Chapter 3

Resolution

3.1 The Resolution Principle

The resolution principle due to Robinson is a cornerstone for mechanical theorem proving. This principle deals with the issue of demonstrating the validity of a proposition by detecting the contradiction of its negation. Instead of transforming the original proposition into an equivalent tautology (when it is possible), the resolution method first puts the proposition into an equivalent, conjunctive normal form (or a set of clauses) and then repeatedly applies a inference rule. The major advantage is that only one inference rule called the resolution rule is needed. When applied to two resolvable clauses, one containing the literal A and the other containing $\neg A$, the negation of A, the resolution rule produces a resolvent, which is a clause consisting of the union of the two clauses with A and $\neg A$ removed.

Given a set of clauses, the original resolution method of Robinson goes as follows. When two clauses are resolvable, add their resolvent to the clause set. Repeat this until an empty clause is found or there is no further resovable clauses. Because there are only finitely many possible resolvents, this procedure eventually stops.

The Davis-Putnam procedure [3], on the other hand, deals with the same problem but it is less widely known. However, it has many virtues which we think make it easier to understand. This procedure can be explained better using the idea of a labeled transition system. Labeled transition systems are very general structures found use in many areas of computer science. A labeled transition system consists a set of states and a collection of transition relations between states. Each step of the Davis-Putnam procedure can be seen as a transition from a clause set S to a clause set S', labeled with a literal A. At an A-transition, written $S \xrightarrow{A} S'$, there is at least one resolution from S involving A and the old clause set S is replaced by a new clause set S' consisting of all the resolvents from S which involve A, together with those clauses of S which do not contain A or its negation. The Davis-Putnam procedure repeatedly formulates new clause sets in this way, resulting in a sequence of transitions like

$$S_0 \xrightarrow{A_0} S_1 \xrightarrow{A_1} S_2 \xrightarrow{A_2} \cdots S_k \xrightarrow{A_k} S_{k+1} \xrightarrow{A_{k+1}} \cdots .$$

Future resolution does not need any old clause. In other words, when $S \xrightarrow{A} S'$, the satisfiability of S is totally determined by that of S'. Note this is weaker than requiring S and S' to be logically equivalent. In fact from $S \xrightarrow{A} S'$ we can not deduce that S and S' are logically equivalent (an example is given later).

The Davis-Putnam procedure works: it is sound and complete. It is sound in the sense that whenever there is a clause set in the transition which contains an empty clause, the original clause set must be inconsistent; It is complete because if the original clause set is inconsistent, we will be able to detect it by deriving an empty clause from the transition. Since the Davis-Putnam procedure is sound and complete, it is as powerful as Robinson resolution. In addition, it enjoys the following advantages compared to the original Robinson resolution procedure.

In terms of the number of transitions, the Davis-Putnam procedure clearly terminates in m steps, where m is the number of propositional variables in the input clause set S (Note this does not imply that the Davis-Putnam procedure is polynomial as the size of the clause set may grow exponentially. In fact it is unlikely that there is a polynomial time algorithm for any resolution because the

3-satisfiability problem is well-known to be NP-complete).

The Davis-Putnam procedure abandons old clause set. After an A-transition like $S \xrightarrow{A} S'$, clauses in S which contain A or $\bar{A}$ do not appear in S', and S will never be needed in the future, compared with Robinson resolution where all old clause sets remain active (i.e. they may be needed in the future) and hence the number of active clauses never decrease.

More heuristic information is provided by the Davis-Putnam procedure. For the Davis-Putnam procedure, there are no two different transitions with the same label A. This means we have done, once and for all, all resolutions which involve A. For Robinson resolution, we cannot easily tell when a resolutions involving a particular literal is not needed any more.

The idea behind the Davis-Putnam procedure is the labeled transition system, used in many areas of computer science. It is very intuitive and easy to understand, resulting in a more intuitive and straightforward proof of its soundness and completeness by mathematical induction.

This chapter is based on a previous report by the author, 'Literal Resolution' [12]. Later when reading Galil ([3]) the author realized that this method was known to Davis and Putnam (although the method mentioned in [3] seems different from the one explained in Davis and Putnam ([2]) and Chang ([1])). However the author discovered it independently. Certainly this method deserves better recognition, and it is the purpose of this chapter to make it better known.

3.2 Conjunctive Normal Form

By the use of Boolean laws we can put every proposition into the form

$$\bigwedge_{j=1}^{n}(\bigvee_{i=1}^{n_j} A_{ij}).$$

For example, we have

$$A \vee B \vee (\neg C \wedge \neg D) = (\neg C \vee A) \wedge (\neg C \vee B) \wedge (\neg D \vee A) \wedge (\neg D \vee B).$$

We call

$$\bigwedge_{j=1}^{n} (\bigvee_{i=1}^{n_j} A_{ij})$$

a conjunctive normal form. Every proposition has a conjunctive normal form (this can be seen more clearly via truth tables), but it is not necessary unique. A conjunctive normal form $\bigwedge_{j=1}^{n}(\bigvee_{i=1}^{n_j} A_{ij})$ can be rewritten as a set of clauses:

$$\begin{aligned} &\{\,\{A_{11},\, A_{12},\, \cdots, A_{1n_1}\},\\ &\{A_{21},\, A_{22},\, \cdots, A_{2n_2}\}, \cdots,\\ &\{A_{n1},\, A_{n2},\, \cdots, A_{nn_n}\}\,\}. \end{aligned}$$

Corresponding to the normal form

$$(\neg C \vee A) \wedge (\neg C \vee B) \wedge (\neg D \vee A) \wedge (\neg D \vee B),$$

for example, we have the clause set

$$\{\, \{\neg C,\, A\},\, \{\neg C,\, B\},\, \{\neg D,\, A\},\, \{\neg D,\, B\}\, \}.$$

More formally, a *clause* is a finite set of literals. It represents the disjunction of its elements. A *clause set* is a set of clauses representing the conjunction of the formulae associated with the clauses. By convention, for a set L of literals $\bigwedge L$ is an abbreviation for the formula

$$\bigwedge_{s \in L} s$$

and similarly $\bigvee L$ is an abbreviation for the formula

$$\bigvee_{s \in L} s.$$

For non-empty L we clearly have

$$\bigwedge L = \bigwedge (L \cup \{\top\})$$

and

$$\bigvee L = \bigvee (L \cup \{\bot\}).$$

Generalizing this notation to the case when L is empty, it is reasonable to let

$$\bigwedge \emptyset = \top, \text{ and } \quad \bigvee \emptyset = \bot.$$

Hence the empty clause, written as $\Box$ to make a distinction from the empty set of clauses, represents $\bot$ and the empty clause set represents $\top$.

A clause set S is *valid* at Φ, written $\Phi \models S$, if for any clause C in S, $\top \in \{ \Phi(A) \mid A \in C \}$. Otherwise we say S is *invalid* at Φ. S is *satisfiable* if it has a valid truth assignment. Otherwise S is *unsatisfiable*. An unsatisfiable clause is also called *inconsistent*. S is *equivalent* to S' if they have the same set of valid truth assignments.

Let S be a clause set. It is easy to see that S is inconsistent if $\Box \in S$. In many applications one needs to check if a statement is always valid, i.e. a tautology. This can be reduced to the problem of deciding if a clause set S is inconsistent. Of course just because $\Box \notin S$ we cannot conclude that S is consistent. Fortunately there are methods that makes it possible to transform a clause set S into S' in such a way that S is inconsistent if $\Box \in S'$. One of the 'classical' method, which involves a single inference rule, is explained in the next section.

3.3 The Resolution Rule

When transforming a clause set S into S' to detect the inconsistency in S by finding a $\Box$ in S', one expects some logical relation to hold between S and S'. At least it has to be the case that the inconsistency of S' implies the inconsistency of S. A stronger requirement, that S and S' are equivalent, will suffice. This is what happens with one of the classical methods due to Robinson. The key is the *resolution rule*. When applied to two clauses, one containing A and the other containing $\neg A$, the negation of A, the resolution rule produces a new clause formed by the union of the two with A and $\neg A$ removed. Formally, the resolution rule is given as

$$\frac{C_0 \cup \{ A \} \qquad C_1 \cup \{ \bar{A} \}}{C_0 \cup C_1},$$

with C_0, C_1 clauses and A a literal. Note that $\bar{A}$ is the negation of A. This rule says that the clause $C_0 \cup C_1$, called the *resolvent* , is a logical consequence of $C_0 \cup \{ A \}$ and $C_1 \cup \{ \bar{A} \}$. In other words, if $C_0 \cup \{ A \}$ and $C_1 \cup \{ \bar{A} \}$ are both valid at a truth assignment Φ, then so does $C_0 \cup C_1$. This is because either $\Phi(A) = \bot$, or $\Phi(\neg A) = \bot$, which implies $\Phi \models C_0$, or $\Phi \models C_1$. Therefore, $\Phi \models C_0 \cup C_1$.

To be more precise we call an application of the resolution rule an *A-resolution* when the literals involved are A and $\bar{A}$. For example, $\{B\}$ is the resolvent of an A-resolution of the two clauses $\{A,\ B\}$, $\{\neg A\}$.

Robinson's method is described as follows. Let T be a finite clause set of atomic propositions (i.e. propositional variables). To detect the inconsistency of T define

$$\mathcal{R}(T) = T \cup \{ C \mid C \text{ is a resolvent of two clauses in } T \}$$

and let

$$\begin{aligned} \mathcal{R}^0(T) \ &= T \\ \mathcal{R}^{i+1}(T) \ &= \mathcal{R}(\mathcal{R}^i(T)) \text{ for each } i \geq 0. \end{aligned}$$

Consider the the clause set

$$\mathcal{R}^*(T) = \bigcup_{i \geq 0} \mathcal{R}^i(T).$$

Because there can only be finitely many resolvents, $\mathcal{R}^*(T) = \mathcal{R}^n(T)$ for some n. One expects to reveal the inconsistency in T by finding the empty clause in $\mathcal{R}^*(T)$ as illustrated by the following example.

Does this method always work? There is a theorem (due to Robinson) which states that T is inconsistent if and only if $\Box \in \mathcal{R}^*(T)$. Therefore the method always works. Another terminology often used for this kind of property is soundness and completeness. In general, soundness means that whenever the empty clause is found, the original clause set is inconsistent while completeness means if the original clause set is inconsistent, the empty clause will eventually be found.

It is easy to see the soundness of Robinson's method because T and $\mathcal{R}^*(T)$ are equivalent. The proof of the completeness, however, is not straightforward. Often it is better to consider the contrapositive of the completeness, which states that if the empty clause is not in $\mathcal{R}^*(T)$, then T is satisfiable. This amounts to demonstrating the existence of a valid truth assignment for T from the assumption that $\Box \notin \mathcal{R}^*(T)$. How can we find a valid truth assignment for T given that $\Box$ is not in $\mathcal{R}^*(T)$? This can be done by mathematical induction on the number of literals involved. However the prove is not intuitive and straightforward. By using a special resolution strategy proposed by Davis and Putnam, we can make the proof much easier to understand.

3.4 Labeled Transition Systems

Think about the modeling of the play of chess. There are two major components involved: one is the chess positions, and the other is the legal moves. The possible positions form a set which we call the states. The legal moves form a set of relations labeled by the description of the moves. For example, $\xrightarrow{\alpha}$ with $\alpha = (W, P, e2, e4)$ is a relation describing the move of *White*'s *Pawn* from position $e2$ to $e4$. Of course for the pair of positions $(P_1, \ P_2)$ to be in $\xrightarrow{\alpha}$, P_1 must have a white pawn at $e2$, and P_2 at $e4$.

A labeled transition system is a set with a collection of relations on it. The set is thought of as the state space, and the relations describe the changing of states via some actions. Because of this nature, labeled transition systems have found wide use in computer science. In particular, the idea can be used in the resolution method given in the previous section. In this section we give a short introduction to labeled transition systems.

Definition 3.4.1 *A labeled transition system is a pair*

$$(S, \{ \xrightarrow{\alpha} \mid \alpha \in L \})$$

where S is the set of states and $\xrightarrow{\alpha}$ is a relation on S for each α in L.

The behavior of a labeled transition system is specified by the sequences of transitions.

Definition 3.4.2 *Let*

$$(S, \{ \xrightarrow{\alpha} \mid \alpha \in L \})$$

be a labeled transition system. A sequence

$$s_0 \xrightarrow{\alpha_0} s_1 \xrightarrow{\alpha_1} s_2 \cdots \xrightarrow{\alpha_{k-1}} s_k \cdots$$

is called a transition sequence of the labeled transition system if (s_0, s_1) is a pair in the relation $\xrightarrow{\alpha_0}$, (s_1, s_2) a pair in $\xrightarrow{\alpha_1}$ $\cdots$, and (s_{k-1}, s_k) a pair in $\xrightarrow{\alpha_{k-1}}$, in general.

The transition sequences we are interested in are usually finite ones, though they may be very long. For example, a game of chess is just a transition sequence with black and white moving alternatively, leading to a final position where the game is over. In many cases the labeled transition system has a *initial state* which every transition sequence starts with. The opening position of chess is a initial state.

Also, sometimes in a transition system the order of the transition does not affect the state that can be reached. When that happens, the transition system is said to have the Church-Rosser property. More precisely, a transition system has the Church Rosser property if whenever there are two transitions from a state s such as

$$\begin{tikzcd} & s \arrow[dl, "\alpha"'] \arrow[dr, "\beta"] & \\ s_0 & & s_1 \end{tikzcd}$$

then there is a state s' such that

$$
\begin{tikzcd}
 & s \arrow[dl, "\alpha"'] \arrow[dr, "\beta"] & \\
s_0 \arrow[dr, "\beta"'] & & s_1 \arrow[dl, "\alpha"] \\
 & s' &
\end{tikzcd}
$$

This diamond shape tells us that the order of making a α or a β transition does not affect where we can go. If this is true for all labels α and β, it menas that we do not have to care about the sequence we make moves.

Probably the best known examples of labeled transition systems are finite automata. Of course labeled transition systems can be used to describe many different situations. They can be used in modeling game playing, in describing concurrent systems, and even in specifying the operational semantics of programming languages.

The operational semantics of programming languages provides a way to understand program behavior by simulate programs on some 'abstract' machines. The abstract machines used are labeled transition systems. Here, each state is a pair $\langle W, \sigma \rangle$, where W is the piece of program left to executed, σ is a function specifying the values stored at each location. The transition relation is given by a set of rules associated with program constructions. For further detail about this approach see [9].

The idea of labeled transition systems can also be applied to resolution. Define a labeled transition system by taking the set of the finite clause sets as the set of states. The transition relations are labeled by literals. To have a transition $S \xrightarrow{A} S'$, S must contain at least two clauses, one with A in it and the other with $\neg A$. And S' must be derived from S, where

$$S' = (S \setminus \{C \in S \ | \ A \in C \text{ or } \neg A \in C\})$$

$$\cup\{\, R \mid R \text{ is an } A-\text{resolvent of some clauses in } S\}.$$

For example, we have

$$\{\,\{\,A,\,B,\,\neg C\,\},\,\{\,A,\,B,\,C\,\},\,\{\,A,\,\neg B\,\},\,\{\,\neg A\,\}\,\}$$
$$\xrightarrow{C}\ \{\,\{\,A,\,B\,\},\,\{\,A,\,\neg B\,\},\,\{\,\neg A\,\}\,\},$$

with the transition labeled by C.

Starting from any finite clause set, we can make transitions step by step according to the specified labeled transition system, resulting in a state where no further transition is possible (this is because there are only finitely many literals in a finite clause set). We hope to be able to detect the inconsistency of the initial clause set by finding a $\Box$ in the final clause set. This method is introduced in the next section, and more detailed analysis of the method follows.

3.5 The Davis-Putnam Procedure

As stated in the previous section, our resolution method comes from the idea of labeled transition systems. The transition system related to resolution has finite clause sets as states. The transition relation is given by the following definition.

Definition 3.5.1 *We say that* $S \xrightarrow{A} S'$ *is a* literal transition *from a clause set* S *to a clause set* S' *labeled with literal* A *if*

1. there exist $C,\ C'$ in S such that $A \in C$, $\bar{A} \in C'$;
2. $S' = \{C \mid C \text{ is a resolvent of an } A\text{-resolution of two clauses in } S\}$
$\cup\{C' \mid C' \in S \text{ and } \{A,\ \bar{A}\} \cap C' = \emptyset\}$

To be definite, we always label a literal transition $S \xrightarrow{A} S'$ with $A \in \mathcal{A}$. Call $S \xrightarrow{A} S'$ an A-transition. Sometimes we omit the label and write $S \longrightarrow S'$, to mean that there is a literal transition from S to S'.

Example

$$\{\,\{\,A,\,B\,\},\{\,\neg B\,\}\,\} \xrightarrow{B} \{\,\{\,A\,\}\,\}$$

is a B-transition.

Naturally, an algorithm can be derived from the idea explained above. We call the derived algorithm the Davis-Putnam procedure, proposed first by Davis-Putnam [2], [3], and re-discovered by the author [12].

The Davis-Putnam Procedure

Step 1. Input a finite clause set S_0.

Step 2. For any literal A, any clause C of S_0, if A and $\bar{A}$ both appear in C, replace that clause by $\{\top\}$.

Step 3. If $\Box \in S_0$ or there is no resolvent from S_0, then stop.

Step 4. Choose a literal A such that for some C, $C' \in S_0$, $A \in C$, $\bar{A} \in C'$; Replace S_0 by S, where S is specified by the A-transition $S_0 \xrightarrow{A} S$; Go to step 2.

After the algorithm stops, if $\Box$ is in the current clause set then the input clause is unsatisfiable; otherwise it is satisfiable. This is exactly the soundness and completeness of the method, which will be proved in section 7.

Note that Robinson's resolution method can also be viewed as a labeled transition system. We have the same set of states, consisting of the finite clause sets. There is one labeled transition relation (and hence we omit the label), given by

$$S \longrightarrow S'$$

if and only if $S' = S \cup \{C\}$, where C is the resolvent of two clauses from S.

3.6 Examples

In this section we give examples for the Davis-Putnam procedure. Comparison is also made with the classical resolution method.

Example For clause set

$$\{\{A, B, \neg C\}, \{A, B, C\}, \{A, \neg B\}, \{\neg A\}\},$$

we have the following sequence of literal transitions:

$$\begin{aligned}
&\{\,\{\,A,\,B,\,\neg C\,\},\,\{\,A,\,B,\,C\,\},\,\{\,A,\,\neg B\,\},\,\{\,\neg A\,\}\,\}\\
&\xrightarrow{C}\{\,\{\,A,\,B\,\},\,\{\,A,\,\neg B\,\},\,\{\,\neg A\,\}\,\}\\
&\xrightarrow{B}\{\,\{\,A\,\},\,\{\,\neg A\,\}\,\}\\
&\xrightarrow{A}\{\,\Box\,\}.
\end{aligned}$$

So

$$\{\,\{\,A,\,B,\,\neg C\,\},\,\{\,A,\,B,\,C\,\},\,\{\,A,\,\neg B\,\},\,\{\,\neg A\,\}\,\}$$

is unsatisfiable.

The Davis-Putnam procedure provides a resolution strategy which *replaces the old clause set by a new clause set.* The key idea is that each step performs resolutions related to a *single literal.* Note that if at each step we only perform *one* resolution, or *all* possible resolution, the corresponding labeled transition systems do not give rise to correct resolution algorithms, as shown by the following two examples.

Example Performing an A-resolution on

$$\{\,\{\,A,\,B\,\},\,\{\,A,\,\neg B\,\},\,\{\,\neg A\,\}\,\}$$

to the clause $\{\,A,\,B\,\}$ and $\{\,\neg A\,\}$ we can get

$$\{\,\{\,B\,\},\,\{\,A,\,\neg B\,\}\,\}.$$

This clause set is satisfiable while the original one is not. The reason is that we did not do all the possible resolutions involving literal A.

Example Performing *all* possible resolutions on

$$\{\,\{\,A,\,B\,\},\,\{\,\neg B\,\},\,\{\,\neg A\,\}\,\}$$

we get

$$\{\,\{\,B\,\},\,\{\,A\,\}\,\}.$$

We also failed in detecting the inconsistency of the original clause set. This is because in one step, we did resolutions involving both A and B.

The Davis-Putnam resolution works for the two examples as demonstrated below. For the first example we have

$$\begin{aligned}\{\,\{\,A,\,B\,\},\,\{\,A,\,\neg B\,\},\,\{\,\neg A\,\}\,\} &\xrightarrow{A} \{\,\{\,B\,\},\,\{\,\neg B\,\}\,\}\\ &\xrightarrow{B} \{\,\Box\,\},\end{aligned}$$

while for the second example,

$$\begin{aligned}\{\,\{\,A,\,B\,\},\,\{\,\neg B\,\},\,\{\,\neg A\,\}\,\} &\xrightarrow{A} \{\,\{\,B\,\},\,\{\,\neg B\,\}\,\}\\ &\xrightarrow{B} \{\,\Box\,\}.\end{aligned}$$

As one more example, let's consider the clause set

$$\{\,\{\,\neg A,\,B\,\},\,\{\,\neg A,\,\neg B,\,C\,\},\,\{\,A\,\},\,\{\,\neg C\,\},\,\{\,\neg A\,\}\,\}.$$

The following sequence of transitions produces an empty clause:

$$\begin{aligned}&\{\,\{\,\neg A,\,B\,\},\,\{\,\neg A,\,\neg B,\,C\,\},\,\{\,A\,\},\,\{\,\neg C\,\},\,\{\,\neg A\,\}\,\}\\ &\xrightarrow{A} \{\,\{\,B\,\},\,\{\,\neg B, C\,\},\,\{\,\neg C\,\}\,\}\\ &\xrightarrow{C} \{\,\{\,B\,\},\,\{\,\neg B\,\}\,\}\\ &\xrightarrow{A} \{\,\Box\,\}.\end{aligned}$$

Therefore the original clause set is unsatisfiable.

3.7 Soundness and Completeness

In this section we prove that the Davis-Putnam procedure is sound and complete. The Davis-Putnam procedure is *sound* in the sense that for any input clause set S_0, whenever $\Box$ is in the current clause set after the Davis-Putnam resolution terminates, S_0 is unsatisfiable. It is *complete* in the sense that if the input clause S_0 is unsatisfiable, then $\Box$ must be in the current clause set when the Davis-Putnam resolution stops.

Note that for Robinson resolution, $\mathcal{R}^*(S)$ is equivalent to S. This is not a necessity for resolution to work. In fact the Davis-Putnam resolution does not have this property, i.e., when $S \longrightarrow S'$, S and S' need not be equivalent as the first example in the previous section shows.

The following lemma is the key for completeness.

Lemma 3.7.1 *Let S be a finite clause set and*

$$S \xrightarrow{A} S'.$$

Then the satisfiability of S' implies the satisfiability of S.

Proof This short proof is provided by Bill Rounds and Phil Haar for the EECS 574 class at the University of Michigan (Thanks to Bill for allowing me to use it here). It is in turn based on the idea of a proof in the book of Davis and Weyuker.

Let S be a set of clauses and let A be a literal. We define

$$\begin{aligned} S_A^0 &= \{C \in S \mid \text{ neither } A \text{ nor } \neg A \text{ occurs in } C\}, \\ S_A^+ &= \{C \in S \mid A \text{ occurs in } C\}, \\ S_A^- &= \{C \in S \mid \neg A \text{ occurs in } C\}. \end{aligned}$$

We also have the notion $res_A(C_1, C_2)$ as the result of resolving out A given that $C_1 \in S_A^+$ and $C_2 \in S_A^-$ (or the other way round). We redefine the Davis-Weyuker notion of RES:

$$RES_A(S) = S_A^0 \cup \{res(C_1, C_2) \mid C_1 \in S_A^+ - S_A^-, C_2 \in S_A^- - S_A^+\}.$$

This new RES does not allow resolution with clauses containing both A and $\neg A$. But because these clauses are always true, they do not contribute anything towards finding a contradiction. Moreover, the new RES always eliminates A and $\neg A$ from a set of clauses.

Now let Φ be a truth assignment and A be the propositional variable mentioned in the lemma. We claim that if $\Phi(RES_A(S)) = \top$ then either $\Phi[\bot/A](S) = \top$ or $\Phi[\top/A](S) = \top$. The lemma clearly follows from the claim.

Assume then that $\Phi[\bot/A](S) = \bot$. We will show that $\Phi[\top/A](S) = \top$. Now clearly
$\Phi[\top/A](S_A^0) = \top$, because A does not occur here, and because $\Phi(RES_A(S)) = \top$. We will now show that $\Phi[\top/A](S_A^+) = \top$ and $\Phi[\top/A](S_A^-) = \top$.

To see the first of these assertions, notice that there can be no clause $C \in S_A^+$ with $\Phi[\top/A](C) = \bot$, by definition of S_A^+. This proves the first assertion, so now we try to show that $\Phi[\top/A](S_A^-) = \top$. Suppose there is a clause C_2 in $S_A^- - S_A^+$ such that $\Phi[\top/A](C_2) = \perp$. Hold on to this clause for the moment.

We next use the hypothesis that $\Phi[\perp/A](S) = \perp$. This means that one of the clauses C_1 of S evaluates to $\perp$ when you set $A = \perp$ in Φ and then evaluate with this new assignment. C_1 cannot be in S_A^-, again by definition of S_A^-. C_1 cannot be in S_A^0 either, because $\Phi[\perp/A](S_A^0) = \top$. So $C_1 \in S_A^+ - S_A^-$.

We now have a clause $C_1 \in S_A^+ - S_A^-$ which evaluates to $\perp$ under Φ when A is reassigned to $\perp$, and a clause $C_2 \in S_A^- - S_A^+$ which evaluates to $\perp$ under Φ when A is reassigned to $\top$. This implies that under Φ, the literals in both clauses, other than A or $\neg A$, must all have value $\perp$. Thus the resolvent $res(C_1, C_2)$ must be defined, and have the value $\perp$ under Φ. But this contradicts our hypothesis $\Phi(RES_A(S)) = \top$. This contradiction proves the claim, and thus the lemma.

$\square$

Now we can prove the soundness and completeness of the Davis-Putnam procedure.

Theorem 3.7.1 *A finite clause set S_0 is unsatisfiable if and only if there are literals A_0, A_1, $\cdots$ A_n in S_0 such that*

$$S_0 \xrightarrow{A_0} S_1 \xrightarrow{A_1} S_2 \xrightarrow{A_2} \cdots S_{n-1} \xrightarrow{A_{n-1}} S_n$$

and $\square \in S_n$, where whenever a literal and its negation appear in some clause at the same time, we always replace that clause by $\{\top\}$.

Proof $(\Leftarrow)$: Suppose S is satisfiable and $S \xrightarrow{A} S'$. Then clearly S' is satisfiable by the soundness of the resolution rule. Thus if $\square \in S_n$, S_0 must be unsatisfiable.

$(\Rightarrow)$: Suppose S_0 is unsatisfiable. Suppose

$$S_0 \xrightarrow{A_0} S_1 \xrightarrow{A_1} S_2 \xrightarrow{A_2} \cdots S_{n-1} \xrightarrow{A_{n-1}} S_n,$$

$\square \notin S_n$, but S_n has no further transition. Then S_n is satisfiable because it has the valid truth assignment Φ_0 given below. For each

$C \in S_n$, $C \neq \emptyset$. Let Φ_0 be an assignment which assigns $\top$ to A if $A \in \bigcup S_n$; assigns $\perp$ to A if $\neg A \in \bigcup S_n$. Clearly $\Phi_0 \models S_n$ because $A \in \bigcup S_n$ implies $\neg A \notin \bigcup S_n$ and $\neg A \in \bigcup S_n$ implies $A \notin \bigcup S_n$.

By the lemma given above, we deduce that S_0 is satisfiable, which is a contradiction. This means we must have $\Box \in S_n$.

$\Box$

It is clear that for a clause set S, the satisfiability of S can be decided in no more than m steps of transition, where m is the number of propositional variables in S.

Using the idea of the Davis-Putnam procedure we propose the following restricted form of Robinson resolution: for any (finite) clause set S of atomic propositions, define

$$\mathcal{L}_{A_0 A_1 A_2 \cdots A_k \cdots}(S) = \bigcup_{i \geq 0} S_i,$$

where $S_0 = S$ and S_i's form a literal transition sequence:

$$S_0 \xrightarrow{A_0} S_1 \xrightarrow{A_1} S_2 \xrightarrow{A_2} \cdots S_k \xrightarrow{A_k} S_{k+1} \xrightarrow{A_{k+1}} \cdots.$$

When no confusion arises, we simply write $\mathcal{L}(S)$, omitting the subscripts.

What is the relation between $\mathcal{L}(S)$ and $\mathcal{R}^*(S)$, the clause set used for Robinson's resolution given in section 3?

It is not difficult to see that we have

Theorem 3.7.2 *For every sequence* $A_0 A_1 A_2 \cdots A_k \cdots$,

$$\mathcal{L}_{A_0 A_1 A_2 \cdots A_k \cdots}(S) \subseteq \mathcal{R}^*(S).$$

In fact, we have $S_0 = \mathcal{R}^0(S) = S$ and $S_i \subseteq \mathcal{R}^i(S)$ for each $i \geq 1$. Therefore

$$\bigcup_{i \geq 0} S_i \subseteq \bigcup_{i \geq 0} \mathcal{R}^i(S),$$

which is exactly the conclusion of the theorem.

However, the number of clauses generated by the Davis-Putnam procedure is sometimes strickly less than that generated by Robinson's method. For example, take S to be the clause set

$$\{ \{ A, \neg B \}, \{ \neg A, B \}, \{ B \} \}.$$

We have

$$\{ \{ A, \neg B \}, \{ \neg A, B \}, \{ B \} \} \xrightarrow{A} \{ \{ \top \}, \{ B \} \}.$$

Therefore

$$\mathcal{L}_A(S) = \{ \{ A, \neg B \}, \{ \neg A, B \}, \{ B \}, \{ \top \} \}.$$

But

$$\mathcal{R}^*(S) = \{ \{ A, \neg B \}, \{ \neg A, B \}, \{ A \}, \{ B \}, \{ \top \} \}.$$

From the completeness of the Davis-Putnam procedure we immediately get

Corollary 3.7.1 *The propositional version of Robinson resolution is sound and complete.*

This means we have a simpler way to prove that Robinson's method works.

In most cases the size of $\mathcal{L}(S)$ can be much smaller than that of $\mathcal{R}^*(S)$. Take second example, for example. The Davis-Putnam procedure produced 6 new clauses while brute force Robinson resolution produces the following 11 new clauses.

$$\{ A, B \}, \{ B, \neg C \}, \{ B, C \}, \{ \neg B \}, \{ A, \neg C \}, \{ A, C \},$$

$$\{ A \}, \{ B \}, \{ C \}, \{ \neg C \}, \Box.$$

3.7.1 Discussion

We have presented a resolution strategy called the Davis-Putnam resolution and we have shown that the Davis-Putnam resolution is sound and complete. The basic idea of the Davis-Putnam resolution is to perform all resolutions related to a particular literal in each step, and to replace the old clause set by a new clause set consisting of the literal-related resolvents, and clauses which can not involve in any resolution related to the literal. From the proof of the soundness and completeness theorem we can see that the order of literals chosen at each step does not affect whether we can find a $\Box$ eventually. This is not to say, however, that if there are two transitions from S such as

$$\begin{tikzcd} & S \arrow[dl, "A"'] \arrow[dr, "B"] & \\ S_0 & & S_1 \end{tikzcd}$$

then there exists a clause set S' such that

$$\begin{tikzcd} & S \arrow[dl, "A"'] \arrow[dr, "B"] & \\ S_0 \arrow[dr, "B"'] & & S_1 \arrow[dl, "A"] \\ & S' & \end{tikzcd}$$

(this is the so called Church-Rosser property) as the following example shows.

Example Consider the clause set

$$\{\, \{\, A, \neg B \,\}, \, \{\, \neg A, \, B \,\}, \, \{\, B \,\} \,\}.$$

We have

$$\{\,\{\,A, \neg B\,\}, \{\,\neg A, B\,\}, \{\,B\,\}\,\} \;\xrightarrow{A}\; \{\,\{\,\top\,\}, \{\,B\,\}\,\}$$

and

$$\{\,\{\,A, \neg B\,\}, \{\,\neg A, B\,\}, \{\,B\,\}\,\} \;\xrightarrow{B}\; \{\,\{\,\top\,\}, \{\,A\,\}\,\}.$$

But there are no transitions for $\{\,\{\,\top\,\}, \{\,B\,\}\,\}$ and $\{\,\{\,\top\,\}, \{\,A\,\}\,\}$ to get to the same clause set.

This leaves us some room to choose a better transition sequence to reduce the time complexity. In general it is suggested that we choose a literal which results in as small the number of resolvents as possible. In particular, if $C \cup \{\,A\,\}$, $C' \cup \{\,\neg A\,\}$ are the only two clauses in S which have A or $\neg A$ present, then a transition $S \xrightarrow{A} S'$ will make the size of S' less than that of S.

This chapter is concerned with simpler ways to explain the principle of the resolution method. It is not about complexity of the resolution. That topic has been addressed by other people (see [3] and [5], etc). However, the following remarks about the complexity issue of the resolution procedures are straightforward.

It is well-known that deciding whether a set of clause is satisfiable or not is an expensive question. In fact, SAT, the collection of satisfiable finite clause sets, is a language in the NP-complete class. Since the resolution method can detect inconsistency (as well as satisfiability), it is unlikely that there is any efficient (i.e. polynomial time) algorithm for resolution. The Davis-Putnam procedure, not surprisingly, does not seem to run in polynomial time in terms of the number of clauses generated.

However, it is clear that if there are no more than k literals in S then the Davis-Putnam procedure will terminate with a time complexity of $O(n^{2^k})$, where n is the size of S. This is because there will be at most k literal transitions. The first transition generates at most n^2 clauses, and the second transition n^4 clauses, and so on. The k-th transition, the final step, generates at most n^{2^k} clauses.

Therefore the total number of clauses generated is at most

$$n^2 + n^4 + \cdots + n^{2^k} = O(n^{2^k}),$$

which is a polynomial in n.

It should be interesting to give an analysis of the average time complexity of the Davis-Putnam procedure and to compare it with other resolution strategies.

Partial List of References

[1] Chang, C.L. and Lee, R.C.T., *Symbolic Logic and Mechanical Theorem Proving*, Academic Press, New York, 1973.

[2] Davis, M., Putnam, H., A Computing Procedure for Quantification Theory, JACM 7, 1960.

[3] Galil, Z. On the complexity of regular resolution and the Davis-Putnam procedure, Theoretical Computer Science 4, 1977.

[4] Gallier, J.H. *Logic for Computer Science-foundations of automatic theorem proving*, Harper and Row, New York, 1986.

[5] Haken, A. The intractability of resolution, Theoretical Computer Science 39, 1985.

[6] Kowalski, R.A., Hayes, P. Semantic Trees in Automatic Theorem Proving, *Machine Intelligence* Vol 4, (B. Meltzer and D. Michie, eds), American Elsevier, New York, 1969.

[7] Kowalski, R.A., *Logic for Problem Solving*, Elsevier North Holland, New York, 1979.

[8] Lewis, H.R. and Papadimitriou, C.H, *Elements of the Theory of Computation,* Prentice-Hall, Englewood Cliffs NJ, 1981.

[9] Plotkin, G.D. *Structural Operational Semantics*, Technical Report DAIMI FN-19, Department of Computer Science, Aarhus University, 1981.

[10] Robinson, J.A., A Machine-oriented Logic Based on the Resolution Principle, JACM, 12, (1) 1965.

[11] Wos, L., Carson D.F. and Robinson, J.A. The Unit Preference Strategy in Theorem Proving, AFIPS Conference Proceedings, 1964.

[12] Zhang, G.Q., Literal Resolution, Technical Report DAIMI PB-277, Department of Computer Science, Aarhus University, 1989