}$ is satisfiable.

Abduction can be characterized as follows:

Proposition 2.13 [Inoue and Helft, 1990] The set of all minimal explanations of G from (Σ, H) is

$$\overline{Newcarc(\Sigma, \neg G, \mathcal{P})}, \quad \text{where } \mathcal{P} = \langle \overline{H} \rangle .$$

There is an extension of (Σ, H) in which G holds iff

$$Newcarc(\Sigma, \neg G, \mathcal{P}) \neq \emptyset, \quad \text{where } \mathcal{P} = \langle \overline{H} \rangle .$$

Another important problem is to predict formulas that hold in all extensions. This problem is known to be equivalent to *circumscription* under the unique-names and domain-closure assumptions. Proving a formula holds in a circumscriptive theory [Przymusinski, 1989; Ginsberg, 1989], as well as other proof methods for nonmonotonic reasoning formalisms (including explanation-based argument systems [Poole, 1989] and variations of closed-world assumptions [Bossu and Siegel, 1985; Minker and Rajasekar, 1990]), are based on finding explanations of the query, and showing that these explanations cannot be refuted:

Proposition 2.14 [Inoue and Helft, 1990] Suppose that $L_{\mathcal{P}} = P^+ \cup Q^+ \cup Q^-$, where P is the minimized predicates and Q is the fixed predicates in circumscription policy and that $\mathcal{P} = \langle L_{\mathcal{P}} \rangle$. Every circumscriptive minimal model satisfies a formula F if and only if there is a conjunction G of clauses from $Th_{\mathcal{P}}(\Sigma \cup \{\neg F\})$ such that $Newcarc(\Sigma, \neg G, \mathcal{P}) = \emptyset$.

Since we have characterized the prime implicates, the CMS/ATMS, abduction and circumscription [5], any application area of these techniques can be directly characterized by using the notion of the (new) characteristic clauses: for instance, diagnosis, synthesis [Finger, 1987] (plan recognition, prediction, design), and natural language understanding [Stickel, 1990].

3 Ordered-Linear Resolution for Consequence-Finding

We now present the basic procedure for implementing the primitive *Newcarc* operation. The important feature of the procedure is that it is *direct,* namely it is both sensitive to the given added clause to the theory and restricted to searching only characteristic clauses.

[4]This definition is based on [Poole, 1989] and deals with *ground* explanations. To get universally quantified explanations, we need to apply the *reverse Skolemization* algorithm [Cox and Pietrzykowski, 1986].

[5] When a query in abduction or circumscription contains existentially quantified variables, it is sometimes desirable to know for what values of these variables the query holds. This *answer extraction* problem is considered in [Helft *et* al., 1991].

3.1 Basic Procedure

Given a theory Σ, a stable production field *V* and a clause C, we show how *Newcarc*(Σ, *C,V)* can be computed by extending *C-ordered linear resolution* [6]. As seen in Propositions 2.8 and 2.9, both *Newcarc*(Σ,*F, V)* for a CNF-formula *F* and *Carc*(Σ, *V)* can be computed by using this primitive *Newcarc* operation. There are two reasons why C-ordered linear resolution is useful for computing the new characteristic clauses:

1. A newly added single clause *C* can be treated as the *top clause* of a linear deduction. This is a desirable feature for consequence-finding since the procedure can directly derive the theorems relevant to the added information.
2. It is easy to achieve the requirement that the procedure should focus on producing only those theorems that belong to *V.* This is implemented by allowing the procedure to *skip* the selected literals belonging to *V.* The computational superiority of the proposed technique compared to set-of-support resolution that is used by Finger's resolution residue [Finger, 1987], apart from the fact that C-ordered linear resolution contains more restriction strategies in natural ways, comes from this relevancy notion of directing search to *V.*

Some procedures are known to perform this computation for restricted theories [7]. For propositional theories, Siegel [1987] proposes a complete algorithm by extending SL-resolution [Kowalski and Kuhner, 1971]. Inoue and Helft [1990] point out that MILO-resolution [Przymusinski, 1989], an extension of OL-resolution [Chang and Lee, 1973], can be viewed as C-ordered linear resolution with skip operation for ground theories with a particular production field for circumscription (see Proposition 2.14).

The following proposed procedure called *SOL (Skipping Ordered Linear) resolution* is a kind of generalization of [Przymusinski, 1989; Siegel, 1987]. An *ordered* clause is a sequence of literals possibly containing *framed literals* [Chang and Lee, 1973] which represents literals that have been resolved upon: from a clause *C* an ordered clause *C* is obtained just by ordering the elements of *C;* conversely, from an ordered clause *C* a clause *C* is obtained by removing the framed literals and converting the remainder to the set. A *structured* clause *(P, Q)* is a pair of a clause *P* and an ordered clause *Q,* whose clausal meaning is *P* U *Q.*

[6]By the term C-ordered linear resolution, we mean the family of linear resolution using ordered clauses and the information of literals resolved upon. Examples of C-ordered linear resolution are Model Elimination [Loveland, 1978], m.c.l. resolution [Reiter, 1971], SL-resolution [Kowalski and Kuhner, 197l], OL-resolution [Chang and Lee, 1973], and the GC procedure [Shostak, 1976]. This family is one of the most familiar and efficient classes of resolution for non-Horn theories because it contains several restriction strategies.

[7]Bossu and Siegel's [1985] saturation procedure finds Carc(E, *V)* where *Lv* are fixed to ground atoms. However, it does not use C-ordering, but A-ordering (a total ordering of all the ground atomic formulas).

Definition 3.1 Given a theory Σ, a clause C, and a production field $\mathcal{P}$, an *SOL-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, $j < i$. This rule is not applied if D_i is generated from D_{i-1} by applying 5(a)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 *left-most* literal of $\vec{Q_i}$. P_{i+1} and $\vec{R_{i+1}}$ are obtained by applying either of the rules:
      i. **(Skip)** If $P_i \cup \{l\}$ belongs to $\mathcal{P}$, 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)** If there is a clause B_i in Σ such that $\neg k \in B_i$ and l and k are unifiable with mgu θ, then $P_{i+1} = P_i\theta$ and $\vec{R_{i+1}}$ is an ordered clause obtained by concatenating $\vec{B_i}\theta$ and $\vec{Q_i}\theta$, framing $l\theta$, and removing $\neg k\theta$.
      iii. **(Reduce)** If either
         A. P_i or $\vec{Q_i}$ contains an unframed literal k different from l (*factoring*), or
         B. $\vec{Q_i}$ contains a framed literal $\boxed{\neg k}$ (*ancestry*),
         and l and k are unifiable with mgu θ, then $P_{i+1} = P_i\theta$ and $\vec{R_{i+1}}$ is obtained from $\vec{Q_i}\theta$ by deleting $l\theta$.
   (b) $\vec{Q_{i+1}}$ is obtained from $\vec{R_{i+1}}$ by deleting every framed literal not preceded by an unframed literal in the remainder (*truncation*).

Remarks. (1) At Rule 5a, we can choose the *selected literal* l with more liberty like SL-resolution or SLI-resolution [Minker and Rajasekar, 1990].

(2) Rule 4 is included for efficiency. This is overlooked in OL-deduction (and so is in MILO-resolution), but is present in Model Elimination [Loveland, 1978].

(3) When $\mathcal{P}$ is in the form of $\langle L_{\mathcal{P}} \rangle$, factoring (5(a)iiiA) can be omitted in intermediate deduction steps like Weak Model Elimination [Loveland, 1978]. In this case, Rules 3 and 4 are omitted, and factoring is performed at the final step, namely it is taken into account only for P_i in a structured clause of the form $\langle P_i, \Box \rangle$.

(4) The selection of rules 5(a)i, 5(a)ii and 5(a)iii must be non-deterministic; for $l \in L_{\mathcal{P}}$ any rule may be applied. This is not a straightforward generalization of MILO-resolution or Siegel's algorithm, because they do not deal with **Reduce** as an alternative choice of other two rules, but make $\vec{Q_{i+1}}$ as the reduced ordered clause of the ordered factor of $\vec{R_{i+1}}$ that is obtained by **Skip** or **Resolve**. Both Przymusinski and Siegel claim that the lifting lemma should work for their procedures. Unfortunately, this simpler treatment violates the completeness described below. Furthermore, even for proof-finding, OL-resolution, which also handles the ancestry rule as a subsequent rule of **Resolve**, is incomplete. For example, when the theory is given as

$$\Sigma = \{ \quad p(a) \vee p(x) \vee \neg q(x), \quad (1) \qquad \neg p(b), \quad (2) \qquad q(b) \quad (3) \ \},$$

it is easy to see that $\Sigma \models p(a)$. However, there is no OL-refutation from $\Sigma + \neg p(a)$:

(4) $\underline{\neg p(a)}$ given top clause
(5) $\underline{p(x)} \vee \neg q(x) \vee \boxed{\neg p(a)}$ resolution with (1)
(6) $\underline{\neg q(a)} \vee \boxed{\neg p(a)}$ reduction

Here, each underlined literal denotes a selected literal in the next step. The clause (6) is the dead-end of the OL-deduction. Hence, the reduction rule must be an alternative rule. Model Elimination and SL-resolution deal with the reduction rule as an alternative.

The **Skip** rule (5(a)i) reflects the following operational interpretation of a *stable* production field $\mathcal{P}$: by Definition 2.1 (4), if a clause C does not belong to $\mathcal{P}$ and a clause D is subsumed by C, then D does not belong to $\mathcal{P}$ either. That is why 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 3.2 (Soundness and Completeness) (1) If a clause S is derived using an SOL-deduction from $\Sigma + C$ and $\mathcal{P}$, then S belongs to $Th_{\mathcal{P}}(\Sigma \cup \{C\})$.
(2) 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 SOL-deduction of a clause S from $\Sigma + C$ and $\mathcal{P}$ such that S subsumes T.

Recall that C-ordered linear resolution is refutation-complete as shown, for example, by [Loveland, 1978], but is incomplete for consequence-finding [Minicozzi and Reiter, 1972]. Theorem 3.2 (2) says that SOL-resolution is complete for characteristic-clause-finding, and thus complete for consequence-finding when $\mathcal{P}$ is $\mathcal{P}_{\mathcal{L}}$.

Example 3.3 Consider the theory Σ and the clause C:

$$\Sigma = \{ \quad \neg c \vee \neg a \quad (1), \qquad \neg c \vee \neg b \quad (2) \ \}, \qquad C = \quad a \vee b .$$

There is no OL-deduction of $\neg c$ from $\Sigma + C$, but $\neg c$ is derived using an SOL-deduction from $\Sigma + C$ and $\mathcal{P}_{\mathcal{L}}$ as:

(3) $\langle \Box, \underline{a} \vee b \rangle$, given top clause
(4) $\langle \Box, \underline{\neg c} \vee \boxed{a} \vee b \rangle$, resolution with (1)
(5) $\langle \neg c, \boxed{a} \vee \underline{b} \rangle$, skip and truncation
(6) $\langle \neg c, \underline{\neg c} \vee \boxed{b} \rangle$, resolution with (2)
(7) $\langle \neg c, \boxed{b} \rangle$. factoring and truncation

Note that an OL-deduction stops at (4).

Definition 3.4 Let $\Delta(\Sigma, C, \mathcal{P})$ be the set of clauses derived using all SOL-deductions from $\Sigma + C$ and $\mathcal{P}$. The *production from $\Sigma + C$ and $\mathcal{P}$* is:

$$Prod(\Sigma, C, \mathcal{P}) = \mu\, \Delta(\Sigma, C, \mathcal{P}).$$

Theorem 3.5 Let C be a clause.

$$Newcarc(\Sigma, C, \mathcal{P}) = Prod(\Sigma, C, \mathcal{P}) - Th_{\mathcal{P}}(\Sigma).$$

Theorem 3.5 says the primitive $Newcarc(\Sigma, C, \mathcal{P})$ is contained in the production from $\Sigma + C$ and $\mathcal{P}$. To remove the clauses in the production derivable from Σ, we can use proof-finding: $\Sigma \models S$ iff there is an SOL-deduction of $\square$ from $\Sigma + \neg S$ and $\langle \emptyset \rangle$. However, when the characteristic literals $L_{\mathcal{P}}$ is small compared with the whole literals $\mathcal{L}$, the computation of $Carc(\Sigma, \mathcal{P})$ can be performed better as the search focuses on $\mathcal{P}$. Then, the check can be reduced to subsumption tests on $Carc(\Sigma, \mathcal{P})$ by Proposition 2.5. The role of $Carc(\Sigma, \mathcal{P})$ in this case is similar to the minimal *nogoods* in the ATMS [de Kleer, 1986]. This checking can be embedded into an SOL-deduction by adding the following rule:

4a. For each $D_i = \langle P_i, \vec{Q}_i \rangle$, P_i is not subsumed by any clause of $Carc(\Sigma, \mathcal{P})$.

Proposition 3.6 If Rule 4a is incorporated into SOL-deduction, then $Prod(\Sigma, C, \mathcal{P}) = Newcarc(\Sigma, C, \mathcal{P})$.

3.2 Computing Prime Implicates

If the given theory is propositional, the prime implicates can be constructed using every clause as a top clause:

Proposition 3.7 [Inoue, 1990] Given $PI(\Sigma)$ and a clause C, $PI(\Sigma \cup \{C\})$ can be found incrementally:

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

Notice that, in practice, no tautology will take part in any deduction; tautologies decrease monotonically. The computation of all prime implicates of E by Proposition 3.7 is much more efficient than the brute-force way of resolution proposed in [Reiter and de Kleer, 1987]. Also, ours uses C-ordered linear resolution and thus naturally has more restriction strategies than set-of-support resolution that is used in Kean and Tsiknis's [1990] extension of the consensus method.

This difference becomes larger when there are some distinguished literals representing assumptions in ATMS cases. The most important difference lies in the fact that the formulations by [Reiter and de Kleer, 1987; Kean and Tsiknis, 1990] require the computation of all prime implicates whereas ours only needs characteristic clauses that are a subset of the prime implicates [inoue, 1990].

4 Variations

In Step 5a of an SOL-deduction (Definition 3.1), we treat two rules Skip (Rule 5(a)i) and Resolve (Rule 5(a)ii) as alternatives in order to guarantee the completeness of SOL-resolution. In this section, we violate this requirement, and show efficient variations of SOL-resolution and their applications to AI problems. Note that Reduce (Rule 5(a)iii) still remains as an alternative choice of other two rules (see Remark (3) of Definition 3.1).

4.1 Preferring Resolution

The first variation, called *SOL-R deduction,* makes Resolve precede Skip, namely Skip is tried to be applied only when Resolve cannot be applied. In a special case of SOL-R deductions, where the production field is fixed to PL, Skip is always applied whenever Resolve cannot be applied for any selected literal in a deduction. In abduction, the resultant procedure in this case "hypothesizes whatever cannot be proven". This is also called *dead-end abduction,* which is first proposed by Pople [1973] in his abductive procedure based on SL-resolution [Kowalski and Kuhner, 1971] [8]. The criterion is also used by Cox and Pietrzykowski [1986].

4.2 Preferring Skip

The next variation, called *SOLS deduction,* places Skip and Resolve in the reverse order of SOL-R deductions. That is, when the selected literal belongs to *L-p,* only Skip is applied by ignoring the possibility of Resolve.

This skip-preference has the following nice properties. Firstly, this enables the procedure to prune the branch of the search tree that would have resulted from the literal being resolved upon. Secondly, SOL-S deductions are correct model-theoretically. Let us divide the set of clauses A produced by using SOL-deductions from $\Sigma + C$ and *V* into two sets, say Δ_1 and Δ_2, such that

$$\Delta = \Delta_1 \cup \Delta_2 \text{ and } \Sigma \cup \Delta_1 \models \Delta_2.$$

Note that $Prod(\Sigma, C, \mathcal{P}) = \mu\Delta$. Then adding Δ_2 to Δ_1 does not change the models of $\Sigma \cup \Delta_1$:

$$Mod(\Sigma \cup \Delta_1) = Mod(\Sigma \cup \Delta) = Mod(\Sigma \cup Prod(\Sigma, C, \mathcal{P})),$$

where $Mod(T)$ is the models of T. Thus only Δ_1 needs to be computed model-theoretically. The next theorem shows SOL-S deductions produce precisely such a Δ_1.

Theorem 4.1 If a clause T is derived by an SOL-deduction from $\Sigma + C$ and $\mathcal{P}$, then there is an SOL-S deduction of a clause S from $\Sigma + C$ and $\mathcal{P}$ such that $\Sigma \cup \{S\} \models T$.

In abduction, recall that for a clause $S \in \Delta$ and $H = \overline{L_{\mathcal{P}}}$, $\neg S$ is an explanation of $\neg C$ from (Σ, H) if $\Sigma \not\models S$. Thus, an explanation in $\overline{\Delta_1}$ is the weakest in the sense that for any clause $S_2 \in \Delta_2$, there exists a clause $S_1 \in \Delta_1$ such that $\Sigma \cup \{\neg S_2\} \models \neg S_1$ holds [9].

In circumscription, this is particularly desirable since we want to answer whether a query holds in every minimal model or not; the purpose of using explanation-based procedures is purely model-theoretic. One of advantages of Przymusinski's [1989] procedure lies in the fact that MILO-resolution performs a kind of SOL-S deductions [Inoue and Helft, 1990].

4.3 Between Skipping and Resolving

One further generalization of this kind of preference would lead us to *best-first* abduction. Stickel [1990] uses the minimal-cost proof where we can choose each operation whose expected computational cost is minimum, but it is difficult to apply the idea to non-Horn theories.

---

[8] Pople's *synthesis* operation performs "factor-and-skip".

[9] An explanation E_1 is said to be *less-presumptive than* E_2 if $\Sigma \cup \{E_2\} \models E_1$ [Poole, 1989]. Therefore, an explanation in $\overline{\Delta_1}$ is a least-presumptive explanation of $\neg C$ from (Σ, H).

5 Conclusion

We have revealed the importance of consequence-finding in AI techniques. Most advanced reasoning mechanisms such as abduction and default reasoning require global search in their proof procedures. This global character is strongly dependent on consequence-finding, in particular those theorems of the theory belonging to production fields. That is why we need some complete procedure for consequence-finding. For this purpose, we have proposed SOL-resolution, an extension of C-ordered linear resolution augmented by the skip rule. The procedure is sound and complete for finding the (new) characteristic clauses. The significant innovation of the results presented is that the procedure is direct relative to the given production field. We have also presented incomplete, but efficient variations of the basic procedures with nice properties.

Acknowledgment

1 especially wish to thank Mark Stickel for his valuable comments on SOL-resolution and Nicolas Helft for discussion on earlier work. I would also like to thank Ray Reiter, Ryuzo Hasegawa, Koichi Furukawa and Wolfgang Bibel for helpful suggestions on this topic, and Kazuhiro Fuchi for giving me the opportunity to do this work.

References

[Bossu and Siegel, 1985] Genevieve Bossu and Pierre Siegel. Saturation, nonmonotonic reasoning, and the closed-world assumption. *Artificial Intelligence,* 25: 23-67, 1985.

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

[Cox and Pietrzykowski, 1986] P.T. Cox and T. Pietrzykowski. Causes for events: their computation and applications. In *Proceedings of the Eighth International Conference on Automated Deduction,* Lecture Notes in Computer Science 230, pages 608-621, Springer-Verlag, 1986.

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

[Finger, 1987] Joseph J. Finger. Exploiting constraints in design synthesis. Technical Report STAN-CS-88-1204, Department of Computer Science, Stanford University, Stanford, CA, April 1987.

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

[Helft *et al,* 1991] Nicolas Helft, Katsumi Inoue and David Poole. Query answering in circumscription. In *Proc. of the 12th IJCAI,* Sydney, Australia, 1991.

[Inoue, 1990] Katsumi Inoue. An abductive procedure for the CMS/ATMS. In: J.P. Martins and M. Reinfrank, editors, *Proceedings of the 1990 Workshop on Truth Maintenance Systems,* Lecture Notes in Artificial Intelligence, Springer-Verlag, 1991.

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

[Kean and Tsiknis, 1990] Alex Kean and George Tsiknis. An incremental method for generating prime implicants/implicates. *J. Symbolic Computation,* 9: 185—206, 1990.

[Kowalski and Kuhner, 1971] Robert Kowalski and Donald Kuhner. Linear resolution with selection function. *Artificial Intelligence,* 2: 227-260, 1971.

[Lee, 1967] Richard Char-Tung Lee. A completeness theorem and computer program for finding theorems derivable from given axioms. Ph.D. thesis, University of California, Berkeley, CA, 1967.

[Levesque, 1989] Hector J. Levesque. A knowledge-level account of abduction (preliminary version). In *Proc. of the 11th IJCAI,* pages 1061-1067, Detroit, MI, 1989.

[Loveland, 1978] Donald W. Loveland. *Automated Theorem Proving: A Logical Basis.* North-Holland, Amsterdam, 1978.

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

[Minker and Rajasekar, 1990] Jack Minker and Arcot Rajasekar. A fixpoint semantics for disjunctive logic programs. *J. Logic Programming,* 9: 45-74, 1990.

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

[Pople, 1973] Harry E. Pople, Jr. On the mechanization of abductive logic. In *Proc. of the 3rd IJCAI,* pages 147-152, Stanford, CA, 1973.

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

[Reiter, 1971] Raymond Reiter. Two results on ordering for resolution with merging and linear format. *J. ACM,* 18: 630-646, 1971.

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

[Robinson, 1965] J.A. Robinson. A machine-oriented logic based on the resolution principle. *J. ACM,* 12: 23-41, 1965.

[Shostak, 1976] Robert E. Shostak. Refutation Graphs. *Artificial Intelligence,* 7: 51-64, 1976.

[Siegel, 1987] Pierre Siegel. Representation et utilisation de la connaissance en calcul propositionnel. These d'Etat. Universite d'Aix-Marseille II, Luminy, 1987.

[Slagle *et al,* 1969] J.R. Slagle, C.L. Chang and R.C.T. Lee. Completeness theorems for semantic resolution in consequence-finding. In *Proc. of the IJCAI,* pages 281-285, Washington, D.C., 1969.

[Stickel, 1990] Mark E. Stickel. Rationale and methods for abductive reasoning in natural-language interpretation. In: R. Studer, editor, *Natural Language and Logic, Proceedings of the International Scientific Symposium,* pages 233-252, Lecture Notes in Artificial Intelligence 459, Springer-Verlag, 1990.