# AN ANATOMY OF ABSTRACTION

Yamamoto, Akihiro
Department of Information Systems, Interdisciplinary Graduate School of Engineering Sciences, Kyushu University

https://doi.org/10.5109/13385

出版情報：Bulletin of informatics and cybernetics. 22 (3/4), pp.179-188, 1987-03. Research Association of Statistical Sciences
バージョン：
権利関係：

*Bulletin of Informatics and Cybernetics Vol. 22, No. 3~4, 1987*

# AN ANATOMY OF ABSTRACTION

By

**Akihiro YAMAMOTO***

**Abstract**

The use of the abstraction in resolution theorem proving is proposed in order to obtain some global proof plans. Then the problem is how to avoid the false proof. We give a new definition of the shapes of proofs so that we can make use of unifiers occurring in the proofs and discuss some typical abstractions. Then we can clarify that such abstractions are concerned with the unifiers. Moreover, we can give a complete strategy with matching. Our formalization will be useful to the further discussions on abstractions.

## 1. Introduction

The use of the abstraction in resolution theorem proving is proposed in [5]. An abstraction is a mapping which transforms a set of clauses $A$ into a simpler set of clauses $B$. The shapes of proofs and $m$-clauses are introduced so that a proof $U$ from $B$ has the same shape of $T$ for every proof $T$ from $A$. Thus after we generate a proof $U$ from $B$, we can make a proof $T$ from $A$ referring the shape of $U$. That is, we obtain some global plans of proving $A$.

Then it is a problem that there may exist a proof $U$ form $B$ such that no proofs from $A$ have the same shape of $U$. The use of more than one abstractions is proposed to avoid such proofs. However, even if sufficiently many abstractions are used, we cannot avoid the proofs in some cases. The cause of this incompleteness is that the shapes of proofs and the strategy using abstraction are formalized without considering of the unifiers in the proofs.

In the present paper, we give a solution of the problem described above in the case of using some typical abstractions by introducing unifiers. The use of unifiers requires more precise definitions of $m$-clauses and the shapes of proofs. Our definitions are analogues to those of Horn clauses and SLD-derivation [3], and they will be usefull for further discussions on abstractions. Then we can clarify that the abstractions which we are treating transform a set of clauses $A$ into a set of clauses $B$ so that the inconsistency of $B$ can be proved by more general unifiers than that of $A$. Moreover, we can give a complete strategy with matching in the case of the abstraction.

This paper is organized as follows. In the next section, we discuss the problem arising when we make use of abstractions. In section 3, we define $L$-clauses as lists

* Department of Information Systems, Interdisciplinary Graduate School of Engineering Sciences, Kyushu University 39, Kasuga, Fukuoka, 816 Japan.

of literals and give the shape of proofs using $L$-clauses. In section 4, we give the complete strategy with matching.

## 2. Theorem Proving with Abstraction

Let $L$ be a first order language. $\Pi$ denotes the sets of predicate symbols. For every $P\in\Pi$, $arity(P)$ denotes the arity of $P$. We assume that $P^i\in\Pi$ for every $P\in\Pi$ and $i=1, \cdots, arity(P)$. $A(L)$ and $Lit(L)$ denote the sets of atoms and literals respectively. $X$ and $X_I$ denote the sets of variables and variables subscripted by natural numbers, respectively. Let $\phi$ be a mapping from $A(L)$ to $A(L)$. Then we extend it to a mapping from $Lit(L)$ to $Lit(L)$ by defining the values for negative literals as follows:

$$\phi(\neg M)=\neg\phi(M) \quad \text{for} \quad M\in A(L).$$

Moreover, we assume as usual that $\phi(C)=\{\phi(L);\ L\in C\}$ for each clause $C$, and that $\phi(S)=\{\phi(C);\ C\in S\}$ for each set of clauses.

We illustrate the fundamental strategy in [5] by a simple example. Let us consider the following set $S$ of clauses:

$$C_1=\{\neg P(a, b), Q(c)\},$$
$$C_2=\{\neg P(a, c), Q(c)\},$$
$$C_3=\{P(x, x)\},$$
$$C_4=\{\neg Q(c)\}.$$

We try to prove the inconsistency of $S$ by the resolution principle. The procedure of the resolution principle contains the following searching procedures:

(2.1) Select two clauses from $S$,

(2.2) Select some literals from each clause selected by (2.1).

Note that we select more than one literals from a clause at a time instead of making a factor of the clause. There are 6 kinds of selections of clauses from $S$.

If we want to avoid the exhaustive search, we need some proof plans. We apply the following transformation $f^1$ to $S$ in order to obtain such plans.

$$f^1(P(x, y))=P^1(x),$$
$$f^1(Q(x, y))=Q(x, y).$$

Then $S$ is transformed as follows:

$$f^1(C_1)=\{\neg P^1(a), Q(c)\},$$
$$f^1(C_2)=\{\neg P^1(a), Q(c)\},$$
$$f^1(C_3)=\{P^1(x)\},$$
$$f^1(C_4)=\{\neg Q(c)\}.$$

Since $f^1(C_1)$ and $f^1(C_2)$ are the same, and the elements of $f^1(S)$ are less than those of $S$, we expect that the search space for the proof from $f^1(S)$ becomes smaller than that from $S$. Thus we can consider the following strategy. First we try to find a deriva-

tion of $\square$ from $f^1(S)$. Then after we find such a refutation, we delete the clauses irrelevant to the derivation of $\square$, and making a refutation for $S$ referring the shape of the proof from $f^1(S)$. We can generalize such a strategy for the mappings from $A(L)$ to $A(L)$ which satisfies some conditions. The precise conditions are defined in Section 3. We call such mappings abstraction mappings. The main algorithm in [5] is as follows:

ALGORITHM 1.

Step 1. Search for the refutation $T$ for $\phi(S)$. If $T$ is found, then go to Step 2.

Step 2. Find the clauses irrelevant to refuting and deleting them from $T$.

Step 3. Search for the refutation $U$ for $S$ by referring the shape of $T$. If $U$ is found, then stop, else go to Step 1 to find another refutation from $U$.

According to the discussions in [5], we choose $\phi$ which it maps distinct clauses onto the same clause by simplifying atoms. We expect such a mapping to reduce the search space for the procedure (2.1).

The main idea is that we can obtain some plans for solving a problem by simplifying the problem. However, we may obtain some plans which cannot help us in the search for refutations for the original set of clauses, because $\phi$ may map ununifiable literals onto unifiable ones and we may make illegal 'resolvents'. Such proofs are called 'false proofs'. Plaisted [5] proposed the strategy using more than one abstraction. It is called the matching strategy. To explain the strategy, we choose another abstraction $\phi'$. If there exist no proofs from $\phi(S)$ which have the same shape $T'$ of the proof from $\phi'(S)$, then there exist no proofs from $S$ which have the same shape of $T'$. Thus we can use the proofs from $\phi(S)$ in order to decide whether a proof from $\phi'(S)$ is a 'false proof' or not. However, this strategy is not complete. For example, we apply the following transformation $f^2$ to $S$.

$$f^2(P(x, y))=P^2(y),$$
$$f^2(Q(x, y))=Q(x, y).$$

Then the clauses in $S$ are transformed as follows:

$$f^2(C_1)=\{\neg P^2(b), Q(c)\},$$
$$f^2(C_2)=\{\neg P^2(c), Q(c)\},$$
$$f^2(C_3)=\{P^2(x)\},$$
$$f^2(C_4)=\{\neg Q(c)\}.$$

We can also derive $\square$ from $f^2(S)$. The proofs from $f^1(S)$ and $f^2(S)$ are illustrated in Fig. 1. The broken lines in the figure represent the shape correspondence, which shows that two proofs have the same shape. Since we cannot derive $\square$ from $S$, the proofs from both $f^1(S)$ and $f^2(S)$ are false. But the combination of $f^1(S)$ and $f^2(S)$ should have the same power as $S$ has, so the false proof should be avoided only by combining the proofs from $f^1(S)$ and $f^2(S)$.

In the above discussion on the shapes of proofs, we have not considered unifiers because the purpose of abstraction is transforming different clauses into the same clause in order to reduce the search space. We cannot continue the discussion under the

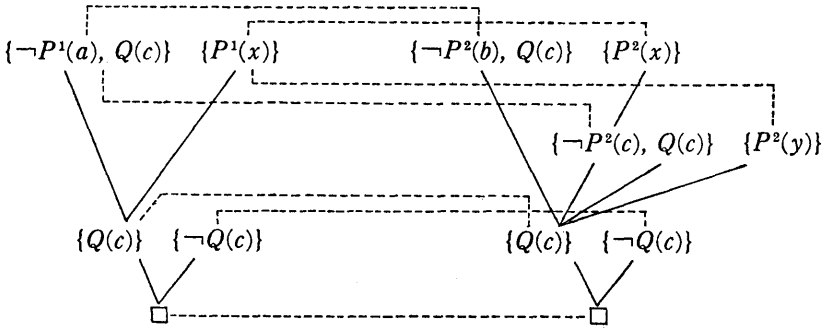


Fig. 1.

formalism any more.

Moreover, the property of $\phi$ is not only the cause of 'false proofs'. That is, selecting a literal resolved upon out of $\phi(C)$ may be selecting more than one literals resolved upon out of $C$, and so we make an illegal 'factor' of $C$ if these literals are not unifiable. Thus the property of $\phi$ described above is also the cause of the incompleteness for Algorithm 1. In order to remove the cause of incompleteness, $m$-clauses have been introduced in [5]. An $m$-clause is a multiset of literals. A clauses is a 'set' of literals for the efficiency of proofs. The introduction of $m$-clause suggests that we cannot discuss abstraction in possessing such efficiency.

Thus, in the next section, we neglect the reducing of search space and give a precise definition attending to the unifiers in the derivation.

### 3. Unifiers in Abstraction

The introduction of the $m$-clause also suggests the possibility to make use of the abstraction for the procedure (2.2). Then we need to indicate each literal in each $m$-clause for the purpose. Thus we introduce the descriptor of each literal in a $m$-clause by using a natural number, and add the descriptors of literals resolved upon to the shape of proofs. Moreover, we admit the iterative use of initial clause as in the SLD-resolution. Such a precise definition enables us to discuss the unifiers in derivations and to refine the matching strategy.

DEFINITION 1. An $L$-clause is a sequence of literals, and denoted by $[L_1, \cdots, L_n]$ where $L_i \in Lit(L)$ for $i=1, \cdots, n$.

Let $C$ and $D$ be clauses. Then $C+D$ denotes the $L$-clause obtained by appending $D$ to $C$. If $C=[L_1, \cdots, L_n]$, then $C-[L_{i(1)}, \cdots, L_{i(m)}]$ is an $L$-clause obtained by deleting the $i(j)$-th element of $C$ for $j=1, \cdots, m$, simultaneously.

DEFINITION 2. Let $C$ and $D$ be two $L$-clauses that share no variables. Then a *resolvent of* $C$ and $D$ is an $L$-clause given as follows:

$$((C-[L_{i(1)}, \cdots, L_{i(l)}])+(D-[M_{j(1)}, \cdots, M_{j(m)}]))\sigma,$$

where $\sigma$ is an mgu of the set

$$\{L_{i(1)}, \cdots, L_{i(l)}, \neg M_{j(1)}, \cdots, \neg M_{j(m)}\}.$$

DEFINITION 3. Let $C$ be an $L$-clause. Then a *variant* of $C$ is an $L$-clause obtained by renaming variables in $C$. If no variables in $X_I$ occurs in $C$, then an *i-variant* of $C$ for a natural number $i$ is an $L$-clause obtained by subscripting variables in $C$ by $i$.

Our definition of the proof is the extension of that for SLD-derivation. We admits the iterative use of clauses in the given set in order not to rename variables at each step of making resolvents.

DEFINITION 4. Let $S$ be a set of $L$-clauses and $C$ be an $L$-clause. Then a proof of $C$ from $S$ is a finite sequence of quadruplets $\langle C_i, e_i, s_i, \sigma_i\rangle$ $(i=1, \cdots, n)$ that satisfies following conditions.

1) $C_i$ is a $L$-clause, $e_i$ is a natural number, $s_i$ is a finite sequence of natural numbers, $\sigma_i$ is a substitution, and $C_n$ is $C$.

2) There is a number $k$ such that $C_i$, which is called an *input clause* of the proof, is an *i-variant* of a clause in $S$ for every $i=1, \cdots, k$,

$$k<e_i\leqq n, \quad \sigma_i=\varepsilon \quad \text{for } i=1, \cdots, k,$$
$$i<e_i\leqq n \quad \text{for } i=k+1, \cdots, n-1,$$

and

$$e_i>n.$$

3) For every $i=1, \cdots, n$, there exists a unique $j$ such that $e_i=e_j$, $C_i$ and $C_j$ which do not share variables, and $C_{e_j}$ which is a resolvent of $C_i$ and $C_j$ in the form:

$$C_{e_i}=((C_i-[L_{s_i(1)}, \cdots, L_{s_i(n_i)}])+(C_j-[M_{s_j(1)}, \cdots, M_{s_j(n_j)}]))\sigma_{e_i},$$

where $\sigma_{e_j}$ is an mgu of the set

$$\{L_{s_i(1)}, \cdots, L_{s_i(n_i)}, \neg M_{s_j(1)}, \cdots, \neg M_{s_j(n_j)}\}.$$

We make the shapes of proofs conclude the positions of literals resolved upon.

DEFINITION 5. The shape of the proof $\{\langle C_i, e_i, s_i, \sigma_i\rangle\}$ $(i=1, \cdots, n)$ is the sequence of $\{\langle e_i, s_i\rangle\}$ $(i=1, \cdots, n)$.

Note that for every proof $\{\langle C_i, e_i, s_i, \sigma_i\rangle\}$ $(i=1, \cdots, n)$ there exists a sequence such that

$$\begin{aligned} n&=e_{i(1)}, \\ i(m)&=e_{i(m+1)}. \end{aligned} \tag{3.1}$$

DEFINITION 6. The depth of a proof $\{\langle C_i, e_i, s_i, \sigma_i\rangle\}$ $(i=1, \cdots, n)$ is the longest sequence that satisfies the condition (3.1).

DEFINITION 7. The subproof of the proof $T$ is a subsequence of $T$ that satisfies the conditions of Definition 4.

Let $\phi$ be a mapping from $A(L)$ to $A(L)$. Then we extend it to the mapping from $Lit(L)$ to $Lit(L)$ in the same way as in Section 1, and assume that $\phi(C)=[\phi(L_1), \cdots, \phi(L_n)]$ for an $L$-clause $C$ of the form $[L_1, \cdots, L_n]$.

DEFINITION 8. A mapping $\phi$ from $A(L)$ to $A(L)$ is an abstraction mapping if it satisfies the following conditions.

1) If $L$-clauses $C_1$ and $C_2$ have a resolvent $C_3$, then $\phi(C_1)$ and $\phi(C_2)$ have a resolvent $D_3$ such that

$$\phi(C_3)=D_3\theta \quad \text{for some } \theta.$$

2) $\phi(\square)=\square$.

When we refer the shape of a proof in Algorithm 1, we delete the irrelevant $L$-clause in the derivation. Thus we can introduce our $L$-clases and our proofs in Algorithm 1. Then the following theorem justifies the completeness of Algorithm 1.

THEOREM 1. (Plaisted). *Let $\phi$ be an abstraction mapping, and $S$ be a set of $L$-clauses. If there exists a proof figure $T$ from $S$ to $C$, then there exist an $L$-clause $D$ and a proof $U$ from $\phi(S)$ to $D$ which satisfy following conditions.*

1) *$\phi(C)=D\theta$ for some $\theta$.*

2) *The shapes of $T$ and $U$ are the same.*

3) *If the input $L$-clauses of $T$ is $C_i$ for $i=1, \cdots, k$ and those of $U$ is $D_i$ for $i=1, \cdots, k$, then*

$$D_i=\phi(C_i) \quad \text{for } i=1, \cdots, k.$$

From now on, we discuss the following abstractions $f_P^i$ for $P\in\Pi$ and $0\leqq i\leqq arity(P)$.

$$f_P^i(P(t^1, \cdots, t^n))=P^i(t^i) \qquad \text{where } n=arity(P),$$
$$f_P^i(Q(t^1, \cdots, t^m))=Q(t^1, \cdots, t^m) \qquad \text{where } m=arity(Q)$$

for each predicate symbol $Q$ that is not $P$.

That is, $f_P^i$ decreases the arity of the predicate symbol $P$.

We introduce the following notations for each substitution.

NOTATION. Let $\theta=\{x_1\leftarrow t_1, \cdots, x_n\leftarrow t_n\}$. Then

$$U(\theta)=\{x_1, \cdots, x_n\},$$
$$V(\theta)=\{\text{all variables occurring in } t_1, \cdots, t_n\}.$$

In general, we can assume that each mgu is computed only by the algorithm in [6]. Such an mgu $\sigma$ is idempotent, that is, $U(\sigma)\cap V(\sigma)=\phi$. Moreover, $\sigma(C)$ has such a property.

LEMMA 1. *If there is a proof of $C$ from $S$, then $\sigma(C)$ for the proof is idempotent.*

LEMMA 2. *If $e_i=e_j=k$ for the proof $\langle C_i, e_i, s_i, \sigma_i\rangle$ $(i=1, , \cdots, n)$, then*

$$U(\sigma(C_i))\cap U(\sigma_k)=\phi,$$
$$U(\sigma(C_j))\cap U(\sigma_k)=\phi.$$

The abstraction using $f_P^i$ is considered to eliminate unification of the $i$-th argument of the predicate $P$. The following lemma justifies the consideration.

LEMMA 3. *Let $T$ be the proof of $C$ from $S$. Let $U$ be the proof of $D$ from $f_P^i(S)$, and $\theta$ be the substitution that satisfy the conditions in Theorem 1. Then there exist a substitution $\delta$ and $\mu$ such that*

$$\sigma(C)=\sigma(D)\delta,$$
$$\delta=\theta\mu.$$

DEFINITION 7. Let $\sigma_1, \cdots, \sigma_k$ be substitutions. Then they are *unifiable* if there exists a substitution $\theta$ such that

$$\sigma_1\theta = \cdots = \sigma_k\theta .$$

We can show the following relations between the unifiers in the proof from $S$ and those of the proof from $f_P^i(S)$.

THEOREM 2. *Let $T$ be the proof of $C$ from $S$ and $U$ be the derivation of $D$ from $f_P^i(S)$ that satisfies the conditions in Theorem* 1. *Then $\sigma(E)$ and $\sigma(D)$ is unifiable for any clause $E$ in $T$.*

PROOF. It suffices to show that

$$\sigma(E)\sigma(C)=\sigma(C),$$
$$\sigma(D)\sigma(C)=\sigma(C).$$

By the definition of the proof and $\sigma(C)$, $\sigma(C)=\sigma(E)\sigma'$ for some $\sigma'$. Then by Lemma 1,

$$\sigma(E)\sigma(C)=\sigma(E)\sigma(E)\sigma'=\sigma(E)\sigma'=\sigma(E).$$

By Lemma 3, $\sigma(C)=\sigma(D)\delta$ for some $\delta$. Then

$$\sigma(D)\sigma(C)=\sigma(D)\sigma(D)\delta=\sigma(C).$$

We can improve Algorithm 1 by using Theorem 1.

ALGORITHM 2.

Step 1. Search for the refutation $U$ for $f_P^i(S)$. If $U$ is found, then go to Step 2.

Step 2. Find the clauses irrelevant to refuting $f_P^i(S)$, delete them from $U$, and go to Step 3.

Step 3. Try to construct the refutation $T$ for $S$ such that the shapes of $Y$ and $U$ are the same. If $\sigma(E)$ and $\sigma(\square)$ (for $U$) are not unifiable in some clause in $T$ under construction, then go to Step 1 to find another $U$.

## 4. A Complete Strategy with Matching

In this section, we consider the strategy with matching. Let $S$ be a set of $L$-clauses and $T^i$ be the proof from $f_P^i(S)$ for $i=1, \cdots, n$, where $n=arity(P)$. Then procedure 'match' in [5] decides whether $T^1, \cdots, T^n$ satisfies the following conditions:

(4.1) $T^1, \cdots, T^n$ have the same shape.

(4.2) There exists an $L$-clause $C_j$ such that $C_j^i=f_P^i(C_j)$ for each input $L$-clause $C_j^i$.

(4.3) For each $n$-tuples $\langle C_j^1, \cdots, C_j^n\rangle$ of $L$-clauses of $T^1, \cdots, T^n$, respectively, there exist an $L$-clause $C_j$ and substitutions $\theta^1, \cdots, \theta^n$ such that

$$C_j^1=f_P^1(C_j)\theta^1, \cdots, C_j^n=f_P^n(C_j)\theta^n .$$

The condition (4.3) is proposed in order to avoid the false proofs. However, this condition is so weak that it cannot avoid the false proof of the examples in Section 2. In the example, the substitutions $\sigma(\square)$ for $f_P^1(S)$ and $\sigma(\square)$ for $f_P^2$ is not unifiable because $x$ is substituted $a$ and $c$ respectively. By generalizing such consideration, we can strengthen the condition (4.3) so that we can avoid any false proof. Such a condition is given by the following theorem.

THEOREM 3. *Let $T^i$ be a proof from $f_P^i(S)$ for $i=1, \cdots, n$ and $T^1, \cdots, T^n$ satisfy*

*the condition* (4.1) *and* (4.2). *Then* $\sigma(D^i)$ *for* $i=1, \cdots, n$ *are unifiable iff there exists a proof figure from* $S$ *that have the same shape of* $T^i$.

PROOF. By Theorem 2, it suffices to show the only-if part. We prove by induction on the depth of $T^i$ that

(4.4) there exists a derivation from $S$ which have the same shape of $T^i$ and that there exists a substitution $\delta$ such that

$$f_P^i(C)\delta = D^i\theta .$$

Suppose first the depth of $T^i$ is 1 for $i=1, \cdots, n$. Then (4.4) holds by the condition (4.2).

Now suppose (4.4) holds for $k-1$ and depth of $T^i$ is $k$. Since the shape of $T_i$ are the same, we can assume that the shape is $\{\langle e_j, s_j\rangle\}$ $(j=1, \cdots, k)$ and $T^i=\{\langle C_j^i, e_j, s_j, \sigma_j^i\rangle\}$ $(j=1, \cdots, k)$. Then there exist $p$ and $q$ such that $1\leqq p$, $q<k$ and $e_p=e_q=k$. Thus

$$\sigma(D^i)=(\sigma(C_p^i)\cup\sigma(C_q^i))\sigma_k^i .$$

We construct a substitution $\zeta$ such that

$$\sigma(C_p^1)\zeta=\cdots=\sigma(C_p^n)\zeta .$$

Let $\zeta^i=\sigma_k^i\theta-\{v\leftarrow s\,;\ v\notin V(\sigma(C_p^i))$ and $v\notin U(\sigma_k^i)\}$ where $\theta$ is the substitution obtained by the induction hypothesis for (4.4). By Lemma 1 and the definition of $\sigma(D^i)$, $\sigma(C_p^i)\sigma_k^i\theta \subset\sigma(D^i)$. We put

$$\zeta^i=\{x^{i(1)}\leftarrow t^{i(1)}\theta, \cdots, x^{i(r)}\leftarrow t^{i(r)}\theta, y^{i(1)}\leftarrow s^{i(1)}, \cdots, y^{i(m)}\leftarrow s^{i(m)}\}.$$

Then

$$\{x^{i(1)}\leftarrow t^{i(1)}\theta, \cdots, x^{i(r)}\leftarrow t^{i(r)}\theta\}\subset\sigma(D^i)\theta , \tag{4.5}$$

$$\{y^{i(1)}\leftarrow s^{i(1)}, \cdots, y^{i(m)}\leftarrow s^{i(m)}\}\subset\theta , \tag{4.6}$$

$$y^{i(j)}\in V(\sigma(C_p^i)) \quad\text{and}\quad y^{i(j)}\notin U(\sigma(C_p^i)\sigma_k^i) \quad\text{for } j=1, \cdots, m .$$

Let

$$\zeta=(\bigcup_{i=1}^n\zeta^i)\cup\zeta' ,$$

$$\zeta'=\{v\leftarrow t\in\sigma(D^i)\theta\,;\ v\in(\bigcup_{i=1}^n U(\sigma(C_p^i)))-(\bigcup_{i=1}^n U(\zeta^i))\}.$$

We show that $\zeta$ is a well-defined substitution.

Case 1. $(x\leftarrow s)\in\zeta^i$ and $(x\leftarrow u)\in\zeta^j$.

If $x\in U(\sigma_k^i)\cap U(\sigma_k^j)$, then $(x\leftarrow s)\in\sigma(D^i)\theta$ and $(x\leftarrow u)\in\sigma(D^j)\theta$ by (4.6). Thus $s=u$.

If $x\in U(\sigma_k^i)$ and $x\notin U(\sigma_k^j)$, then $x\leftarrow u\in\theta$. Since $x\notin U(\sigma(C_p^j)\sigma_n^j)$,

$$x\leftarrow u\in\sigma(C_p^j)\sigma_n^j\theta\subset\sigma(D^j)\theta .$$

Thus $s=u$.

Case 2. $(x\leftarrow s)\in\zeta^i$ and $(x\leftarrow u)\in\zeta'$.

Since $(x\leftarrow u)\in\sigma(D_i)\theta$, $s=u$ in the same way of Case 1.

Then $\zeta$ satisfies the condition (4.2) by the way of its construction. We can construct a substitution $\eta$ such that

$$\sigma(C_q^1)\eta=\cdots=\sigma(C_q^n)\eta .$$

Moreover $U(\zeta)\cap U(\eta)=\phi$. Let the subproofs of $C_p^i$ and $C_q^i$ be $T_p^i$ and $T_q^i$, respectively.

Then each depth of $T_p^i$ and $T_q^i$ is less than $n$. Thus by the induction hypothesis, there exist derivations of $C_p$ and $C_q$ from $S$ such that its shape is same as that of $T_p^i$ and $T_q^i$ and

$$f_P^i(C_p)\delta_p = C_p^i\xi,$$
$$f_P^i(C_q)\delta_q = C_q^i\eta,$$
$$U(\delta_p)\cap U(\delta_q)=\phi.$$

Then we can put

$$C_p=[L_1, \cdots, L_h],$$
$$C_q=[M_1, \cdots, M_k].$$

$\zeta\cup\eta$ is a unifier of

$$\{L^i_{s_p(1)}, \cdots, L^i_{s_p(n_p)}, \neg M^i_{s_q(1)}, \cdots, \neg M^i_{s_q(n_q)}\}.$$

If their predicate symbol is not $P$, then $\delta_p\cup\delta_q$ is a unifier of

$$\{L_{s_p(1)}, \cdots, L_{s_p(n_p)}, \neg M_{s_q(1)}, \cdots, \neg M_{s_q(n_q)}\}.$$

If their predicate symbol is $P$, then $\zeta\cup\eta$ is a unifier of

$$\{P(t_h^1, \cdots, t_h^n), \cdots, P(s_k^1, \cdots, s_k^n)\}$$

and $\delta_p\cup\delta_q$ is a unifier of

$$\{L_{s_p(1)}, \cdots, L_{s_p(n_p)}, \neg M_{s_q(1)}, \cdots, \neg M_{s_q(n_q)}\}.$$

Thus $C_p$ and $C_q$ has a resolvent $C$ and the shape of the derivation of $C$ is same as that of $T^i$. Moreover, since $C_p^i\zeta\mu=C_p^i\theta$ and $C_q^i\eta\mu=C_q^i\theta$, there exists a substitution $\delta$ such that

$$f_P^i(C)=D^i\theta.$$

## 5. Concluding Remarks

The main purpose of using abstractions is obtaining the global search plans. Then it is desirable that the abstracted space is smaller than the original space. It is founded only on the fact that some clauses becomes variant and they can be regarded as the same clause. However, after the simplification, we cannot remove tautologies. Thus we make resolvents of a clause and itself. Simplified clauses may have more resolvents than the original ones. Thus many false proofs will be made, and the strategy may not decrease the search space in the above example.

Our formalization will be useful to discuss the efficiency of abstractions. Such considerations will give some sufficient conditions for selecting abstractions efficiently.

We can consider an applications of the abstractions to logic programming. However, $f_P^i$ is not be suitable for the execution planning because of the above mentioned efficiency. The essential point is the fact that data are passed with variables. thus the decomposition of arguments of predicates contributes the execution plans. We hope that abstractions will useful for developing programs. We are preparing a discussion on applying abstractions to logic programming with equality. Our formalization of the proof figure enables us to make use of the relations of unifications and equality theory

in the discussions on abstractions.

**Acknowledgements**

The author would like to thank Prof. Setsuo Arikawa for his introduction to the abstractions and his advice. The author also wishes to express his thanks to Dr. Makoto Haraguchi for many discussions on the theorem proving and analogy.

**References**

[1] HAGIYA, M.: *Foundation of Prolog*, Information Processing, **25**, 12 (1984), 1336-1344.
[2] HARAGUCHI, M.: *Towards a Mathematical Theory of Analogy*, Bull. Inform. Cybernetics, **21** (1985), 29-56.
[3] LLOYD, J. W.: *Foundations of Logic Programming*, Springer-Verlag, (1984).
[4] MARTELLI, A. and MONTANARI, U.: *An Efficient Unification Algorithm*, ACM TOPLAS, **4**, 2 (1982), 258-282.
[5] PLAISTED, D. A.: *Theorem Proving with Abstraction*, Artificial Intelligence, **16** (1981), 47-108.
[6] ROBINSON, J. A.: *A machine Oriented Logic based on Resolution Principle*, JACM, **12**, 1 (1965), 23-41.

*Communicated by S. Arikawa*
*Received September 30, 1986*