

UNSEARCHMO: Eliminating Redundant Search Space on Backtracking for Forward Chaining Theorem Proving*

Lifeng He

Faculty of Information Science and Technology
Aichi Prefectural University, Aichi, 480-1198 Japan

Abstract

This paper introduces how to eliminate redundant search space for forward chaining theorem proving as much as possible. We consider how to keep on minimal useful consequent atom sets for necessary branches in a proof tree. In the most cases, an unnecessary non-Horn clause used for forward chaining will be split only once. The increase of the search space by invoking unnecessary forward chaining clauses will be nearly linear, not exponential anymore. In a certain sense, we “unsearch” more than necessary. We explain the principle of our method, and provide an example to show that our approach is powerful for forward chaining theorem proving.

1 Introduction

Automated reasoning is one of the most important topics for artificial intelligence and computer science. Among others, theorem proving technologies for first-order predicate calculus have been attracting much interesting [Robinson, 1965; Loveland, 1968; Manthey and Bry, 1988].

It is well-known that if S' is an unsatisfiable subset of a clause set S , it is generally easier to show S' unsatisfiable than to show S unsatisfiable. Obviously, the smaller such a S' is, the shorter a proof is. If forward chaining is used for reasoning about non-Horn clauses, it means that we need not consider every violated clause, but only those that can help us to find a refutation. It is obvious that invoking unnecessary non-Horn clauses will explode the search space exponentially.

SATCHMO (SATisfiability CHEcking by MOdel generation) [Manthey and Bry, 1988] is potentially inefficient, since it uses all violated clauses to forward chaining. Addressing to this problem, two strategies have been developed. One is utilizing an intelligent strategy to only select those relevant-like non-Horn clauses for forward chaining, which is proposed in [Ramsay, 1991], refined in [Loveland *et al.*, 1995] (SATCHMORE) and further improved in [He *et al.*, 1998]

(A-SATCHMORE). The other method, proposed in [He, 2001] (called I-SATCHMO) is eliminating redundant searching space after unnecessary non-Horn clauses have been used for forward chaining by intelligent backtracking.

As indicated in [He, 2001], there are mainly two problems about the approach to select a suitable non-Horn clause for forward chaining. One is that it takes much overhead cost to decide whether a non-Horn clause is suitable or not. In general, the stricter the checking is, the more expensive the overhead cost is. The other is that any such strategy is only effective in a limited range and certainly fails in some more complicated cases. In fact, a non-Horn clause recognized as “relevant” by some advanced selecting strategy might be practically unnecessary to the refutation being made. In other words, there are always such irrelevant non-Horn clauses that can pass any relevancy test. These problems have been illustrated on some examples in [He, 2001].

The principle of the method of eliminating redundant searching space by intelligent backtracking is very simple: If one of the consequent atoms of a forward chaining clause used for forward chaining is found to be useless to the reasoning on backtracking, the use of this clause is known as unnecessary, and the remaining processing over the clause's consequence is immediately abandoned. It only takes a little overhead cost to find irrelevant non-Horn clauses and is compatible with the former strategy introduced above.

However, marking those consequent atoms contributed to derive antecedents of irrelevant forward chaining clauses as useful and mixing up the useful consequent atoms for different branches in a proof tree, I-SATCHMO might make redundant search.

This paper introduces a solution for these problems. As we will see, in our improved method, for each node in the proof tree, only those indispensable consequent atoms to derive the refutation at the node are marked as useful. Thus, our method can show a refutation for a given unsatisfiable clause set without UNSEARCHing MOre than necessary (hence the acronym UNSEARCHMO). In other words, our method is one of the most efficient strategies for forward chaining theorem proving.

2 SATCHMO and I-SATCHMO

We assume familiarity with the details relating to SATCHMO [Manthey and Bry, 1988], SATCHMORE [Loveland *et al.*,

*This work is partially supported by the Japanese Ministry of Education, Science, Sports and Culture and the Artificial Intelligence Research Promotion Foundation of Japan.

1995] as well as $\mathcal{A}$ -SATCHMORE and limit ourselves to briefly reviewing the basic material needed for our presentation.

In this paper, a problem for checking unsatisfiability is represented by means of positive implicational clauses, each of which has of the form $A_1, \dots, A_n \rightarrow C_1; \dots; C_m$ ($n, m \geq 0$). We refer to the implicit conjunction $A_1, \dots, A_n$ as the antecedent of the clause, with each A_i being an antecedent atom. The implicit disjunction $C_1; \dots; C_m$ is referred to as the consequent of the clause, and each C_j is a consequent atom. A clause with an empty consequent, i.e., $m = 0$, is called a *negative clause* and is written as $A_1, \dots, A_n \rightarrow \perp$, where $\perp$ means *false*. A clause with an empty antecedent, i.e., $n = 0$, is written with consequent part only if it is a fact (Horn clause), and otherwise the antecedent atom *true* is added. Moreover, a given clause set is divided into two subsets. One is $\mathcal{BC}$, the Backward Chaining component that is a decidable Horn clause set consisting of all negative clauses, facts and any Horn clauses selected by user. The other is $\mathcal{FC}$, the Forward Chaining component that contains all the remaining clauses.

If goal $\mathcal{G}$ (a conjunction of atoms) can be proven to logically follow the clause set $\mathcal{S}$, we denote that with $\mathcal{S} \vdash \mathcal{G}$. On the other hand, if $\mathcal{I}$ is a ground atom set and A a ground atom, $A \in \mathcal{I}$ indicates that atom A is a member of $\mathcal{I}$. Because we utilize forward chaining on non-Horn clauses same as SATCHMO, all of clauses must hold the so-called *range-restricted property*¹ to guarantee the soundness for refutation.

Suppose that $\mathcal{BC}$ be a decidable Horn clause set and $\mathcal{I}$ a ground atom set, then a conjunction (disjunction) of ground atoms is *satisfiable* in $\mathcal{BC} \cup \mathcal{I}$ if all (some) of its members can be derived from $\mathcal{BC} \cup \mathcal{I}$, and else *unsatisfiable*. A ground clause $A \rightarrow C$ is said *satisfiable* in $\mathcal{BC} \cup \mathcal{I}$ if C is satisfiable or A is unsatisfiable in $\mathcal{BC} \cup \mathcal{I}$, and else *violated*.

It is well-known that a clause set $\mathcal{S}$ is satisfiable if and only if we can find a model $\mathcal{M}$ for $\mathcal{S}$. If $\mathcal{M}$ is a model of $\mathcal{S}$, then each clause of $\mathcal{S}$ is satisfiable in $\mathcal{M}$. Based on such consideration, to check whether a given clause set $\mathcal{S} = \mathcal{BC} \cup \mathcal{FC}$ is satisfiable, SATCHMO goes to construct a model for this clause set by trying to satisfy all clauses in $\mathcal{S}$.

The reasoning made by SATCHMO to search a model for a given clause set $\mathcal{BC} \cup \mathcal{FC}$ can be graphically illustrated in a *Proof Tree* (abbreviated to *PT* hereafter) described as follows.

For the current node D (initially the root node $\{\mathcal{BC}\}$), and the ground consequent atom set $\mathcal{I}_D$ (initially empty):

1. If $\mathcal{BC} \cup \mathcal{I}_D \vdash \perp$, create a leaf node $\perp$ below node D . The process for the branch terminates.
2. If $\mathcal{BC} \cup \mathcal{I}_D \not\vdash \perp$, select an instantiated ground violated clause $A_1, \dots, A_n \rightarrow C_1; \dots; C_m$ from $\mathcal{FC}$. If no such clause exists, the process terminates to report that $\mathcal{BC} \cup \mathcal{FC}$ is satisfiable.

¹A clause is range-restricted iff every variable in its consequent also occurs in its antecedent. An important property of such clauses is that if its antecedent is satisfied by a set of ground atoms, then every consequent atom is certainly ground. As indicated by Manthey and Bry (1988) and Bry and Yahya (2000), any first-order clause set can be transformed into the same “meaning” range-restricted one.

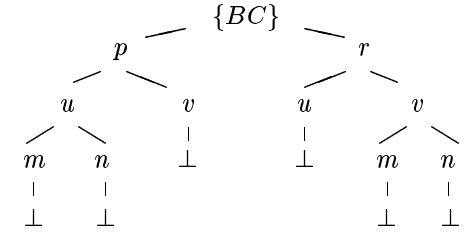


Figure 1: The proof tree *PT* constructed by SATCHMO

3. For each a consequent atom C_i of the selected violated $\mathcal{FC}$ clause, create a child node C_i below the current node. Taking node C_i as new current node, call this procedure recursively in depth-first strategy with the augmented ground consequent atom set $\mathcal{I}_{C_i}$, where $\mathcal{I}_{C_i} = \mathcal{I}_D \cup \{C_i\}$.

For node D , where the ground consequent atom set is $\mathcal{I}_D$, if all branches terminate in a leaf node $\perp$, it means that satisfying atom D there will lead to a refutation anyway. In other words, $\mathcal{BC} \cup \mathcal{I}_D$ cannot be extended as a model of $\mathcal{BC} \cup \mathcal{FC}$. Such node is said to be *unsatisfiable*. When the node is the root node $\{\mathcal{BC}\}$, then $\mathcal{BC} \cup \mathcal{FC}$ is unsatisfiable. Corresponding to creating node D in *PV*, an atom D is asserted into the reasoning database. On the other hand, when node D is proven to be unsatisfiable, the atom D will be retracted from the database. Moreover, whenever a node is proven unsatisfiable, the reasoning process will backtrack to its parent node if any.

Example 2.1 Consider the following propositional clause set $\mathcal{S} = \mathcal{BC} \cup \mathcal{FC}$.

$$\begin{aligned}
 \mathcal{BC} : \quad & r, u \rightarrow \perp. & p, v \rightarrow \perp. & m \rightarrow \perp. \\
 & n \rightarrow \perp. & v, s \rightarrow \perp. & u, t \rightarrow \perp. \\
 \mathcal{FC} : \quad & \text{true} \rightarrow p; r. & \text{true} \rightarrow u; v. & w \rightarrow s; t. \\
 & \text{true} \rightarrow m; n.
 \end{aligned}$$

The proof tree generated by SATCHMO is shown in Figure 1. It is obvious that SATCHMO made some redundant search. In fact, a refutation can be derived from the subset consisting of $\mathcal{BC}$ and the only $\mathcal{FC}$ clause $\text{true} \rightarrow m; n$.

By the way, for the given clause set in this example, every violated $\mathcal{FC}$ clause used by SATCHMO is recognized as “relevant” by either SATCHMORE or $\mathcal{A}$ -SATCHMORE, the proof tree generated by SATCHMORE or $\mathcal{A}$ -SATCHMORE is the same as that shown in Figure 1.

The redundant search space can be cut down by intelligent backtracking presented in I-SATCHMO. As we have seen, a consequent atom D is retracted from the reasoning database only if the node D in the proof tree has been proven as unsatisfiable. Suppose that $A \rightarrow C_1; \dots; C_m$ be the violated $\mathcal{FC}$ clause used for forward chaining at node D in the proof tree, then, node D holds m child nodes $C_1, \dots, C_m$. If some consequent atom C_i is found to be useless to the reasoning at the time being retracted from the database (hence node C_i has been proven as unsatisfiable), i.e., it is useless to construct the partial proof tree below node C_i , the same partial proof tree can be directly constructed below node D , and therefore,

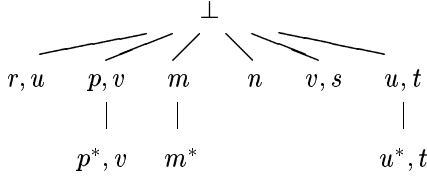


Figure 2: The deriving tree for goal $\perp$ at node m

at this point, we know that node D can be proven as unsatisfiable just in the same way made at node C_i . Since D has been known as *unsatisfiable*, the remaining process for showing node D unsatisfiable, i.e., the process on the consequent atoms $C_{i+1}, \dots, C_m$ (if any), can be removed at once. On the other hand, the use of an $\mathcal{FC}$ clause $A \rightarrow C_1; \dots; C_m$ is *necessary* only if each C_i ($1 \leq C_i \leq m$) is useful to constructing the partial tree below node C_i .

Now we introduce I-SATCHMO how to decide what consequent atom is *useful* for the reasoning. Obviously, a consequent atom asserted can only be used to derive goal $\perp$ or antecedents of violated $\mathcal{FC}$ clauses. Therefore, I-SATCHMO marks an asserted consequent atom as useful whenever it contributed to derive goal $\perp$ or an antecedent of a violated $\mathcal{FC}$ clause.

The useful consequent atoms to derive goal $\mathcal{G}$ can be found from the deriving tree of goal $\mathcal{G}$ [He, 2001].

Definition 2.1 Suppose that a give clause set be $\mathcal{S} = \mathcal{BC} \cup \mathcal{FC}$, $\mathcal{I}$ the current consequent atom set established by forward chaining, $\mathcal{G}$ a goal such that $\mathcal{BC} \cup \mathcal{I} \vdash \mathcal{G}$. The deriving tree of goal $\mathcal{G}$ with respect to $\mathcal{BC} \cup \mathcal{I}$, denoted by $DT_{\mathcal{G}}^{\mathcal{BC} \cup \mathcal{I}}$, is the result tree constructed as follows.

1. The root node is $\mathcal{G}$.
2. Suppose that $A_1, A_2, \dots, A_u, C_1^*, \dots, C_j^*$ is a node in the tree being constructed.
 - (a) For each $\mathcal{BC}$ clause $D_1, \dots, D_t \rightarrow H$ such that $H\sigma = A_1\sigma$, where σ is a most general unifier of H and A_1 , create a child node $D_1\sigma, \dots, D_t\sigma, A_2\sigma, \dots, A_u\sigma, C_1^*, \dots, C_j^*$.
 - (b) For each atom $C \in \mathcal{I}$ such that $C = A_1\theta$, where θ is a ground substitution (since C is a ground atom), create a child node $A_2\theta, \dots, A_u\theta, C_1^*, \dots, C_j^*, C_{j+1}^*$, where $C_{j+1} = C$.
3. Repeat this procedure recursively until an empty leaf or a leaf such that all of its atoms hold “*” is derived. Such a leaf is called *successful deriving leaf*.

Since $\mathcal{BC}$ is decidable and $\mathcal{BC} \cup \mathcal{I} \vdash \mathcal{G}$, $DT_{\mathcal{G}}^{\mathcal{BC} \cup \mathcal{I}}$ can be finally constructed with finding a successful deriving leaf. The consequent atoms that contributed to derive $\mathcal{G}$ from $\mathcal{BC} \cup \mathcal{I}$ are those atoms listed in the successful deriving leaf if any.

Example 2.2 Suppose that $\mathcal{S} = \mathcal{BC} \cup \mathcal{FC}$ be the clause set given in Example 2.1. Consider the leftmost node m in Figure 1, where $\mathcal{I}_m = \{p, u, m\}$. The deriving tree for goal $\perp$, $DT_{\perp}^{\mathcal{BC} \cup \mathcal{I}_m}$, is shown in Figure 2. To derive goal $\perp$, only m is useful.

The algorithm of I-SATCHMO to construct PT for checking whether a given clause set $\mathcal{S} = \mathcal{BC} \cup \mathcal{FC}$ is unsatisfiable can be described as follows.

For the current node D (initially the root node $\{\mathcal{BC}\}$) and the corresponding consequent atom set $\mathcal{I}_D$ (initially empty):

1. If $\mathcal{BC} \cup \mathcal{I}_D \vdash \perp$, create a leaf node $\perp$ under the current node, which means that node D is proven unsatisfiable. Mark all consequent atoms listed in the successful deriving node in $DT_{\perp}^{\mathcal{BC} \cup \mathcal{I}_D}$ as useful.
2. If $\mathcal{BC} \cup \mathcal{I}_D \not\vdash \perp$, select an instantiated ground violated clause from $\mathcal{FC}$. If no such clause exists, the process terminates to report that $\mathcal{BC} \cup \mathcal{FC}$ is not unsatisfiable.
3. For the selected violated non-Horn clause $A \rightarrow C_1; \dots; C_m$, mark all consequent atoms listed in the successful deriving node in $DT_{\perp}^{\mathcal{BC} \cup \mathcal{I}_D}$ as useful. Then, create a node C_i below the current node for each i such that $1 \leq i \leq m$. Taking C_i as the new current node, call this procedure recursively in depth-first strategy with the augmented consequent atom set $\mathcal{I}_{C_i} = \mathcal{I}_D \cup \{C_i\}$. However, if C_i has been found to be useless on backtracking, instead of each node C_j such that $i < j \leq m$, a special leaf node $\times$ is created, where $\times$ means that the process from there is pruned away. On the other hand, if all C_i are marked as useful, the use of $A \rightarrow C_1; \dots; C_m$ is known as necessary. In either case, node D is proven as unsatisfiable.

Example 2.3 Suppose that $\mathcal{S} = \mathcal{BC} \cup \mathcal{FC}$ be the clause set given in Example 2.1.

In the branch $\{\mathcal{BC}\} \rightarrow p \rightarrow u \rightarrow m \rightarrow \perp$ in Figure 1, only atom m is used in forward chaining and marked as useful. In the same way, in the branch $\{\mathcal{BC}\} \rightarrow p \rightarrow u \rightarrow n \rightarrow \perp$, only atom n is used in forward chaining and marked as useful.

Since both m and n are proven to be unsatisfiable, its parent node, i.e., u is proven to be unsatisfiable. We backtrack to node u to retract u . At that time, we know u is not marked as useful in the reasoning. Therefore, it is useless for the reasoning and the use of the violated $\mathcal{FC}$ clause $true \rightarrow u; v$ selected at node p is unnecessary. I-SATCHMO prunes away the process to be made on the remaining consequent atom v and backtracks to node p at once.

Now, we know that p is also not marked as useful in the reasoning, therefore the violated $\mathcal{FC}$ clause $true \rightarrow p; r$ selected at the root node $\{\mathcal{BC}\}$ is unnecessary. Accordingly, we stop to process the remaining consequent atom r .

In this way, the proof tree constructed by I-SATCHMO is shown in Figure 3. As we see, unnecessary $\mathcal{FC}$ clauses are split only once.

3 UNSEARCHMO

Since I-SATCHMO marks an asserted consequent atom as useful whenever it contributes to derive the antecedent of a violated $\mathcal{FC}$ clause, those consequent atoms contributed to unnecessary violated clauses are also marked as useful.

This problem can be easily solved by improving the marking work as follows: an asserted consequent atom is marked as useful only when it contributes to derive goal $\perp$ or the

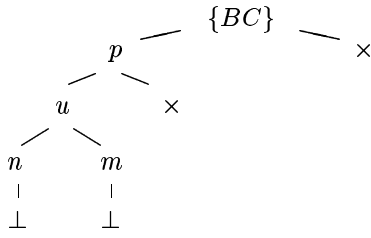


Figure 3: The proof tree PT constructed by I-SATCHMO in Example 2.3

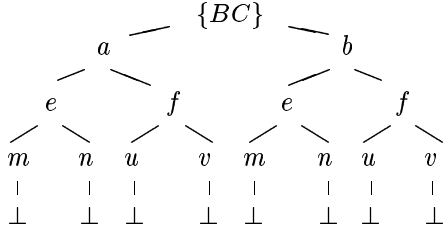


Figure 4: The proof tree PT constructed by I-SATCHMO for the clause set given in Example 3.1

antecedent of a violated $\mathcal{FC}$ clause that has been verified as necessary.

Moreover, without distinguishing the useful consequent atoms for different branches in proof tree, I-SATCHMO might mismark those consequent atoms that are useful for the refutation of unnecessary branches in a proof tree but useless for the necessary branches as “useful”. In such cases, I-SATCHMO also possibly makes some redundant search. Let us see the following example.

Example 3.1 Consider the following propositional clause set $S = BC \cup \mathcal{FC}$.

$BC : m \rightarrow \perp. \quad n \rightarrow \perp. \quad u \rightarrow \perp. \quad v \rightarrow \perp.$

$\mathcal{FC} : \text{true} \rightarrow a; b. \quad \text{true} \rightarrow e; f. \quad a, e \rightarrow m; n. \\ b, e \rightarrow m; n. \quad \text{true} \rightarrow u; v.$

The proof tree generated by I-SATCHMO is shown in Figure 4.

Consider the leftmost node labeled with e , where $\mathcal{I}_e = \{a, e\}$, the clause $a, e \rightarrow m; n$ becomes violated and is used to forward chaining. Since both m and n are useful for deriving $\perp$, the use of the clause is found to be necessary to the refutation, a and e are marked as useful. In this way, the process for the consequent atom f (the sibling atom of e) and b (the sibling atom of a) are made, respectively.

Obviously, the process for node b in the above example is unnecessary. Although the consequent atom a is useful for each branch from node e , it is useless for any branch from node f . It means that the refutation at node f can be made without the consequent atom a , that is, that can be made on the root node $\{BC\}$ in the same way. Therefore, the redundant process for the consequent atom b can be eliminated.

Although the process made at the leftmost node e is also unnecessary in a strict meaning, it is unavoidable unless we

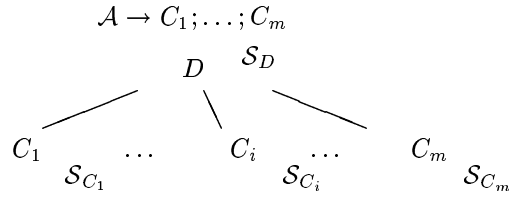


Figure 5: Finding the useful consequent atom set for parent node from those of its children nodes

select atom f first to process at node a or we first select the last non-Horn clause for forward chaining. It is the well-known nondetermination problem for theorem proving.

Now, we consider how to distinguish and manage the useful consequent atoms for different branches in a proof tree. Without the loss of generality, shown as in Figure 5, suppose that $A \rightarrow C_1; \dots; C_m$ be the violated clause selected at node D and $\mathcal{S}_{C_i}$ be the useful consequent atom set relative to C_i for all i such that $1 \leq i \leq m$. We consider how to derive the useful consequent atom set $\mathcal{S}_D$ relative to node D .

Since our reasoning procedure is depth-first one, C_{i+1} is not taken into processing until node C_i has been proven to be unsatisfiable. Moreover, only when some of C_i is found to be useless or all node C_i ($1 \leq i \leq m$) are proven to be unsatisfiable, their parent node D is proven to be unsatisfiable.

As we will know from the algorithm of UNSEARCHMO, when node C_i is proven to be unsatisfiable, the useful consequent atom set $\mathcal{S}_{C_i}$ for deriving the refutation at node C_i has already been established. The useful consequent atom set $\mathcal{S}_D$ for node D is constructed according to the way how node D is proven as unsatisfiable.

If node D is proven as unsatisfiable in the way that all of C_i ($1 \leq i \leq m$) are proven as unsatisfiable, then the use of the clause $A \rightarrow C_1; \dots; C_m$ is found to be necessary. In order to complete the refutation at node D , the clause should be ably found to be violated there. Therefore, the useful consequent atoms to derive the antecedent A are useful for our refutation. Moreover, in order to complete the refutation made at each node C_i , each useful consequent atom for the refutation made at node C_i , except C_i itself (since it can be obtained by splitting $A \rightarrow C_1; \dots; C_m$ at node D), is also needed. Concludingly, in this case, the useful consequent atom set $\mathcal{S}_D$ can be derived as follows.

$$\mathcal{S}_D = \{\mathcal{S}_{C_1} - \{C_1\}\} \cup \dots \cup \{\mathcal{S}_{C_m} - \{C_m\}\} \cup \mathcal{S}_A \quad (1)$$

where $\mathcal{S}_A$ is the useful consequent atom set for deriving the antecedent A .

On the other hand, if node D is proven as unsatisfiable in the way that some of C_i ($1 \leq i \leq m$) is found to be useless, then the use of the clause $A \rightarrow C_1; \dots; C_m$ is found to be unnecessary for the refutation made on node C_i , i.e., $C_i \notin \mathcal{S}_{C_i}$. In other words, the refutation found at node C_i can also be made at node D without using the clause $A \rightarrow C_1; \dots; C_m$. Thus, the useful consequent atom set $\mathcal{S}_D$ for deriving a refutation at node D is exactly the same as $\mathcal{S}_{C_i}$, i.e.,

$$\mathcal{S}_D = \mathcal{S}_{C_i} \quad (2)$$

In any case, when $\mathcal{S}_D$ is derived, all of $\mathcal{S}_{C_i}$ ($1 \leq i \leq m$) are removed from database.

Let us make a comparison of I-SATCHMO's method and our new strategy. When node D is proven as unsatisfiable in the way that all of C_i are proven as unsatisfiable, the two methods are exactly similar, the useful consequent atom set $\mathcal{S}_D$ is obtained according to the formula (1). However, when node D is proven as unsatisfiable in the way that node C_i is found to be useless, the useful consequent atom set $\mathcal{S}_D$ derived by I-SATCHMO is as follows.

$$\mathcal{S}_D = \{\mathcal{S}_{C_1} - \{C_1\}\} \cup \dots \cup \{\mathcal{S}_{C_{i-1}} - \{C_{i-1}\}\} \cup \mathcal{S}_{C_i} \quad (3)$$

Obviously, although the consequent atoms contained in $\{\mathcal{S}_{C_1} - \{C_1\}\}, \dots, \{\mathcal{S}_{C_{i-1}} - \{C_{i-1}\}\}$ are useful for the branches beginning from $C_1, \dots, C_{i-1}$, but useless for the refutation made at node C_i , and therefore is also useless to prove D unsatisfiable.

As we have seen, if a consequent atom, except the last atom of the consequence, of a violated $\mathcal{FC}$ clause is found to be useful, then its next sibling atom will be taken into processing. Therefore, the fewer the useful consequent atoms are, the more efficient a refutation is. It is quite clear that the useful consequent atom set for any node derived by our new strategy cannot be reduced any further, i.e., it is a minimal one. Since we can keep a minimal useful consequent atom set for a refutation, our new strategy is one of the most efficient approaches for forward chaining theorem proving.

The algorithm of our prover UNSEARCHMO to construct PT for checking whether a given clause set $\mathcal{S} = \mathcal{BC} \cup \mathcal{FC}$ is unsatisfiable can be described as follows.

For the current node D (initially the root node $\{\mathcal{BC}\}$) and its corresponding consequent atom set $\mathcal{I}_D$ (initially empty):

1. If $\mathcal{BC} \cup \mathcal{I}_D \vdash \perp$, create a leaf node $\perp$ below node D , which means node D is proven unsatisfiable. The useful consequent atom set for node D , $\mathcal{S}_D$, that consists of those consequent atoms listed in the successful deriving leaf in $DT_{\perp}^{\mathcal{BC} \cup \mathcal{I}_D}$, is established.
2. If $\mathcal{BC} \cup \mathcal{I}_D \not\vdash \perp$, select an instantiated ground violated clause $\mathcal{A} \rightarrow C_1; \dots; C_m$ from $\mathcal{FC}$. If no such clause exists, $\mathcal{BC} \cup \mathcal{FC}$ is satisfiable.
3. For the selected violated non-Horn clause $\mathcal{A} \rightarrow C_1; \dots; C_m$, create a node C_i for all i such that $1 \leq i \leq m$. Taking C_i as the new current node, call this procedure recursively in depth-first strategy with the augmented consequent atom set $\mathcal{I}_{C_i} = \mathcal{I}_D \cup \{C_i\}$. However, if C_i is found to be useless on backtracking, instead of each node C_j such that $i < j \leq m$, create a special node $\times$. The useful consequent atom set for node D , $\mathcal{S}_D$, is established according to the formula (2). On the other hand, if every C_i for $1 \leq i \leq m$ is found to be useful, the useful consequent atom set $\mathcal{S}_D$ is established according to the formula (1). In either case, node D is proven unsatisfiable.

Example 3.2 Suppose that $\mathcal{S} = \mathcal{BC} \cup \mathcal{FC}$ be the clause set given in in Example 2.1. The proof tree constructed by UNSEARCHMO is shown in Figure 6. Redundant search space has been eliminated.

The proof of the correctness of our UNSEARCHMO is not difficult. Since UNSEARCHMO just cuts down some branches in the proof tree constructed by SATCHMO, if SATCHMO can show a given clause set unsatisfiable, then UNSEARCHMO can certainly show the same thing. This shows the completeness of UNSEARCHMO.

On the other hand, as we have indicated above, an asserted consequent atom D is found useless at the time being retracted from the reasoning database, its parent node is able to be proven as unsatisfiable. Therefore, all remaining process to show its parent node unsatisfiable can be removed without the loss of the soundness. The further details are omitted here since the lack of space.

It is easy to implement UNSEARCHMO by Prolog. The code is omitted here also because of the lack of space.

4 An Example

Example 4.1 Let $\mathcal{S}$ be the following clause set, an extension of the benchmark problem SYN009-1 given in TPTP problem library [Sutcliffe and Sutner, 2000].

$$\begin{aligned} \mathcal{BC} : \quad & p(X, Y, Z), m(X, Y, Z) \rightarrow \perp. \\ & q(X, Y, Z), m(X, Y, Z) \rightarrow \perp. \\ & r(X, Y, Z), m(X, Y, Z) \rightarrow \perp. \\ & n(c, c, c) \rightarrow \perp. \\ & s(a). \quad s(b). \quad s(c). \end{aligned}$$

$$\begin{aligned} \mathcal{FC} : \quad & s(X), s(Y), s(Z) \\ & \rightarrow p(X, Y, Z); q(X, Y, Z); r(X, Y, Z). \\ & s(X), s(Y), s(Z) \rightarrow m(X, Y, Z); n(X, Y, Z). \end{aligned}$$

The first $\mathcal{FC}$ clause will be fully split by any of SATCHMO, SATCHMORE and $\mathcal{A}$ -SATCHMORE. That means more than 3^{27} cases will be considered. As a result, we gave up the our tests on SATCHMO, SATCHMORE and $\mathcal{A}$ -SATCHMORE with a run of three days without termination on an Intel PentiumIII/650MHZ workstation, respectively.

Now, we see how I-SATCHMO and UNSEARCHMO work. Initially, the first $\mathcal{FC}$ clause is used for forward chaining with each substitution such that its antecedent becomes satisfiable. In this way, a branch $\{\mathcal{BC}\} \rightarrow p(a, a, a) \rightarrow p(a, a, b) \rightarrow p(a, a, c) \rightarrow \dots \rightarrow p(c, c, c)$ is generated.

At node $p(c, c, c)$, the second $\mathcal{FC}$ clause with the substitution $\{X = a, Y = a, Z = a\}$, i.e., $s(a), s(a), s(a) \rightarrow m(a, a, a); n(a, a, a)$, is first split. For node $m(a, a, a)$, $\perp$ can be derived and the useful consequent atom set is established as $\{p(a, a, a), m(a, a, a)\}$. At node $n(a, a, a)$, $\perp$ can

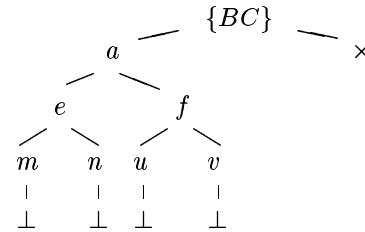


Figure 6: The proof tree PT constructed by UNSEARCHMO for the clause set given in Example 3.1

not be derived, and the second $\mathcal{FC}$ clause with the substitution $\{X = a, Y = a, Z = b\}$, i.e., $s(a), s(a), s(b) \rightarrow m(a, a, b); n(a, a, b)$, is split. At node $m(a, a, b)$, similar as node $m(a, a, a)$, $\perp$ is derived and the useful consequent atom set is $\{p(a, a, b), m(a, a, b)\}$. For node $n(a, a, b)$, similar as node $n(a, a, a)$, no refutation can be derived, then the second non-Horn clause will be split with the substitution $\{X = a, Y = a, Z = c\}$, and so on.

The second non-Horn clause will be further split until $s(c), s(c), s(c) \rightarrow m(c, c, c); n(c, c, c)$ is used for forward chaining at node $n(b, c, c)$. Then, at last a refutation can be derived from each of node $n(b, c, c)$'s child node simultaneously. The useful consequent atom set for the refutation at node $m(c, c, c)$ is $\{p(c, c, c), m(c, c, c)\}$ and that at node $n(c, c, c)$ is just $\{n(c, c, c)\}$. According to the formula (1), the useful consequent atom set for node $n(b, c, c)$ is constructed as $\{p(c, c, c)\}$.

If we do not distinguish the useful consequent atoms for different branches, as I-SATCHMO does, we will find that all p -facts have been marked as useful. Then, the same process made on each p -fact $p(x, y, z)$, where each of x, y and z is one of a, b and c , would be repeated for the corresponding q -fact $q(x, y, z)$ (also for the corresponding r -fact $r(x, y, z)$). Similar to other existing strategies, no answer can be obtained within a reasonable time.

However, at node $n(b, c, c)$, where the useful consequent atom set is $\{p(c, c, c)\}$, UNSEARCHMO finds $n(b, c, c)$ useless for the refutation, i.e., the clause $s(b), s(c), s(c) \rightarrow m(b, c, c); n(b, c, c)$ selected at node $n(b, c, c)$'s parent $n(a, c, c)$ is not necessary. According to the formula (1), the useful consequent atom set derived at its sibling node $m(b, c, c)$, i.e., $\{p(b, c, c), m(b, c, c)\}$, is abandoned, and the useful consequent atom set for node $n(a, c, c)$ is still $\{p(c, c, c)\}$.

Similar backtracking repeatedly continues to node $p(c, c, c)$, where the useful consequent atom set established by UNSEARCHMO is still $\{p(c, c, c)\}$. The same process made on node $p(c, c, c)$ will be also made on its sibling node $q(c, c, c)$ and node $r(c, c, c)$, and the corresponding useful consequent atom set are found to be $\{q(c, c, c)\}$ and $\{r(c, c, c)\}$, respectively. The violated clause $s(c), s(c), s(c) \rightarrow p(c, c, c); q(c, c, c); r(c, c, c)$ selected at node $p(b, c, c)$ is found to be necessary.

Now, let us see what happens when UNSEARCHMO backtracks to $p(c, c, c)$'s parent node $p(b, c, c)$, where, according to the formula (1), useful consequent atom set is surprisingly found to be empty! no other consequent atom is useful for our refutation! UNSEARCHMO simply backtracks to the root node $\{BC\}$ to conclude that the given clause set is unsatisfiable.

With the help of distinguishing the useful consequent atoms for different branches, UNSEARCHMO only takes 0.002 seconds to find its solution on an Intel Pentium-III/650MHZ workstation.

5 Conclusion

In this paper, we have introduced how to eliminate redundant search space on backtracking for forward chaining theorem

proving as much as possible. The redundant search branches are immediately pruned away when unnecessary ones are found on backtracking. At any reasoning point, only those necessary violated clauses are completely split for forward chaining. Repeated search by invoking unnecessary non-Horn clauses can be effectively eliminated. The experimental result has shown that our method is powerful for forward chaining theorem proving.

Since our strategy prunes away unnecessary proof branches after unnecessary non-Horn clauses are used for forward chaining, the violated non-Horn clauses for forward chaining can be used in any order decided by any advance checking strategy. That is, we can incorporate the relevancy checking proposed in SATCHMORE and the availability testing introduced in $\mathcal{A}$ -SATCHMORE into our strategies to improve the performance further.

It is obvious that our method can be applied to disjunctive logic programming (database). The principle of our method to eliminate redundant search space in forward chaining based prover can also be applied to reason about other logics whenever forward chaining strategy is used. This gives our method a wide application in automated reasoning field.

As future works, we will use our prover to test those benchmark problems provided in TPTP library [Sutcliffe and Suttner, 2000], to clear which class of problems will be helped by our approach.

References

- [Bry and Yahya, 2000] Bry, F. and Yahya, A.: Positive Unit Hyperresolution Tableaux and Their Application to Minimal Model Generation. *Journal of Automated Reasoning*, 25:35-82, 2000.
- [He et al., 1998] He, L., Chao, Y., Simajiri, Y., Seki, H. and Itoh, H.: $\mathcal{A}$ -SATCHMORE: SATCHMORE with Availability Checking. *New Generation Computing*, 16:55-74, 1998.
- [He, 2001] He, L.: I-SATCHMO: an Improvement of SATCHMO. *To appear in J. of automated reasoning*.
- [Loveland, 1968] Loveland, D.W.: Mechanical Theorem Proving by Model Elimination. *J. of the ACM*, 15:236-251, 1968.
- [Loveland et al., 1995] Loveland, D.W., Reed, D.W. and Wilson, D.S.: SATCHMORE: SATCHMO with Relevancy. *Journal of Automated Reasoning*, 14:325-351, 1995.
- [Manthey and Bry, 1988] Manthey, R. and Bry, F.: SATCHMO: a theorem prover implemented in Prolog. In *Proceedings of 9th intl. Conf. on Automated Deduction*, 1988.
- [Ramsay, 1991] Ramsay, A.: Generating Relevant Models. *Journal of Automated Reasoning*, 7:359-368, 1991.
- [Robinson, 1965] Robinson, J.A.: A Machine-oriented logic based on the resolution principle. *J. of Ass. comput. Mach.*, 12, 23-41, 1965.
- [Sutcliffe and Suttner, 2000] Sutcliffe, G. and Suttner, C.: <http://www.cs.jcu.edu.au/~tptp/>