Chao YY, He LF, Nakamura T *et al.* An improvement of Herbrand's theorem and its application to model generation theorem proving. JOURNAL OF COMPUTER SCIENCE AND TECHNOLOGY 22(4): 541~553 July 2007

An Improvement of Herbrand's Theorem and Its Application to Model Generation Theorem Proving

Yu-Yan Chao[1,2] (巢宇燕), Li-Feng He[3,4,*] (何立风), Tsuyoshi Nakamura[5] (中村剛士), Zheng-Hao Shi[5] (石争浩), Kenji Suzuki[6] (鈴木賢治), and Hidenori Itoh[5] (伊藤英則)

[1] *Graduate School of Environment Management, Nagoya Sangyo University, Aichi, Japan*

[2] *College of Mechanical and Electrical Engineering, Shaanxi University of Science and Technology, China*

[3] *Graduate School of Information Science and Technology, Aichi Prefectural University, Aichi, Japan*

[4] *College of Computer and Information Engineering, Shaanxi University of Science and Technology, China*

[5] *Graduate School of Computer Science and Engineering, Nagoya Institute of Technology, Nagoya, Japan*

[6] *Department of Radiology, The University of Chicago, Chicago, U.S.A.*

E-mail: chao@nagoya-su.ac.jp; helifeng@ist.aichi-pu.ac.jp; {tnaka, zhshi, itoh}@ics.nitech.ac.jp; suzuki@uchicago.edu

Received June 21, 2006; revised December 21, 2006.

Abstract This paper presents an improvement of Herbrand's theorem. We propose a method for specifying a sub-universe of the Herbrand universe of a clause set $\mathcal{S}$ for each argument of predicate symbols and function symbols in $\mathcal{S}$. We prove that a clause set $\mathcal{S}$ is unsatisfiable if and only if there is a finite unsatisfiable set of ground instances of clauses of $\mathcal{S}$ that are derived by only instantiating each variable, which appears as an argument of predicate symbols or function symbols, in $\mathcal{S}$ over its corresponding argument's sub-universe of the Herbrand universe of $\mathcal{S}$. Because such sub-universes are usually smaller (sometimes considerably) than the Herbrand universe of $\mathcal{S}$, the number of ground instances may decrease considerably in many cases. We present an algorithm for automatically deriving the sub-universes for arguments in a given clause set, and show the correctness of our improvement. Moreover, we introduce an application of our approach to model generation theorem proving for non-range-restricted problems, show the range-restriction transformation algorithm based on our improvement and provide examples on benchmark problems to demonstrate the power of our approach.

Keywords Herbrand's theorem, Herbrand universe, model generation theorem proving, SATCHMO, really non-propositional

1 Introduction

Herbrand's theorem[1] is the basis for most modern automatic proof procedures in mechanical first-order theorem proving. By Herbrand's theorem, for a given clause set $\mathcal{S}$, a special universe, called *Herbrand universe*, can be mechanically created. $\mathcal{S}$ is unsatisfiable if and only if there is an unsatisfiable set of ground instances of clauses of $\mathcal{S}$, where a ground instance of a clause is derived by instantiating variables in the clause with elements of the Herbrand universe of $\mathcal{S}$.

Herbrand's theorem enables us to make theorem proving mechanically. However, theorem proving methods based directly on Herbrand's theorem, e.g., the multiplication method[2], are usually inefficient, because there may be too many ground instances need to be considered. Therefore, various efficient strategies were developed to enhance their performances. For example, SATCHMO[3,4], a model generation (forward chaining) theorem prover, is usually inefficient in cases where variables in a given clause set $\mathcal{S}$ need to be instantiated over the Herbrand universe of $\mathcal{S}$[5], and was improved in various ways[6~12].

In this paper, we propose an improvement on Herbrand's theorem. Instead of establishing the standard Herbrand universe for a given clause set $\mathcal{S}$, we specify a *sub-universe* of the Herbrand universe for each argument of predicates or functions in $\mathcal{S}$. We prove that $\mathcal{S}$ is unsatisfiable if and only if there is a finite unsatisfiable set of ground instances of clauses of $\mathcal{S}$ derived by instantiating each variable, which appears as an argument of predicate symbols or function symbols, in $\mathcal{S}$ over its corresponding arguments' sub-Herbrand universes. Because such sub-universes are usually smaller (sometimes considerably) than the Herbrand universe of $\mathcal{S}$, the number of ground instances need to be considered for reasoning can be reduced in many cases. For model generation theorem proving, this means that the number of forward chaining clauses used for reasoning is reduced, thus, leading to efficiency.

Some methods were proposed to reduce the number of ground instances used for theorem proving. Schulz[13] presented a method to reduce the number of the necessary instances in cases where Herbrand universes are finite. Lee and Plaisted[14] presented the so-called Hyper-Linking Strategy to avoid generating unnecessary instances during theorem proving. Yu *et al.*[15] developed a heuristic method to cut off branches unnecessary for reasoning. Our method can be considered as a generalization of such approaches.

The rest of the paper is organized as follows. In the next section, we introduce terminology and notations,

Regular Paper

This work was supported partially by TOYOAKI Scholarship Foundation, Japan.

*Currently, Research Associate, The University of Chicago, USA.

review Herbrand's theorem and resolution principle, define range-restricted clauses, and recall SATCHMO, the basic model generation theorem prover. Section 3 considers how to specify sub-Herbrand universes for arguments of predicate symbols and function symbols in a clause set, and Section 4 shows the correctness of our approach. We introduce our range-restriction transformation algorithm in Section 5, and report the experimental results on benchmarks in Section 6. Lastly, we give our conclusion in Section 7.

2 Preliminaries and Background Material

We assume familiarity with the basic concepts relating to first-order theorem proving, e.g., presented in [16, 17], and limit ourselves to briefly recalling the basic material needed for our presentation.

2.1 Terminology and Notations

In this paper, a clause $\neg A_1 \vee \cdots \vee \neg A_n \vee C_1 \vee \cdots \vee C_m$ is also represented by means of *positive implicational form* $A_1, \ldots, A_n \rightarrow C_1; \ldots; C_m$ $(n, m \geqslant 0)$. We refer to the implicit conjunction $A_1, \ldots, A_n$ as the *antecedent* of the clause, with each A_i being an *antecedent atom*. The implicit disjunction $C_1; \ldots; C_m$ is referred to as the *consequent* of the clause, and each C_j a *consequent atom*. A clause with an empty consequent, i.e., $m = 0$, is written as $A_1, \ldots, A_n \rightarrow \bot$, where $\bot$ means *falsity*. On the other hand, for a clause with an empty antecedent, i.e., $n = 0$, the antecedent atom $\top$, that indicates *truth*, is added.

The lower-case letters are used to represent predicate symbols, function symbols and constants, while the upper-case letters are used for variables. On the other hand, the Greek letters are used to represent arbitrary predicate symbols, function symbols, terms, substitutions, and other necessary information.

A predicate (function) α with n arguments is called *n-place predicate* (*function*), and the i-th $(1 \leqslant i \leqslant n)$ argument of α is denoted to $\alpha^n\langle i \rangle$. $\emptyset$ is denoted the empty set, $A \in \mathcal{I}$ means that A is a member of $\mathcal{I}$. Moreover, we view clauses as sets and assume that there is no the same variable symbol in different clauses of the set of clauses under consideration.

2.2 Herbrand's Theorem

Definition 1 (Herbrand Universe). *Let $\mathcal{S}$ be a set of clauses, $\mathcal{C}$ and $\mathcal{F}$, the sets of constants and function symbols occurring in $\mathcal{S}$, respectively. Let*

$$\mathcal{H}_0 = \begin{cases} \mathcal{C}, & \text{if } \mathcal{C} \neq \emptyset; \\ \{a\}, & \text{if } \mathcal{C} = \emptyset; \end{cases}$$

where a is an artificial constant, and for $i = 0, 1, 2, \ldots$,

$$\mathcal{H}_{i+1} = \mathcal{H}_i \cup \{f(\tau_1, \ldots, \tau_n) | f \in \mathcal{F}, \tau_j \in \mathcal{H}_i, \quad 1 \leqslant j \leqslant n\}.$$

Then, $\mathcal{H}_\infty$, or $\lim_{i\to\infty} \mathcal{H}_i$, is called the Herbrand universe of $\mathcal{S}$.

According to the above definition, the Herbrand universe of $\mathcal{S}$ depends only on the constants and functions in $\mathcal{S}$. In other words, all clause sets that contain the same constants and functions have the same Herbrand universe, regardless their occurring positions and predicates in the clause sets. Notice that, if a clause set contains a function, then its Herbrand universe is infinite.

Example 1. Let $\mathcal{S}_1$ be the following set of clauses:

$$\top \rightarrow p(f(a), c).$$

Let $\mathcal{S}_2$ be the following set of clauses:

$$p_1(c) \rightarrow \bot,$$
$$p_2(a) \rightarrow \bot,$$
$$p_1(X) \rightarrow p_2(f(X)),$$
$$\top \rightarrow p_1(Y); p_2(Y).$$

Because $\mathcal{S}_1$ and $\mathcal{S}_2$ contain exactly the same constants and functions, they have the same Herbrand universe: $H = \{a, c, f(a), f(c), f(f(a)), f(f(c)), f(f(f(a))), f(f(f(c))), \ldots\}$.

Definition 2 (Ground Atom and Ground Instance). *An atom or a clause is said to be ground if it contains no variable. A ground instance of a clause C of a set $\mathcal{S}$ of clauses is a clause obtained by replacing variables in C by elements of the Herbrand universe of $\mathcal{S}$.*

Theorem 1 (Herbrand's Theorem). *A set $\mathcal{S}$ of clauses is unsatisfiable if and only if there is a finite unsatisfiable set of ground instances of clauses of $\mathcal{S}$.*

2.3 Resolution Principle

An atom A or the negation of an atom $\neg A$ is called *literal*, and they are said to be each other's *complement*.

Definition 3 (Factor). *If two or more literals with the same sign of a clause C have a most general unifier σ, then $C\sigma$ is called a* factor *of C.*

Definition 4 (Binary Resolvent). *Let C_1 and C_2 be two clauses, L_1 and L_2, two literals that can be unified to complementary literals by a most general unifier σ in C_1 and C_2. Then the clause*

$$(C_1\sigma - L_1\sigma) \cup (C_2\sigma - L_2\sigma)$$

is called a binary resolvent of C_1 and C_2. The literals L_1 and L_2 are called the literals resolved upon.

Definition 5 (Resolution). *Let $\mathcal{S}$ be a set of clauses, and $\mathcal{T} = \mathcal{S}$. Resolution extends $\mathcal{T}$ by adding a following clause recursively:*

1. *a factor*[①] *of a clause in $\mathcal{T}$,*
2. *a binary resolvent of C_1 and C_2, where C_1, C_2 are two arbitrary clauses in $\mathcal{T}$.*

[①] For convenience, we consider a factor as a derived clause in resolution.

A set $\mathcal{S}$ of clauses is unsatisfiable if and only if the empty clause can be derived by resolution[17].

2.4 Range-Restriction

Definition 6 (Range-Restriction). *A variable is said to be range-restricted if it occurs in the antecedent of a clause. Contrastingly, a variable that only occurs in the consequent of a clause is said to be non-range-restricted. A clause is said to be range-restricted if all of its variables are range-restricted. A set of clauses is range-restricted if all of its clauses are range-restricted.*

Obviously, if the antecedent of a range-restricted clause is satisfied (or instantiated) by a set of ground atoms, then every consequent atom of the clause is ground.

A non-range-restricted clause set can be transformed into a range-restricted one, which is satisfiability equivalent to the original one[3,4].

Definition 7 (Range-Restriction Transformation Algorithm). *Let $\mathcal{S}$ be a non-range-restricted clause set. Its range-restriction form $\mathcal{S}'$ is transformed from $\mathcal{S}$ as follows*:

1. *Every range-restricted clause $\mathcal{A} \rightarrow \mathcal{C}$ is transformed into itself.*

2. *Every non-range-restricted clause $\mathcal{A} \rightarrow \mathcal{C}$ that contains non-range-restricted variables $X_1, \ldots, X_n$ is transformed into $\mathcal{A}$, $dom(X_1), \ldots, dom(X_n) \rightarrow \mathcal{C}$. However, $\mathcal{A}$ is omitted if it is $\top$.*

3. *For every constant c in $\mathcal{S}$, a clause $\top \rightarrow dom(c)$ is added. If $\mathcal{S}$ contains no constant, a single clause $\top \rightarrow dom(a)$ is added, where a is an artificial constant.*

4. *For every m-place function symbol f occurring in $\mathcal{S}$, a clause $dom(Y_1), \ldots, dom(Y_m) \rightarrow dom(f(Y_1, \ldots, Y_m))$ is added.*

In Definition 7, *dom* is an introduced unary predicate symbol. Obviously, by the range-restriction transformation, non-range-restricted variables are instantiated over the Herbrand universe of the set of clauses under consideration.

Example 2. Let $\mathcal{S}$ be the following non-range-restricted clause set:

$$p_1(c) \rightarrow \bot,$$
$$p_2(f(c)) \rightarrow \bot,$$
$$\top \rightarrow p_1(X); p_2(X).$$

The transformed range-restricted clause set $\mathcal{S}'$ of $\mathcal{S}$ is:

$$p_1(c) \rightarrow \bot,$$
$$p_2(f(c)) \rightarrow \bot,$$
$$dom(X) \rightarrow p_1(X); p_2(X),$$
$$\top \rightarrow dom(c),$$
$$dom(Y) \rightarrow dom(f(Y)).$$

The non-range-restricted variable X in $\mathcal{S}$ is instantiated over the whole Herbrand universe of $\mathcal{S}$ in $\mathcal{S}'$.

2.5 SATCHMO

SATCHMO[3] is a model-generation based theorem prover applicable to the class of range-restricted clause sets.

Suppose that $\mathcal{M}$ be a set of ground atoms, then a conjunction (disjunction) of ground atoms is *satisfied* in $\mathcal{M}$ if all (some) of its members belong to $\mathcal{M}$.

Let $\mathcal{S}$ be a set of clauses. For a ground instance $\mathcal{A}_g \rightarrow \mathcal{C}_g$ of clause $\mathcal{A} \rightarrow \mathcal{C}$ in $\mathcal{S}$, $\mathcal{A}_g \rightarrow \mathcal{C}_g$ is said to be *satisfied* in $\mathcal{M}$, if $\mathcal{C}_g$ is satisfied or $\mathcal{A}_g$ is not satisfied in $\mathcal{M}$, and else *violated*. If every ground instance of clauses in $\mathcal{S}$ is satisfied in $\mathcal{M}$, then $\mathcal{S}$ is satisfiable and $\mathcal{M}$ is said to be a *model* of $\mathcal{S}$.

Accordingly, to check whether a given clause set $\mathcal{S}$ is satisfiable, SATCHMO goes to construct a model for the clause set: from the empty model candidate $\mathcal{M} = \emptyset$, it satisfies each violated clause in $\mathcal{M}$ by adding one of the consequent atoms of the clause into the model candidate $\mathcal{M}$ (in this way, the violated clause becomes satisfiable in $\mathcal{M}$) in turn. If it succeeds with no violated clause in $\mathcal{M}$ anymore, $\mathcal{M}$ is a model of $\mathcal{S}$. Then, the given clause set is proved to be satisfiable. Otherwise, if we run out of possibilities to establish a model for $\mathcal{S}$, then $\mathcal{S}$ is unsatisfiable.

Given a clause set $\mathcal{S}$, the reasoning procedure of SATCHMO can be graphically illustrated by constructing a proof tree as follows.

Definition 8 (SATCHMO). *Starting from the root node $\top$, for the current node D, where D is an atom, let $\mathcal{I}_D$ be the set of atoms on the path from the root node to node D*:

1. *If D is $\bot$, the branch is said to be closed, which means that $\mathcal{S} \cup \mathcal{I}_D$ is unsatisfiable.*

2. *If D is not $\bot$, select a ground clause from $\mathcal{S}$ that is violated in $\mathcal{I}_D$.* ② *If no such clause exists, $\mathcal{I}_D$ is a model of $\mathcal{S}$, $\mathcal{S}$ is satisfiable.*

3. *Let $\mathcal{A}\sigma \rightarrow C_1\sigma; \ldots; C_m\sigma$ be the selected violated clause, where σ is a ground institution. For each ground consequent atom $C_i\sigma$ in the clause, create a node $C_i\sigma$ under node D. Then, take node $C_i\sigma$ as the current node, call this procedure recursively in the depth-first strategy.*

Let $\mathcal{S}$ be a set of clauses. For a node D in a proof tree of $\mathcal{S}$, if all branches below node D are closed, it means that $\mathcal{I}_D$ cannot be extended to a model of $\mathcal{S}$, i.e., $\mathcal{S} \cup \mathcal{I}_D$ is unsatisfiable. If all branches below the root node are closed, then $\mathcal{S}$ is unsatisfiable (at root node, $\mathcal{I}_D = \emptyset$).

If a clause set $\mathcal{S}$ is non-range-restricted, before applying it to SATCHMO, we should transform $\mathcal{S}$ to its corresponding range-restricted one, $\mathcal{S}'$. Because non-range-restricted variables in $\mathcal{S}$ are instantiated over the

② To find a ground clause violated in $\mathcal{I}_D$ from $\mathcal{S}$, for each clause $\mathcal{A} \rightarrow \mathcal{C}$ in $\mathcal{S}$, we unify $\mathcal{A}$ with elements of $\mathcal{I}_D$. If it succeeded with ground unifier σ (i.e., $\mathcal{A}\sigma$ is satisfied in $\mathcal{I}_D$.) such that $\mathcal{C}\sigma$ is not satisfied in $\mathcal{I}_D$, then $\mathcal{A}\sigma \rightarrow \mathcal{C}\sigma$ is a ground clause violated in $\mathcal{I}_D$. Otherwise, other unification will be tested.

whole Herbrand universe in $\mathcal{S}'$, some (maybe infinite) instantiated clauses introduced by range-restriction transformation might be unwanted redundant for reasoning, but would be used for reasoning. Therefore, model generation based theorem provers are often inefficient on $\mathcal{S}'$[5~7].

Example 3. Let $\mathcal{S}$ and $\mathcal{S}'$ be the clause sets and the range-restricted form of $\mathcal{S}$ derived in Example 2, respectively.

The proof tree of $\mathcal{S}'$ constructed by SATCHMO is shown in Fig.1.

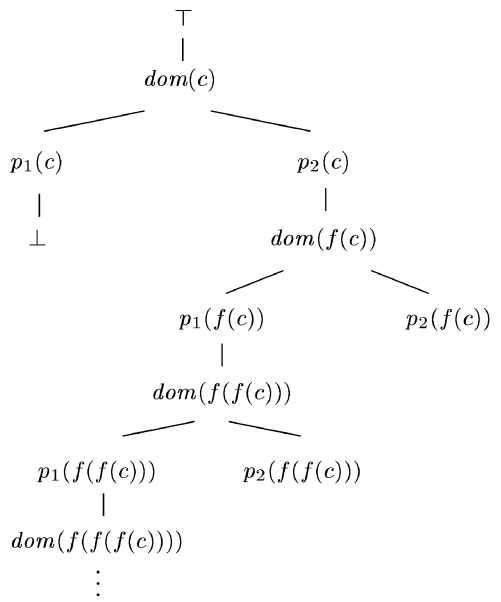

Fig.1. Proof tree constructed by SATCHMO in Example 3.

The proof of SATCHMO on $\mathcal{S}'$ does not terminate. The branch $\top \to dom(c) \to p_2(c)$, $dom(f(c)) \to p_1(f(c))$, $dom(f(f(c))) \to p_1(f(f(c)))$, $dom(f(f(f(c)))) \to \cdots$ is open.

In the above example, the reason that SATCHMO failed to terminate is that the non-range-restricted variable X is instantiated over the whole Herbrand universe of $\mathcal{S}$. However, to prove $\mathcal{S}$ unsatisfiable, not all instances are necessary. In fact, we need only consider the instances c and $f(c)$ for X, other instances do not contribute to derive $\perp$, and therefore are unnecessary.

It would be nice if we could find the necessary instances for variables. That is just the motivation of this paper.

3 Specification of Sub-Herbrand Universes for Arguments of Predicate Symbols and Function Symbols

Let $\mathcal{S}$ be a set of clauses. Because variables in $\mathcal{S}$ occur as arguments of predicate symbols or function symbols, we generalize our work to specify a *sub-universe of the Herbrand universe* (*SHU* for short) for each argument in $\mathcal{S}$. For convenience, the fact that a term τ appears as a value of an argument $\alpha^n\langle i\rangle$ in $\mathcal{S}$ is denoted by $app(\tau, \alpha^n\langle i\rangle)$. For example, suppose that there is $p(f(a), X)$ in $\mathcal{S}$, then we have $app(a, f^1\langle 1\rangle)$, $app(f(a), p^2\langle 1\rangle)$, and $app(X, p^2\langle 2\rangle)$.

Definition 9 (Algorithm for Computing a Preliminary *SHU*). *Let $\mathcal{S}$ be a set of clauses, and $\alpha^n\langle i\rangle$, an argument in $\mathcal{S}$. The selected constant c is a constant arbitrarily selected from $\mathcal{S}$ if there is any, otherwise, it is an artificial constant.*

The preliminary SHU for $\alpha^n\langle i\rangle$ is a set $\mathcal{H}$ derived as follows:

Step 1. *Initially, set $\mathcal{H} = \emptyset$, $\mathcal{M} = \{\alpha^n\langle i\rangle\}$, and $\mathcal{N} = \emptyset$.*

Step 2. *If $\mathcal{M}$ is empty, then $\mathcal{H}$ is the preliminary SHU for $\alpha^n\langle i\rangle$. However, if $\mathcal{H}$ contains no constant, then $\mathcal{H} = \mathcal{H} \cup \{c\}$. On the other hand, if $\mathcal{M}$ is not empty, continue.*

Step 3. *Move the first element $\beta^m\langle j\rangle$ of $\mathcal{M}$ to $\mathcal{N}$.* [③] *For each $app(\varepsilon, \beta^m\langle j\rangle)$ in each clause C of $\mathcal{S}$:*

Case 1. *ε is a constant c. Let $\mathcal{H} = \mathcal{H} \cup \{c\}$;*

Case 2. *ε is a functional term with function symbol f. Let $\mathcal{H} = \mathcal{H} \cup \mathcal{V}\langle f\rangle$, where $\mathcal{V}\langle f\rangle$ is the possible value set corresponding to f, whose definition will be given later;*

Case 3. *ε is a variable X. For each argument $\gamma^u\langle k\rangle$ such that there exists* $\mathrm{app}(X, \gamma^u\langle k\rangle)$ *in C, add the argument $\gamma^u\langle k\rangle$ into $\mathcal{M}$ if $\gamma^u\langle k\rangle \notin \mathcal{M} \cup \mathcal{N}$.*

Step 4. *Go to Step* 2.

In the above algorithm, $\mathcal{M}$ is the set of arguments to be processed and $\mathcal{N}$ is the set of arguments having been processed. $\mathcal{M}$ is expanded only when Case 3 of Step 3 is executed. In that case, let $\beta^m\langle j\rangle$ be the argument being processed, and C a clause in which there is $app(X, \beta^m\langle j\rangle)$. A new argument $\gamma^u\langle k\rangle$ is added into $\mathcal{M}$ if and only if there is $app(X, \gamma^u\langle k\rangle)$ in C. Intuitively, by any substitution in resolution, $\beta^m\langle j\rangle$ and $\gamma^u\langle k\rangle$ in the clause always take the same term. Thus, the two arguments are considered to have the same *SHU*. This conclusion can be extended to all elements in $\mathcal{M} \cup \mathcal{N}$. Therefore, when the algorithm terminates, all arguments in $\mathcal{N}$ have the same *SHU*. Such arguments are called the *same SHU arguments*. Especially, for a variable X in a clause C, all arguments of $\alpha^n\langle i\rangle$ such that there is $app(X, \alpha^n\langle i\rangle)$ in C, called *the arguments corresponding to variable X*, are the same *SHU* arguments.

Because there are finitely many arguments of predicate symbols and function symbols, constants and clauses in a set of clauses, the above algorithm certainly finitely terminates. For convenience, we also use $\mathcal{H}(\alpha^n\langle i\rangle)$ to indicate the preliminary *SHU* corresponding to $\alpha^n\langle i\rangle$.

Moreover, *SHU*s derived in the above algorithm are said to be *preliminary* because there might be elements like $\mathcal{V}\langle f\rangle$, which need to be further extended (see Definition 11 later).

Example 4. Let $\mathcal{S}$ be the following set of clauses:

$$\neg p(f(a, X_1), X_2),$$

[③] For convenience to describe, if a set $\mathcal{M}$ is presented to be $\{\alpha_1, \ldots, \alpha_n\}$, then α_1 is said to be the first element of $\mathcal{M}$.

$$p(a, X_3) \vee p(X_4, X_3),$$
$$\neg p(X_5, X_6) \vee p(f(X_5, b), X_6).$$

The preliminary *SHU* for $p^2\langle 1\rangle$, denoted as $\mathcal{H}_1$, is calculated as follows:

1) Initially, $\mathcal{H}_1 = \emptyset$, $\mathcal{M}_1 = \{p^2\langle 1\rangle\}$, $\mathcal{N}_1 = \emptyset$.

2) Move the first element of $\mathcal{M}_1$, i.e., $p^2\langle 1\rangle$, to $\mathcal{N}_1$. Then, $\mathcal{M}_1 = \emptyset$ and $\mathcal{N}_1 = \{p^2\langle 1\rangle\}$.

There are five appearances of $p^2\langle 1\rangle$ in $\mathcal{S}$: $app(f(a, X_1), p^2\langle 1\rangle)$, $app(a, p^2\langle 1\rangle)$, $app(X_4, p^2\langle 1\rangle)$, $app(X_5, p^2\langle 1\rangle)$ and $app(f(X_5, b), p^2\langle 1\rangle)$.

For $app(f(a, X_1), p^2\langle 1\rangle)$ and $app(f(X_5, b), p^2\langle 1\rangle)$, both are Case 2 in Definition 9. Therefore, $\mathcal{H}_1 = \mathcal{H}_1 \cup \mathcal{V}\langle f\rangle = \mathcal{V}\langle f\rangle$.

For $app(a, p^2\langle 1\rangle)$, it is Case 1 in Definition 9, therefore, $\mathcal{H}_1 = \mathcal{H}_1 \cup \{a\} = \mathcal{V}\langle f\rangle \cup \{a\}$.

For $app(X_4, p^2\langle 1\rangle)$ and $app(X_5, p^2\langle 1\rangle)$, both are Case 3 in Definition 9. Because X_4 does not appear any elsewhere in C, nothing need to do. On the other hand, because there is $app(X_5, f^2\langle 1\rangle)$ in C, $f^2\langle 1\rangle$ is added into $\mathcal{M}_1$. Therefore, $\mathcal{M}_1 = \{f^2\langle 1\rangle\}$.

3) Move the first element of $\mathcal{M}_1$, i.e., $f^2\langle 1\rangle$, to $\mathcal{N}_1$. Then, $\mathcal{M}_1 = \emptyset$ and $\mathcal{N}_1 = \{p^2\langle 1\rangle, f^2\langle 1\rangle\}$.

There are two appearances of $f^2\langle 1\rangle$ in $\mathcal{S}$: $app(a, f^2\langle 1\rangle)$ and $app(X_5, f^2\langle 1\rangle)$.

For $app(a, f^2\langle 1\rangle)$, it is Case 1 in Definition 9. However, because $a \in \mathcal{H}_1$ is already true, $\mathcal{H}_1$ does not change.

For $app(X_5, f^2\langle 1\rangle)$, it is Case 3 in Definition 9. There is $app(X_5, p^2\langle 1\rangle)$ in $\mathcal{S}$. However, because $p^2\langle 1\rangle \in \mathcal{M} \cup \mathcal{N}$, nothing need to do.

4) Because $\mathcal{M}_1 = \emptyset$, by Step 2 in Definition 9, the calculation of the preliminary *SHU* for $p^2\langle 1\rangle$ terminates with $\mathcal{H}_1 = \mathcal{V}\langle f\rangle \cup \{a\}$, and $\mathcal{N}_1 = \{p^2\langle 1\rangle, f^2\langle 1\rangle\}$.

On the other hand, the preliminary *SHU* for $p^2\langle 2\rangle$, denoted as $\mathcal{H}_2$, is calculated as follows:

1) Initially, $\mathcal{H}_2 = \emptyset$, $\mathcal{M}_2 = \{p^2\langle 2\rangle\}$, $\mathcal{N}_2 = \emptyset$.

2) Move the first element of $\mathcal{M}_2$, i.e., $p^2\langle 2\rangle$, to $\mathcal{N}_2$. Then, $\mathcal{M}_2 = \emptyset$ and $\mathcal{N}_2 = \{p^2\langle 2\rangle\}$.

There are three appearances of $p^2\langle 2\rangle$ in $\mathcal{S}$: $app(X_2, p^2\langle 2\rangle)$, $app(X_3, p^2\langle 2\rangle)$, and $app(X_6, p^2\langle 2\rangle)$. Each is Case 3 in Definition 9. Because none of X_2, X_3 and X_6 appears any elsewhere in $\mathcal{S}$, nothing need to do.

3) Because $\mathcal{M}_2 = \emptyset$, by in Step 2 Definition 9, the calculation of the preliminary *SHU* for $p^2\langle 2\rangle$ terminates with $\mathcal{H}_2 = \emptyset$, and $\mathcal{N}_2 = \{p^2\langle 2\rangle\}$. Because $\mathcal{H}_2$ is empty, from Step 2 in Definition 9, let a be the selected constant, $\mathcal{H}_2$ is set to $\{a\}$.

For the preliminary *SHU* $\mathcal{H}_3$ for $f^2\langle 2\rangle$ is $\{b\}$, and the corresponding set of the same domain arguments $\mathcal{N}_3$ is $\{f^2\langle 2\rangle\}$. Because the calculation is trivial, the detail is omitted.

Definition 10 (Algorithm for Deriving all Preliminary *SHU*s). *Let $\mathcal{S}$ be a clause set. All preliminary SHUs for the arguments of predicate symbols and function symbols in $\mathcal{S}$ can be established as follows:*

Step 1. *Let $\mathcal{T}$ be the set of all arguments of predicate symbols and function symbols in $\mathcal{S}$, $j = 0$.*

Step 2. *If $\mathcal{T}$ is empty, terminate; $\mathcal{H}_1, \ldots, \mathcal{H}_j$ are the derived preliminary SHUs. All arguments in $\mathcal{N}_k$ $(1 \leqslant k \leqslant j)$ have the same SHUs $\mathcal{H}_k$. Otherwise, if $\mathcal{T}$ is not empty, continue.*

Step 3. *Let $\alpha^n\langle i\rangle$ be the first element of $\mathcal{T}$, $j = j + 1$. According to Definition* 9, *derive the preliminary SHU $\mathcal{H}_j$ for the argument $\alpha^n\langle i\rangle$ and the set $\mathcal{N}_j$ of the same SHU arguments of $\alpha^n\langle i\rangle$. Remove all elements of $\mathcal{N}_j$ from $\mathcal{T}$, and go to Step* 2.

Similar to the algorithm given in Definition 9, because the number of arguments of predicate symbols and function symbols in a set $\mathcal{S}$ of clauses is finite, the above algorithm also terminates within finite steps.

Example 5. Consider the set $\mathcal{S}$ of clauses given in Example 4. All preliminary *SHU*s of the arguments in $\mathcal{S}$ can be derived as follows:

1) Initially, $\mathcal{T} = \{p^2\langle 1\rangle, p^2\langle 2\rangle, f^2\langle 1\rangle, f^2\langle 2\rangle\}$, and $j = 0$.

2) $j = 1$. For the first element of $\mathcal{T}$, i.e., $p^2\langle 1\rangle$, using the algorithm given in Definition 9 to calculate the preliminary *SHU* $\mathcal{H}_1$ and the corresponding set of the same *SHU* arguments $\mathcal{N}_1$. From Example 4, we have $\mathcal{H}_1 = \mathcal{V}\langle f\rangle \cup \{a\}$ and $\mathcal{N}_1 = \{p^2\langle 1\rangle, f^2\langle 1\rangle\}$. Remove the elements in $\mathcal{N}_1$ from $\mathcal{T}$, then, $\mathcal{T} = \{p^2\langle 2\rangle, f^2\langle 2\rangle\}$.

3) $j = 2$. For the first element of $\mathcal{T}$, i.e., $p^2\langle 2\rangle$, from Example 4, we have $\mathcal{H}_2 = \{a\}$ and $\mathcal{N}_2 = \{p^2\langle 2\rangle\}$. Remove the elements in $\mathcal{N}_2$ from $\mathcal{T}$, then, $\mathcal{T} = \{f^2\langle 2\rangle\}$.

4) $j = 3$. For the first element of $\mathcal{T}$, i.e., $f^2\langle 2\rangle$, from Example 4, we have $\mathcal{H}_3 = \{b\}$, and $\mathcal{N}_3 = \{f^2\langle 2\rangle\}$. Remove the elements in $\mathcal{N}_3$ from $\mathcal{T}$, then, $\mathcal{T} = \emptyset$.

5) Because $\mathcal{T} = \emptyset$, the calculation terminates. $\mathcal{H}_1, \mathcal{H}_2$ and $\mathcal{H}_3$ are the derived preliminary *SHU*s of $\mathcal{S}$.

All *SHU*s of the arguments of predicate symbols and function symbols in a set of clauses in the form of the Herbrand universe can be generated as follows:

Definition 11 (Algorithm for Deriving *SHU*s in the Form of the Herbrand Universe). *Let $\mathcal{S}$ be a clause set, and $f_1, \ldots, f_m$, all function symbols in $\mathcal{S}$, and $\mathcal{H}_1, \ldots, \mathcal{H}_n$ the preliminary SHUs of arguments of predicate symbols and function symbols in $\mathcal{S}$ derived according to Definition* 10.

For each i such that $1 \leqslant i \leqslant n$, let $\mathcal{C}_i$ be the set of constants appearing in $\mathcal{H}_i$,

$$\mathcal{H}_i^*(0) = \mathcal{C}_i$$

and for each j such that $1 \leqslant j \leqslant m$,

$$\mathcal{V}^*(f_j, 0) = \emptyset.$$

Suppose that for $1 \leqslant j \leqslant m$, f_j is an h_j-place function symbol, and the SHU for the argument $f_j^{h_j}\langle t\rangle$ $(1 \leqslant t \leqslant h_j)$ is $\mathcal{H}_{u_t}$, where $1 \leqslant u_t \leqslant n$. For $k = 0, 1, 2, \ldots$, let

$$\mathcal{V}^*(f_j, k+1) = \{f_j(\alpha_1, \ldots, \alpha_{h_j}) \mid \alpha_1 \in \mathcal{H}_{u_1}^*(k), \ldots, \alpha_{h_j} \in \mathcal{H}_{u_{h_j}}^*(k)\}$$

and

$$\mathcal{H}_i^*(k+1) = \mathcal{H}_i^*(k) \cup \{\mathcal{V}^*(f_j, k+1) \mid \mathcal{V}\langle f_j\rangle \in \mathcal{H}_i\}.$$

Then, $\mathcal{V}^*(f_j, \infty)$ is the set of the possible values *of f_j, and $\mathcal{H}_i^*(\infty)$ is the form of the Herbrand universe of $\mathcal{H}_i$.*

Because each element in $\mathcal{H}_i^*(\infty)$ $(1 \leqslant i \leqslant n)$ generated according to Definition 11 only contains constants and function symbols that occur in the given set $\mathcal{S}$ of clauses, each $\mathcal{H}_i^*(\infty)$ is obviously a subset of the Herbrand universe of $\mathcal{S}$. For convenience, we use SHU^* to denote the form of the Herbrand universe of *SHU*.

Example 6. Let $\mathcal{S}$ be the set of clauses given in Example 4. From Example 4, the preliminary *SHU*s are $\mathcal{H}_1 = \mathcal{V}\langle f\rangle \cup \{a\}$, $\mathcal{H}_2 = \{a\}$, and $\mathcal{H}_3 = \{b\}$. The sets of the constants in $\mathcal{H}_1$, $\mathcal{H}_2$ and $\mathcal{H}_3$ are $\mathcal{C}_1 = \{a\}$, $\mathcal{C}_2 = \{a\}$, and $\mathcal{C}_3 = \{b\}$, respectively. Moreover, f is a 2-place function symbol and the preliminary *SHU* for $f^2\langle 1\rangle$ is $\mathcal{H}_1$ and that for $f^2\langle 2\rangle$ is $\mathcal{H}_3$.

Applying the algorithm given in Definition 10, the *SHU*s in the form of the Herbrand universe are calculated as follows:

$$\begin{aligned}
\mathcal{V}^*(f,0) &= \emptyset,\\
\mathcal{H}_1^*(0) &= \mathcal{C}_1 = \{a\},\\
\mathcal{H}_2^*(0) &= \mathcal{C}_2 = \{a\},\\
\mathcal{H}_3^*(0) &= \mathcal{C}_3 = \{b\},\\
\mathcal{V}^*(f,1) &= \{f(\alpha_1,\alpha_2) | \alpha_1 \in \mathcal{H}_1^*(0), \alpha_2 \in \mathcal{H}_3^*(0)\}\\
&= \{f(a,b)\},\\
\mathcal{H}_1^*(1) &= \mathcal{H}_1^*(0) \cup \mathcal{V}^*(f,1) = \{a, f(a,b)\},\\
\mathcal{H}_2^*(1) &= \mathcal{H}_2^*(0) = \{a\},\\
\mathcal{H}_3^*(1) &= \mathcal{H}_3^*(0) = \{b\},\\
\mathcal{V}^*(f,2) &= \{f(\alpha_1,\alpha_2) | \alpha_1 \in \mathcal{H}_1^*(1), \alpha_2 \in \mathcal{H}_3^*(1)\}\\
&= \{f(a,b), f(f(a,b),b)\},\\
\mathcal{H}_1^*(2) &= \mathcal{H}_1^*(1) \cup \mathcal{V}^*(f,2) = \{a, f(a,b), f(f(a,b),b)\},\\
\mathcal{H}_2^*(2) &= \mathcal{H}_2^*(1) = \{a\},\\
\mathcal{H}_3^*(2) &= \mathcal{H}_3^*(1) = \{b\},\\
&\vdots\\
\mathcal{V}^*(f,\infty) &= \{f(a,b), f(f(a,b),b), f(f(f(a,b),b),b), \ldots\},\\
\mathcal{H}_1^*(\infty) &= \{a, f(a,b), f(f(a,b),b),\\
&\quad f(f(f(a,b),b),b), \ldots\},\\
\mathcal{H}_2^*(\infty) &= \{a\},\\
\mathcal{H}_3^*(\infty) &= \{b\}.
\end{aligned}$$

As a comparison, the Herbrand universe $\mathcal{H}$ of $\mathcal{S}$ is $\{a, b, f(a,a), f(a,b), f(b,a), f(b,b), f(f(a,a),a), f(f(a,a),b), f(f(a,b),a), f(f(a,b),b), f(f(b,a),a), f(f(b,a),b), \ldots\}$. Obviously, all $\mathcal{H}_i^*(\infty)$ are smaller than $\mathcal{H}$.

Example 7. Let $\mathcal{S}$ be the clause set given in Example 2. By the algorithms given in Definition 9 and Definition 10,

$$\begin{aligned}
\mathcal{T} &= \{p_1^1\langle 1\rangle, p_2^1\langle 1\rangle, f^1\langle 1\rangle\},\\
\mathcal{H}_1 &= \{c\} \cup \mathcal{V}\langle f\rangle,\\
\mathcal{N}_1 &= \{p_1^1\langle 1\rangle, p_2^1\langle 1\rangle\},\\
\mathcal{H}_2 &= \{c\},\\
\mathcal{N}_2 &= \{f^1\langle 1\rangle\}.
\end{aligned}$$

By the algorithm given in Definition 11,

$$\begin{aligned}
\mathcal{V}^*(f,0) &= \emptyset,\\
\mathcal{H}_1^*(0) &= \{c\},\\
\mathcal{H}_2^*(0) &= \{c\},\\
\mathcal{V}^*(f,1) &= \{f(\alpha) \mid \alpha \in \mathcal{H}_2^*(0)\} = \{f(c)\},\\
\mathcal{H}_1^*(1) &= \mathcal{H}_1^*(0) \cup \mathcal{V}^*(f,1) = \{c, f(c)\},\\
\mathcal{H}_2^*(1) &= \{c\},\\
\mathcal{V}^*(f,2) &= \{f(\alpha) \mid \alpha \in \mathcal{H}_2^*(1)\} = \{f(c)\},\\
\mathcal{H}_1^*(2) &= \mathcal{H}_1^*(1) \cup \mathcal{V}^*(f,2) = \{c, f(c)\},\\
\mathcal{H}_2^*(2) &= \{c\},\\
&\vdots\\
\mathcal{V}^*(f,\infty) &= \{f(c)\},\\
\mathcal{H}_1^*(\infty) &= \{c, f(c)\},\\
\mathcal{H}_2^*(\infty) &= \{c\}.
\end{aligned}$$

$p_1^1\langle 1\rangle$ and $p_2^1\langle 1\rangle$ are the same *SHU* arguments, their SHU^* is $\mathcal{H}_1^*(\infty)$. The SHU^* for $f^1\langle 1\rangle$ is $\mathcal{H}_2^*(\infty)$. Both are finite, even though there is a function in $\mathcal{S}$.

4 Correctness

In this section, we prove that a clause set $\mathcal{S}$ is unsatisfiable if and only if there is a finite unsatisfiable set of ground instances of clauses of $\mathcal{S}$ derived by only instantiating the variables in $\mathcal{S}$ over their corresponding subuniverses of the Herbrand universe of $\mathcal{S}$, respectively.

Definition 12 (SHU^* Ground Instance). *Let $\mathcal{S}$ be a set of clauses. An SHU^* ground instance of a clause C of $\mathcal{S}$ is a clause obtained by replacing each variable in C by a member of the SHU^* for the arguments corresponding to the variable.*

Because every SHU^* of a set $\mathcal{S}$ of clauses is a subset of the Herbrand universe of $\mathcal{S}$, an SHU^* ground instance of a clause C is certainly a ground instance of the clause, but the converse is not always true.

Definition 13 (Depth of a Ground Term). *Let τ be a ground term. The depth of τ, denoted by $dep\langle\tau\rangle$, is defined as follows*:

1) $dep\langle\tau\rangle = 1$ *if τ is a constant.*

2) $dep\langle f(\beta_1, \ldots, \beta_n)\rangle = h + 1$, *where f is an n-place function symbol and h is the maximum value among $dep\langle\beta_1\rangle, \ldots, dep\langle\beta_n\rangle$.*

For example, $dep\langle f(a,b)\rangle = 2$, $dep\langle f(a, g(b,c))\rangle = 3$ and $dep\langle f(a, g(b, h(c)))\rangle = 4$, respectively.

Let $\mathcal{S}$ be an unsatisfiable set of clauses. Then, the empty clause $\square$ can be derived from $\mathcal{S}$ by resolution. We can obtain an unsatisfiable set of ground instances of clauses of $\mathcal{S}$ by recording the clauses used in resolution as follows:

1) When deriving a factor $C\sigma$ of a clause C, instead of deleting all repeated literals from $C\sigma$, we enclose each of them in a box.

2) When deriving a resolvent, instead of deleting the two literals resolved up, we enclose each of them in a box.

Literals enclosed in boxes will not be used in further resolution. A clause with all literals enclosed in boxes is the empty clause. When such a clause, called *extended empty clause*, is derived, for each variable X remains in the clause (if any), we substitute it with a constant in the SHU^* for the arguments corresponding to X. Let E be the resulted clause, then, the set of the clauses occurring in E, denoted as $\mathcal{S}_E$, is an unsatisfiable set of the ground instances of clauses of $\mathcal{S}$.

Example 8. Let $\mathcal{S}$ be the unsatisfiable set of clauses given in Example 4:

$$\neg p(f(a, X_1), X_2), \quad (1)$$
$$p(a, X_3) \vee p(X_4, X_3), \quad (2)$$
$$\neg p(X_5, X_6) \vee p(f(X_5, b), X_6). \quad (3)$$

The empty clause can be derived by resolution as follows:

$$\underbrace{\boxed{p(a, X_3)} \vee p(a, X_3)}_{(2)} \quad (4)$$

where (4) is a factor of (2).

$$\underbrace{\boxed{p(a, X_3)} \vee \boxed{p(a, X_3)} \vee}_{(2)}$$
$$\underbrace{\boxed{\neg p(a, X_3)} \vee p(f(a, b), X_3)}_{(3)} \quad (5)$$

where (5) is the resolvent of (4) and (3).

$$\underbrace{\boxed{p(a, X_3)} \vee \boxed{p(a, X_3)} \vee}_{(2)}$$
$$\underbrace{\boxed{\neg p(a, X_3)} \vee \boxed{p(f(a, b), X_3)} \vee}_{(3)}$$
$$\underbrace{\boxed{\neg p(f(a, b), X_3)}}_{(1)} \quad (6)$$

where (6) is the resolvent of (5) and (1).

Because all literals in clause (6) are enclosed in a box, clause (6) is an extended empty clause. Moreover, because there are $app(X_3, p^2\langle 2\rangle)$ in (6) and $a \in \mathcal{H}(p^2\langle 2\rangle)$ (see Example 6), substituting variable X_3 in clause (6) with constant a, we have the following clause E:

$$\underbrace{\boxed{p(a, a)} \vee \boxed{p(a, a)} \vee}_{(2)}$$
$$\underbrace{\boxed{\neg p(a, a)} \vee \boxed{p(f(a, b), a)} \vee}_{(3)}$$
$$\underbrace{\boxed{\neg p(f(a, b), a)}}_{(1)}. \quad (7)$$

The unsatisfiable set $\mathcal{S}_E$ of ground instances of clauses of $\mathcal{S}$ derived from clause (7) is:

$\neg p(f(a, b), a)$
$\cdots$ a ground instance of (1),
$p(a, a) \vee p(a, a)$
$\cdots$ a ground instance of (2),
$\neg p(a, a) \vee p(f(a, b), a)$
$\cdots$ a ground instance of (3).

Lemma 2. *Let $\mathcal{S}$ be a set of clauses, T a factor or a resolvent derived in resolution on $\mathcal{S}$, and X a variable in T. Then, all arguments corresponding to X in T are the same domain arguments.*

Proof. We prove Lemma 2 by induction on the following statement:

$I(n)$: Suppose that T_n is the clause derived in the n-th step in resolution. Then, all arguments corresponding to a variable X in T_n are the same domain arguments.

Base case: show $I(0)$. T_0 is a clause in $\mathcal{S}$. According to Definition 9, all arguments corresponding to a variable X in T_0 are the same domain arguments.

Induction step: suppose that $I(0), \ldots, I(n)$, to show $I(n+1)$.

T_{n+1} is a factor of a clause C (a resolvent of two clauses C_1 and C_2), where C (each of C_1 and C_2) is a clause derived in $I(i)$ $(0 \leqslant i \leqslant n)$.

Because X is a variable in T_{n+1}, X is certainly a variable in C (C', where C' is one of C_1 and C_2). For the appearances $app(X, \alpha_1^{n_1}\langle i_1\rangle), \ldots, app(X, \alpha_r^{n_r}\langle i_r\rangle)$ in T_{n+1} such that there are also $app(X, \alpha_1^{n_1}\langle i_1\rangle), \ldots, app(X, \alpha_r^{n_r}\langle i_r\rangle)$ in C (C'), by the induction assumption, $\alpha_1^{n_1}\langle i_1\rangle, \ldots, \alpha_r^{n_r}\langle i_r\rangle$ are the same domain arguments.

The remain appearances of X in T_{n+1} are generated by substituting other variables, say $Y_1, \ldots, Y_t$, in C (C_1 or C_2) with X. For each Y_k $(1 \leqslant k \leqslant t)$, there is certainly a sequence of $app(X, \beta_1^{m_1}\langle j_1\rangle)$, $app(Y_{s_1}, \beta_1^{m_1}\langle j_1\rangle)$, $app(Y_{s_1}, \beta_2^{m_2}\langle j_2\rangle)$, $app(Y_{s_2}, \beta_2^{m_2}\langle j_2\rangle)$, $app(Y_{s_2}, \beta_3^{m_3}\langle j_3\rangle), \ldots$, $app(Y_{s_u}, \beta_u^{m_u}\langle j_u\rangle)$, $app(Y_k, \beta_u^{m_u}\langle j_u\rangle)$, where $1 \leqslant s_v \leqslant t$, $1 \leqslant v \leqslant u$, and $\beta_l^{m_l}\langle j_l\rangle$ $(1 \leqslant j_l \leqslant m_u,\ 1 \leqslant l \leqslant u)$ is an argument in C (C_1 or/and C_2). By Definition 9, all arguments corresponding to variable X and those corresponding to variable $Y_{s_1}, \ldots, Y_{s_u}, Y_k$ in C (C_1 or/and C_2) respectively are the same domain argument. Then, all arguments cor-

responding to variable X in T_{n+1} are the same domain arguments.

Therefore, $I(n+1)$ is true. □

For example, let C be a clause $p(X,Y,Y,Z) \vee q(Z) \vee p(U,U,V,V)$. Then, $p(X,X,X,X) \vee q(X) \vee \boxed{p(X,X,X,X)}$ is a factor of C. For variable Z, which is substituted to X in factorization, there is a sequence $app(X,p^4\langle 1\rangle)$, $app(U,p^4\langle 1\rangle)$, $app(U,p^4\langle 2\rangle)$, $app(Y,p^4\langle 2\rangle)$, $app(Y,p^4\langle 3\rangle)$, $app(V,p^4\langle 3\rangle)$, $app(V,p^4\langle 4\rangle)$, and $app(Z,p^4\langle 4\rangle)$. By Definition 9, all arguments corresponding to variables X, Y, Z, U, and V in C, i.e., $\{p^4\langle 1\rangle\}$, $\{p^4\langle 2\rangle,\ p^4\langle 3\rangle\}$, $\{p^4\langle 4\rangle,\ q^1\langle 1\rangle\}$, $\{p^4\langle 1\rangle,\ p^4\langle 2\rangle\}$, and $\{p^4\langle 3\rangle,\ p^4\langle 4\rangle\}$, respectively, are the same domain arguments. The set of such arguments, i.e., $\{p^4\langle 1\rangle,\ p^4\langle 2\rangle,\ p^4\langle 3\rangle,\ p^4\langle 4\rangle,\ q^1\langle 1\rangle\}$, is just the set of the arguments corresponding to variable X in the factor.

Lemma 3. *During our proposed resolution, whenever there is $app(c,\alpha^n\langle i\rangle)$ ($app(f,\alpha^n\langle i\rangle)$), there is $c \in \mathcal{H}(\alpha^n\langle i\rangle)$ ($\mathcal{V}\langle f\rangle \in \mathcal{H}(\alpha^n\langle i\rangle)$), respectively.*

Proof. For each constant c and each argument $\alpha^n\langle i\rangle$ such that there is $app(c,\alpha^n\langle i\rangle)$ in $\mathcal{S}$, by Definition 9, $c \in \mathcal{H}(\alpha^n\langle i\rangle)$.

For each functional term $f(\tau_1,\ldots,\tau_n)$ and each argument $\alpha^n\langle i\rangle$ such that there is $app(f(\tau_1,\ldots,\tau_n),\alpha^n\langle i\rangle)$ in $\mathcal{S}$, by Definition 9, $\mathcal{V}\langle f\rangle \in \mathcal{H}(\alpha^n\langle i\rangle)$.

If a variable X is substituted to a functional term $f(\tau_1,\ldots,\tau_n)$ in resolution, then there are two cases. One is that there are $app(X,\alpha^n\langle i\rangle)$ and $app(f(\tau_1,\ldots,\tau_n),\alpha^n\langle i\rangle)$ in $\mathcal{S}$, by Definition 9, $\mathcal{V}\langle f\rangle \in \mathcal{H}(\alpha^n\langle i\rangle)$. The other is that there are $app(Y,\beta^m\langle j\rangle)$ and $app(f(\tau_1,\ldots,\tau_n),\beta^m\langle j\rangle)$ in $\mathcal{S}$ such that Y is substituted to X in resolution, by Definition 9, $\mathcal{V}\langle f\rangle \in \mathcal{H}(\beta^m\langle j\rangle)$, and by Lemma 2, all arguments corresponding to variables X and Y are the same domain arguments, i.e., for each $app(X,\alpha^n\langle i\rangle)$ in the resolvent, $\mathcal{H}(\alpha^n\langle i\rangle) = \mathcal{H}(\beta^m\langle j\rangle)$. Therefore, we have $\mathcal{V}\langle f\rangle \in \mathcal{H}(\alpha^n\langle i\rangle)$.

Now, consider the case where a variable X is substituted to a constant c. If it occurs in our proposed resolution, same as above, there are also two cases. One is that there is $app(X,\alpha^n\langle i\rangle)$ and $app(c,\alpha^n\langle i\rangle)$ in $\mathcal{S}$, by Definition 9, $c \in \mathcal{H}(\alpha^n\langle i\rangle)$. The other is that there are $app(Y,\beta^m\langle j\rangle)$ and $app(c,\beta^m\langle j\rangle)$ in $\mathcal{S}$ such that Y is substituted to X in resolution, by Definition 9, $c \in \mathcal{H}(\beta^m\langle j\rangle)$, and by Lemma 2, all arguments corresponding to variables X and Y are the same domain arguments, i.e., for each $app(X,\alpha^n\langle i\rangle)$ in the resolvent, $\mathcal{H}(\alpha^n\langle i\rangle) = \mathcal{H}(\beta^m\langle j\rangle)$. Therefore, we have $c \in \mathcal{H}(\alpha^n\langle i\rangle)$. On the other hand, when a variable X in the extended empty clause derived by our proposed resolution is substituted to a constant c, by our method proposed above, for some argument $\alpha^n\langle i\rangle$, there are $app(X,\alpha^n\langle i\rangle)$ in a clause of $\mathcal{S}$ and $c \in \mathcal{H}(\alpha^n\langle i\rangle)$. □

Lemma 4. *Let $\mathcal{S}$ be an unsatisfiable set of clauses, and $\mathcal{S}_E$ the unsatisfiable set of ground instances of clauses of $\mathcal{S}$ derived by our proposed resolution. Then, each clause of $\mathcal{S}_E$ is an SHU^* ground clause of $\mathcal{S}$.*

Proof. Because there is no variable in $\mathcal{S}_E$, all terms in $\mathcal{S}_E$ are ground. We show that for each ground term τ, if there is $app(\tau,\alpha^n\langle i\rangle)$ in $\mathcal{S}_E$, then $\tau \in \mathcal{H}(\alpha^n\langle i\rangle)$. We do it by induction on $dep\langle\tau\rangle$.

If $dep\langle\tau\rangle = 1$, τ is a constant. By Lemma 3, $\tau \in \mathcal{H}(\alpha^n\langle i\rangle)$.

Assume the above statement holds when $dep\langle\tau\rangle = i$, $1 \leqslant i \leqslant t$. We show that it holds when $dep\langle\tau\rangle = t+1$.

Without loss of generality, let $\tau = f(\rho_1,\ldots,\rho_m)$, where $dep\langle\rho_j\rangle \leqslant t$, $1 \leqslant j \leqslant m$. By the above discussion, $\mathcal{V}\langle f\rangle \subset \mathcal{H}(\alpha^n\langle i\rangle)$. By the induction hypothesis, $\rho_j \in \mathcal{H}(f^m\langle j\rangle)$, and by Definition 11, $f(\rho_1,\ldots,\rho_m) \in \mathcal{H}(\alpha^n\langle i\rangle)$. That is, $\tau \in \mathcal{H}(\alpha^n\langle i\rangle)$ for $dep\langle\tau\rangle = t+1$.

Then, by above discussion and Definition 12, every clause of $\mathcal{S}_E$ is an SHU^* ground instance of some clause of $\mathcal{S}$.[④] □

Theorem 5 (Correctness). *A set $\mathcal{S}$ of clauses is unsatisfiable if and only if there is a finite unsatisfiable set $\mathcal{S}^*$ of the SHU^* ground instances of clauses of $\mathcal{S}$.*

Proof. ($\Rightarrow$) Suppose that $\mathcal{S}$ is unsatisfiable. By our proposed resolution, we can derive an unsatisfiable set $\mathcal{S}_E$ of ground instances of clauses of $\mathcal{S}$. By Lemma 4, each clause of $\mathcal{S}_E$ is an SHU^* ground instance of some clause of $\mathcal{S}$. Let $\mathcal{S}^* = \mathcal{S}_E$, then $\mathcal{S}^*$ is an unsatisfiable set of SHU^* ground instances of clauses of $\mathcal{S}$.

($\Leftarrow$) Suppose that there is a finite unsatisfiable set $\mathcal{S}^*$ of SHU^* ground instances of clauses of $\mathcal{S}$. Because each SHU^* ground instance clause of $\mathcal{S}$ is a ground instance clause of $\mathcal{S}$, by Herbrand's theorem, $\mathcal{S}$ is unsatisfiable. □

5 Application to Model Generation Theorem Proving

By Theorem 5, to check the unsatisfiability of a clause set, instead of the set of ground instances over the Herbrand universe of the clause set, we need only consider the sets of SHU^* ground instances of clauses in the clause set. Therefore, when we transform a non-range-restricted clause set to its corresponding range-restricted one, we can limit each non-range-restricted variable to elements of the SHU^* of the arguments corresponding to the variable.

Definition 14 (SHU Range-Restriction Transformation Algorithm). *Let $\mathcal{S}$ be a non-range-restricted clause set. Suppose that there are t different SHUs, says, $\mathcal{H}_1,\ldots,\mathcal{H}_t$, for the arguments in $\mathcal{S}$ derived according to Definition 10. For each $\mathcal{H}_l$ ($1 \leqslant l \leqslant t$), we introduce an auxiliary predicate symbol dom_l. $\mathcal{S}$ is transformed to a range-restricted clause set $\mathcal{S}^*$, called the SHU range-restricted form of $\mathcal{S}$, in the following way:*

1. *Every range-restricted clause $\mathcal{A} \to \mathcal{C}$ is transformed into itself.*

2. *For every non-range-restricted clause $\mathcal{A} \to \mathcal{C}$, let $X_1,\ldots,X_n$ be non-range-restricted variables in the*

[④]For example, each ground instance of clauses derived in Example 8 is an SHU^* ground instance.

clause, and $\mathcal{H}_{s_1}, \ldots, \mathcal{H}_{s_n}$ $(1 \leqslant s_k \leqslant t,\ 1 \leqslant k \leqslant n)$, *the SHUs for the arguments corresponding to* $X_1, \ldots, X_n$. *Transform the clause to* $\mathcal{A}$, $dom_{s_1}(X_1), \ldots, dom_{s_n}(X_n) \to \mathcal{C}$. *However,* $\mathcal{A}$ *is omitted if it is* $\top$.

3. *For every* $\mathcal{H}_j$ *such that* $1 \leqslant j \leqslant t$,

(1) *for each constant* $c \in \mathcal{H}_j$, *add a clause* $\top \to dom_j(c)$;

(2) *for each* $\mathcal{V}\langle f \rangle \in \mathcal{H}_j$, *suppose that* f *is an* m*-place function, and* $\mathcal{H}_{h_1}, \ldots, \mathcal{H}_{h_m}$ *are the SHUs for the arguments* $f^m\langle 1 \rangle, \ldots, f^m\langle m \rangle$, *add a clause* $dom_{h_1}(Y_1), \ldots, dom_{h_m}(Y_m) \to dom_{h_j}(f(Y_1, \ldots, Y_m))$.

Obviously, by the transformation procedures given in Definition 14, a non-range-restricted variable is only instantiated over the SHU^* corresponding to the variable.

Because each *SHU* for the arguments corresponding to a variable in a non-range-restricted clause of a clause set $\mathcal{S}$ is a subset of the Herbrand universe of $\mathcal{S}$, by limiting non-range-restricted variables to elements of the *SHU*s of the arguments corresponding to those variables, the number of ground instances might be reduced considerably. For model generation theorem proving, this means that the number of ground clauses used forward chaining might be reduced, thus, reasoning could be made efficiently.

Example 9. Let $\mathcal{S}$ be the clause set given in Example 2. By Example 7, the *SHU* for the argument $p_1^1\langle 1 \rangle$ and $p_2^1\langle 1 \rangle$ is $\{c\} \cup \mathcal{V}\langle f \rangle$ and the *SHU* for the argument $f^1\langle 1 \rangle$ is $\{c\}$. By the algorithm given in Definition 14, $\mathcal{S}$ can be transformed into the following *SHU* range-restricted clause set $\mathcal{S}^*$:

$$
\begin{aligned}
&p_1(c) \to \bot,\\
&p_2(f(c)) \to \bot,\\
&dom_1(X) \to p_1(X); p_2(X),\\
&\top \to dom_1(c),\\
&\top \to dom_2(c),\\
&dom_2(Y) \to dom_1(f(Y)).
\end{aligned}
$$

SATCHMO can immediately show $\mathcal{S}^*$ satisfiable. The proof tree on $\mathcal{S}^*$ constructed by SATCHMO is shown in Fig.2.

Example 10. Consider the problem PUZ017-1 given in TPTP (the Thousand Problems for Theorem Proving) library[19], which is described as follows.

There are 5 houses, 5 people, 5 colors, 5 drinks, 5 games, and 4 pets. Each house has a person, a color, a drink, and a game, and all but one of the houses has a pet. The problem is to match each house with as many properties as possible. House 1 is at the left end and house 5 is at the right end. The Englishman lives in the Red house. The White house is at the left of the Green house. The Italian has a Guppy. Lemonade is drunk in the Green house. The Swede lives in the house where Coffee is drunk. The Toad lives in the house where Backgammon is played. Racquetball is played in the Yellow house. Milk is drunk in the third house. The Russian lives in the first house. The Camel lives next to the house where Quoits is played. The Rat lives next to the house where Racquetball is played. Solitaire is played in the house where vodka is drunk. The American lives in the house where Charades is played. The Russian lives next to the Blue house.

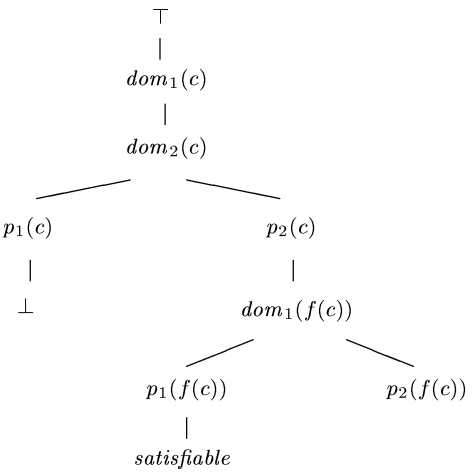

Fig.2. Proof tree constructed by SATCHMO in Example 9.

For convenience, we use $h1$, $h2$, $h3$, $h4$, $h5$ to denote house 1, house 2, house 3, house 4, house 5, and hd for *hasdrink*, respectively. The Herbrand universe $\mathcal{H}$ of the problem consists of all constants in the problem:

$\mathcal{H} = \{h1, h2, h3, h4, h5,$ no_pet, rat, camel, toad, guppy, charades, solitaire, quoits, racquetball, backgammon, unknown_drink, vodka, milk, coffee, lemonade, blue, yellow, green, white, red, american, russian, swede, italian, englishman$\}$

There is, for example, the following non-range-restricted clause in the problem:

$$\top \to hd(h1, A); hd(h2, A); hd(h3, A); hd(h4, A); hd(hd5, A). \quad (1)$$

By the general range-restricted transformation algorithm given in Definition 6, for each element c in $\mathcal{H}$, a clause $\top \to dom(c)$ will be added, and the clause (1) is transformed to

$$dom(A) \to hd(h1, A); hd(h2, A); hd(h3, A); hd(h4, A); hd(h5, A).$$

The variable A in the above clause can take any element of the Herbrand universe $\mathcal{H}$. For this reason, for example, the following ground instance of the above clause becomes violated and is used for forward chaining:

$$dom(rat) \to hd(h1, rat); hd(h2, rat); hd(h3, rat); hd(h4, rat); hd(h5, rat).$$

Obviously, such ground instance is unreasonable and therefore is redundant for a proof, because *rat* is obviously not a *drink*. In the same way, there are such unreasonable and redundant ground instances of clause

(1) for each *house*, each *person*, each *color*, each *pet* and each *game*. For this sake, no model generation based theorem prover can complete the proof within an hour.

By our algorithm given in Definition 11, the following six SHU^*s can be derived:

$$\begin{aligned}
\mathcal{H}_1 &= \{h1, h2, h3, h4, h5\},\\
\mathcal{H}_2 &= \{no_pet, rat,\ camel,\ toad,\ guppy\},\\
\mathcal{H}_3 &= \{charades,\ solitaire,\ quoits,\\
&\qquad racquetball,\ backgammon\},\\
\mathcal{H}_4 &= \{unknown_drink,\ vodka,\ milk,\\
&\qquad coffee,\ lemonade\},\\
\mathcal{H}_5 &= \{\text{blue, yellow, green, white, red}\},\\
\mathcal{H}_6 &= \{\text{american, russian, swede, italian,}\\
&\qquad englishman\}.
\end{aligned}$$

It is worth noticing that, the above SHU^*s are naturally divided according to *house*, *pet*, *game*, *drink*, *color* and *people*, respectively.

According to the *SHU* range-restriction transformation given in Definition 14, clause (1) is transformed into the following one:

$$\begin{aligned}
dom_4(A) \rightarrow\ & hd(h1, A); hd(h2, A);\\
& hd(h3, A); hd(h4, A); hd(h5, A)
\end{aligned}$$

where dom_4 is the auxiliary predicate symbol corresponding to $\mathcal{H}_4$. In this way, the variable A in the above clause can only take elements in $\mathcal{H}_4$, i.e., *unknown_drink*, *vodka*, *milk*, *coffee* and *lemonade*, all are really existing *drink*s. Thus, no unreasonable ground instances of the clause would be generated. The model generation based theorem provers SATCHMO and R-SATCHMO[11] gave the proof of the transformed *SHU* range-restricted one of PUZ017-1 within 3 seconds.

6 Experimental Results

There are 3317 non-Horn problems in the TPTP library version 2.5.0, a benchmark problem library for automated theorem proving, where there are 5181 problems in all[19]. Among them, 2881 problems are non-range-restricted, and the number of the problems from which 2 or more *SHU*s can be derived by our approach is 451. Among them, 316 problems are *really non-propositional*[5], and other 135 problems are really propositional. The largest number of the derived *SHU*s is, somewhat surprisingly, 271 (Problem SYN837-1).

We run all the 451 applicable problems by SATCHMO and R-SATCHMO without and with (the cases with symbol $*$), our improvement on an Intel PentiumIII/980MHz workstation with a time limit of 300 seconds. Both systems were implemented by SWI-Prolog. The running times included the range-restricted transformation, compilation and reasoning.

Table 1 shows the number of the problems solved by SATCHMO and R-SATCHMO. We found that if a problem could be solved by SATCHMO and/or R-SATCHMO, then it could also be solved by SATCHMO* and/or R-SATCHMO* within almost the same time, but the converse was not always true. This shows that the *SHU* range-restriction transformation almost takes no additional cost than the general range-restriction transformation.

Table 2 shows some selected experimental results, where the running times are given in seconds and T.o. means time out, i.e., > 300 seconds, U means unsatisfiable, and S for satisfiable. All problems are non-range-restricted. We can find that our approach is powerful for model generation theorem proving on non-range-restricted problems.

Among 143 problems resolved by R-SATCHMO*, 81 problems are satisfiable and the other 62 problems are unsatisfiable. The largest number of derived *SHU*s is 22 (SYN890-1). 24 problems have the rating beyond zero (such problems are said to be difficult problems), the highest rating is 0.83 (TOP002-1 and TOP003-1, both are satisfiable) and the lowest rating is 0.33 (NLP061-1, SYN851-1, etc.). Among them, 15 problems are satisfiable and the other 9 problems are unsatisfiable. It turns out that our method is more suitable for satisfiable problems.

As a comparison of R-SATCHMO* with other automated reasoning systems that entered the most recent automated theorem prover system competition held in IJCAI-3 (CASC-J3)[20], where the used computers were AMD Athlon XP2200+/1797MHz, we run R-SATCHMO* on the problems used in CASC-J3. The results were not as good as expected for R-SATCHMO*. The reasons are: 1) as indicated in [21], most of the problems in the TPTP library are designed with refutation-oriented theorem provers in mind, therefore favoring those systems; 2) R-SATCHMO* lacks built-in equality treatment; 3) R-SATCHMO* was implemented in SWI-Prolog, while all of systems that entered CASC-J3 were implemented in C or C++ (it was reported in [22] that a system implemented in C or C++ would be at least 300 times faster than that implemented in SWI-Prolog); 4) the computers used in CASC-J3 were almost two times faster than ours.

For the EPS (Effectively Propositional Satisfiable clause sets) division, R-SATCHMO* seems quite good for problems from which two or more *SHU*s can be derived. The results are shown in Table 3, where P_0, P_1, P_2, P_3, P_3, P_4, P_5, P_6, P_7, P_8, P_9 and R^* are denoted provers Darwin1.3, DCTP10.21p, DarwinFM1.3, Paradox2.0a, Geo2006i, iProver0.1, E0.99, Vampire8.1, Equinox1.0a and R-SATCHMO*, No., for the number of divided *SHU*s, G.u., for give-up, and T.o., for time-out, respectively. All data except for those of R-SATCHMO* are taken from [20].

[5] Really non-propositional means with an infinite Herbrand universe[20].

Table 1. Experimental Results (I)

Number of Problems	SATCHMO	SATCHMO*	R-SATCHMO	R-SATCHMO*
451	77	132	85	143

Table 2. Experimental Results (II)

Problem	Number of Derived *SHU*s	U/S	SATCHMO	SATCHMO*	R-SATCHMO	R-SATCHMO*
COM002-2	12	U	T.o.	0.6	0.3	0.2
KRS016-1	2	S	T.o.	0.2	T.o.	0.2
MSC004-1	3	U	T.o.	T.o.	T.o.	0.2
NLP048-1	4	S	T.o.	1.0	T.o.	1.0
NLP060-1	7	S	T.o.	160	T.o.	160
NLP067-1	5	S	T.o.	0.3	T.o.	0.2
NLP079-1	5	U	T.o.	0.6	T.o.	1.0
NLP131-1	6	S	T.o.	0.3	T.o.	1.0
NLP161-1	7	S	T.o.	1.0	T.o.	1.0
NLP221-1	5	S	T.o.	0.3	T.o.	0.2
NLP230-1	5	S	T.o.	2.2	T.o.	0.4
PLA029-1	4	S	T.o.	0.3	T.o.	0.2
PUZ017-1	6	U	T.o.	0.6	T.o.	0.2
PUZ019-1	2	U	T.o.	1.0	1.0	0.3
PUZ044-1	2	S	T.o.	0.3	T.o.	0.3
SET777-1	2	S	T.o.	0.2	T.o.	0.2
SYN304-1	5	S	T.o.	0.2	T.o.	1.0
SYN320-1	3	S	T.o.	0.3	T.o.	1.0
SYN345-1	2	U	T.o.	0.2	T.o.	0.3
SYN419-1	2	S	T.o.	242	T.o.	252
SYN423-1	2	S	T.o.	163	T.o.	167
SYN425-1	2	S	T.o.	126	T.o.	127
SYN513-1	2	S	T.o.	131	T.o.	129
SYN545-1	2	S	T.o.	227	T.o.	23
SYN870-1	21	S	T.o.	T.o.	T.o.	146
SYN871-1	22	U	T.o.	135	T.o.	86
SYN875-1	22	U	T.o.	70	T.o.	25
SYN879-1	21	U	T.o.	T.o.	T.o.	114
SYN891-1	22	U	T.o.	90	T.o.	87
SYN892-1	20	U	T.o.	206	T.o.	133
TOP003-1	3	S	T.o.	8.26	T.o.	0.27
TOP006-1	3	S	35.75	0.27	1.83	0.15

Table 3. Experimental Results on the Problems in the EPS Division in CASC-J3

Problems	P_1	P_2	P_3	P_4	P_5	P_6	P_7	P_8	P_9	*No.*	R*
PUZ018-2	0.0	0.1	0.1	0.0	3.4	0.0	T.o.	T.o.	G.u.	2	0.1
SYN307-1	0.0	0.0	0.0	0.0	0.0	0.0	T.o.	T.o.	G.u.	2	0.1
SYN317-1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	G.u.	2	0.1
SYN419-1	0.1	1.1	0.1	0.6	59.1	T.o.	1.1	T.o.	G.u.	2	252.8
SYN423-1	0.1	1.9	11.2	1.1	T.o.	T.o.	T.o	T.o.	G.u.	2	166.6
SYN521-1	0.0	0.1	0.0	0.1	0.5	0.0	0.0	0.0	G.u.	4	0.1
SYN534-1	0.0	0.1	0.0	0.1	0.1	0.0	0.0	0.0	G.u.	2	0.1
SYN539-1	0.0	0.2	0.0	0.2	0.5	0.0	1.2	1.5	G.u.	2	0.8
SYN541-1	0.0	0.1	0.0	0.2	0.5	0.0	T.o.	0.2	G.u.	2	0.5
SYN823-1	0.5	1.6	0.7	15.0	T.o.	284.4	0.2	0.4	G.u.	128	T.o.
SYN872-1	0.5	0.6	1.2	179.8	T.o.	T.o.	0.3	0.5	G.u.	62	T.o.
SYN888-1	0.8	1.8	1.9	281.7	T.o.	T.o.	0.4	0.6	G.u.	64	T.o.

R-SATCHMO* had an adequate performance among all provers, where SYN419-1 and SYN423-1 were resolved with the help of our approach.

R-SATCHMO* failed on SYN823-1, SYN872-1 and SYN888-1, each of them has a large size. For example, there are 386 clauses, 5849 atoms, 241 predicates and 5029 variables (345 singleton) in SYN823-1. A theorem proving system implemented in Prolog hardly resolves so large sized problems within 300 seconds. Keeping in mind that R-SATCHMO* was implemented in Prolog, while other theorem provers were implemented in C or C++ and the computer used in CASC-J3 was almost two times faster than ours, we would expect that R-SATCHMO* could resolve SYN823-1, SYN872-1 and SYN888-1 if it had been implemented in C or C++.

7 Conclusion

In this paper, we presented an improvement of Herbrand's theorem and proposed the algorithm for deriving sub-Herbrand universes for arguments of predicate symbols and function symbols in a given clause set. We have shown that to check unsatisfiability of a set $\mathcal{S}$ of clauses, instead of considering the set $\mathcal{S}'$ of ground in-

stances of clauses of $\mathcal{S}$ derived by instantiating variables in $\mathcal{S}$ over the Herbrand universe of $\mathcal{S}$, we can consider the set $\mathcal{S}^*$, which is a subset of $\mathcal{S}'$, of ground instances of clauses of $\mathcal{S}$ derived by instantiating variables in $\mathcal{S}$ to elements of *SHU*s corresponding to the variables. In cases where non-range-restricted problems are applied to a model generation theorem prover, this directly leads to reduction the number of ground clauses used for forward chaining, therefore the search space is limited and the efficiency of reasoning might be enhanced considerably.

It is worth mentioning that, the strategy proposed in this paper is a basic one and therefore can be further improved. For example, we can derive a smaller sub-universe of the Herbrand universe for an argument of predicates or functions by differentiating the positive atoms and the negative atoms in a given set of clauses.

We will also try to find other applications on the full first-order logic for our approach.

In order to further investigate the power of our approach, we will improve R-SATCHMO* by incorporating built-in equality treatment, use C or C++ to implement our system, and make a challenge to refutation-oriented theorem provers in future CASCs.

It is said that a problem is really non-propositional if it has an infinite Herbrand universe, i.e., if it contains a function. With our improvement, the definition should be changed to: a problem is really non-propositional if it has at least an infinite sub-Herbrand universe for some argument in the problem.

Acknowledgments We are deeply grateful to the anonymous referees, who provided a lot of valuable comments that improved this paper greatly. We thank the editor for his/her kind cooperation and help. We also thank E. F. Lanzl for improving the English of this paper.

References

[1] Herbrand J. Recherches sur la théorie de la démonstration [Dissertation]. University of Paris, 1930.
[2] Gilmore P C. A proof method for quantification theory: Its justification and realization. *IBM J. Res. Develop.* 1960, pp.28~35.
[3] Manthey R, Bry F. SATCHMO: A theorem prover implemented in Prolog. In *Proc. 9th Int. Conf. Automated Deduction*, Argonne, Illinois, USA, 1988, pp.415~434.
[4] Bry F, Yahya A. Positive unit hyperresolution tableaux and their application to minimal model generation. *J. Automated Reasoning*, 2000, 25(1): 35~82.
[5] Stickel M E. Upside-down meta-interpretation of the model elimination theorem-proving procedure for deduction and abduction. *J. Automated Reasoning*, 1994, 13(2): 189~210.
[6] Geisler T, Panne S, Schutz H. Satchmo: The compiling and functional variants. *J. Automated Reasoning*, 1997, 18(2): 227~236.
[7] Loveland D W, Reed D W, Wilson D S. SATCHMORE: SATCHMO with RElevancy. *J. Automated Reasoning*, 1995, 14(2): 325~351.
[8] He L, Chao Y, Simajiri Y *et al.* $\mathcal{A}$-SATCHMORE: SATCHMORE with availability checking. *New Generation Computing*, 1998, 16(1): 55~74.
[9] He L. I-SATCHMO: An Improvement of SATCHMO. *J. Automated Reasoning*, 2001, 27(3): 313~322.
[10] He L, Chao Y, Nakamura T *et al.* $\mathcal{I}$-SATCHMORE: An improvement of $\mathcal{A}$-SATCHMORE. *J. Comput. Sci. & Technol.*, 2003, 18(2): 181~189.
[11] He L, Chao Y, Itoh H. R-SATCHMO: Refinements on I-SATCHMO. *J. Logic and Computation*, 2004, 14(2): 117~143.
[12] Loveland D W, Yahya A H. SATCHMOREBID: SATCHMO(RE) with BIDirectional relevancy. *New Generation Computing*, 2003, 21(3): 175~206.
[13] Schulz S. A comparison of different techniques for grounding near-propositional CNF formulae. In *Proc. the 15th FLAIRS*, London, 2002, pp.72~76.
[14] Lee S J, Plaisted D A. Eliminating duplication with the hyper-linking strategy. *J. Automated Reasoning*, 1992, 9(1): 25~42.
[15] Yu Q, Almulla M, Newborn M. Heuristics used by HERBY for semantic tree theorem proving. *Ann. Math. Artif. Intell.*, 1998, 23(3/4): 247~266.
[16] Chang C L, Lee K C T. Symbolic Logic and Mechanical Theorem Proving. New York: Academic Press, 1997.
[17] Loveland D W. Automated Theorem Proving: A Logic Basis. Amsterdam: North-Holland, 1978.
[18] Robinson J A. A machine-oriented logic based on the resolution principle. *J. ACM*, 1965, 12(1): 23~41.
[19] Sutcliffe G, Suttner C. The TPTP problem library for automated theorem proving. http://www.cs.miami.edu/~tptp/.
[20] The CADE ATP System Competition held at the *Third International Joint Conference on Automated Reasoning*, Seattle, USA, 2006. http://www.cs.miami.edu/~tptp/CASC/J3/
[21] Schutz H, Geisler T. Efficient model generation through compilation. *Information and Computation*, 2000, 162(2): 138~157.
[22] Morales J, Carro M, Hermenegildo M. Improving the compilation of Prolog to C using type and determinism information: Preliminary results. In *Proc. Colloquium on Implementation of Constraint and Logic Programming Systems* (*ICLP Associated Workshop*), Mumbai, India, December 2003, pp.89~102.

Yu-Yan Chao received the B.E. degree from Northwest Institute of Light Industry, China, in 1984, and the M.S. and the Ph.D. degrees from Nagoya University, Japan, in 1997 and 2000, respectively. From 2000 to 2002, she was a special foreign researcher of Japan Society for the Promotion of Science in Nagoya Institute of Technology. She is an associate professor in Nagoya Sangyo University, Japan and a guest professor in the Shaanxi University of Science and Technology, China. Her research interests include automated reasoning, artificial intelligence, image processing, graphic understanding and CAD.

Li-Feng He received the B.E. degree from Northwest Institute of Light Industry, China, in 1982, the M.S. and the Ph.D. degrees in AI and computer science from Nagoya Institute of Technology, Japan, in 1994 and 1997, respectively. He is an associate professor in Aichi Prefectural University, Japan and a guest professor in the Shaanxi University of Science and Technology, China. From Sept. 2006 to June 2007, he works in the University of Chicago (USA) as a visiting scholar. His research interests include automated reasoning, theorem

proving, multi-agent cooperative computing, image processing and artificial intelligence. He is a member of the Associate of Automated Reasoning (AAR).

Tsuyoshi Nakamura received the Ph.D. degree from Nagoya Institute of Technology, Japan, in 1998. He is an associate professor in Nagoya Institute of Technology. His current research interests include AI, softcomputing, CG and emotional information processing. He is a member of IEEE.

Zheng-Hao Shi received the M.S. degree in computer science from Xi'an University of Technology, China and Ph.D. degree from Xi'an Institute of Microelectronics Technology, China, in 2000 and 2005 respectively. He is currently a post-doctoral fellow of Itoh Laboratory at Nagoya Institute of Technology, Japan, and an associate professor of computer science at Xi'an University of Technology, China. His research interests are in the area of image process ,computer vision, neural networks and AI.

Kenji Suzuki received his B.S. and M.S. degrees in engineering from Meijo University, Japan, in 1991 and 1993, respectively, and his Ph.D. degree in information engineering from Nagoya University in 2001. From 1993 to 1997, he worked in Research and Development Center at Hitachi Medical Corporation as a researcher. From 1997 to 2001, he worked in Department of Applied Information Science and Technology at Aichi Prefectural University, Japan, as a faculty member. In 2001, he joined Department of Radiology at The University of Chicago, as research associate. He was promoted to research associate (Instructor) and then to research associate (assistant professor). Since 2006, he has been assistant professor of Radiology, Medical Physics, and Cancer Research Center. Dr. Suzuki' research interests include computer-aided diagnosis, machine learning, artificial intelligence, and pattern recognition. He has published more than 100 papers (including 45 peer-reviewed journal papers) in these fields. He has been serving as a referee for more than 15 journals including IEEE Trans. Medical Imaging, IEEE Trans. Image Processing, and IEEE Trans. Neural Networks. He has received Paul C. Hodges Award, two Certificate of Merit Awards and Research Trainee Prize from RSNA, Young Investigator Award from Cancer Research Foundation, and Honorable Mention Poster Award at SPIE International Symposium on Medical Imaging. He has been a senior member of IEEE since 2004.

Hidenori Itoh received the Ph.D. degree from Nagoya University, Japan, in 1974. He is a professor in Nagoya Institute of Technology. His current research interests include image processing, human robotics, artificial life and automated reasoning.