ソフトウェア基礎論 15－1
（1985. 12. 20）

Program Transformations based on Term Rewriting Systems

Yoshihito TOYAMA

NTT Electrical Communications Laboratories
Midori-cho, Musashino-shi, 180 Japan

Abstract

Simple methods for testing equivalence of term rewriting systems are presented. Using the Church-Rosser property, sufficient conditions for equivalence in a restricted domain of term rewriting systems are proved. These conditions can be effectively applied to obtain equivalence transformation rules for term rewriting systems. Program transformations based on these rules are discussed.

1. Introduction

The concept of the equivalence in a restricted domain of two term rewriting systems plays an important role in transforming recursive programs [2][12] and proving an equation in abstract data types [3][5][6][9]. For example, consider a recursive program computing the factorial function on the set **N** of natural numbers represented by 0, S(0), S(S(0)), ...;

F(x)=IF equal(x,0) THEN S(0) ELSE x*F(x-S(0)).

By using the successor function S, we can also define the factorial function by;

F(0)=S(0),
F(S(x))=S(x)*F(x).

Regarding equations as rewriting rules from the left hand side to the right hand side, we can obtain two term rewriting systems [4][5] from the above two definitions. The first term rewriting system can reduce "F(M)" to "IF equal(M,0) THEN S(0) ELSE M*F(M-S(0))" for any term M, but the second system can not reduce "F(M)" unless M is either "0" or the form of "S(M')". Therefore the two term rewriting systems produce different results in the reduction of "F(M)", although they can reduce "F(M)" to the same result unless M can be reduced to a natural number. Thus, the equivalence for the recursive programs must be regarded as the equivalence in the restricted domain **N** for the term rewriting systems.

We consider in this paper sufficient conditions for the equivalence in a restricted domain for two term rewriting systems. We first treat this problem in an abstract framework and show sufficient conditions for two

abstract reduction systems. It is shown how one can formally validate the equivalence in the restricted domain for term rewriting systems by using these conditions. Finally, the problems related to the rules for transforming programs described by Burstall and Darlington [2], and Scherlis [12] are discussed.

2. Reduction Systems

We explain notions of reduction systems and give definitions for the following sections. These reduction systems have only an abstract structure, thus they are called abstract reduction systems [4][7][11].

A reduction system is a structure $R=\langle A,\rightarrow\rangle$ consisting of some object set A and some binary relation $\rightarrow$ on A, called a reduction relation. A reduction (starting with x_0) in R is a finite or infinite sequence $x_0\rightarrow x_1\rightarrow x_2\rightarrow\ldots$. The identity of elements of A (or syntactical equality) is denoted by $\equiv$. $\xrightarrow{*}$ is the transitive reflexive closure of $\rightarrow$, $\xrightarrow{=}$ is the reflexive closure of $\rightarrow$, and $=$ is the equivalence relation generated by $\rightarrow$ (i.e., the transitive reflexive symmetric closure of $\rightarrow$). If $x\in A$ is minimal with respect to $\rightarrow$, i.e., $\neg\exists y\in A[x\rightarrow y]$, then we say that x is a normal form, and let NF be the set of normal forms. If $x\xrightarrow{*}y$ and $y\in NF$ then we say x has a normal form y and y is a normal form of x.

Definition. $R=\langle A,\rightarrow\rangle$ has the Church-Rosser property, or Church-Rosser, (denoted by CR(R)) iff

$$\forall x,y,z\in A[x\xrightarrow{*}y\wedge x\xrightarrow{*}z \Rightarrow \exists w\in A, y\xrightarrow{*}w\wedge z\xrightarrow{*}w].$$

The following properties are well known [1][4][7].

Property 2.1. Let R have the Church-Rosser property, then,

(1) $\forall x,y\in A[x=y \Rightarrow \exists w\in A, x\xrightarrow{*}w\wedge y\xrightarrow{*}w]$,

(2) $\forall x,y\in NF[x=y \Rightarrow x\equiv y]$,

(3) $\forall x\in A,\forall y\in NF[x=y \Rightarrow x\xrightarrow{*}y]$.

3. Basic Results

Let $R_1=\langle A,\rightarrow_1\rangle$, $R_2=\langle A,\rightarrow_2\rangle$ be two abstract reduction systems having the same object set A, and let $\xrightarrow[i]{*}$, $=_i$ and NF_i be the transitive reflexive closure, the equivalence relation and the set of normal forms in R_i respectively (i=1,2). Note that $=$ and $\xrightarrow{*}$ are subsets of $A\times A$: for example, $=_1\subseteq=_2$ means that the set $=_1$ is contained in the set $=_2$.

Let B, C be any subsets of the object set A. We write $=_1 = =_2$ (in B) for

$\forall x,y\in B[x \underset{1}{=} y \iff x \underset{2}{=} y]$, and say R_1 and R_2 are equivalent in the restricted domain B for the equivalence relation. We will show sufficient conditions for $\underset{1}{=} = \underset{2}{=}$ (in B).

Lemma 3.1. Let R_1, R_2 satisfy the following conditions:

(1) $\underset{1}{=} \subseteq \underset{2}{=}$,

(2) $\underset{1}{=} = \underset{2}{=}$ (in C),

(3) $\forall x\in B, \exists y\in C[x \underset{1}{=} y]$.

Then $\underset{1}{=} = \underset{2}{=}$ (in B).

Proof. Prove $\forall x,y\in B[x \underset{1}{=} y \iff x \underset{2}{=} y]$. $\Rightarrow$ is trivial from condition(1), hence we show $\Leftarrow$. Assume $x \underset{2}{=} y$ where $x,y\in B$. By using condition(3), there are some elements $z,w\in C$ such that $x \underset{1}{=} z$ and $y \underset{1}{=} w$. Since $x \underset{2}{=} z$ and $y \underset{2}{=} w$ are obtained from condition(1) , $z \underset{2}{=} w$ can be derived from $z \underset{2}{=} x \underset{2}{=} y \underset{2}{=} w$. From condition(2), $z \underset{1}{=} w$ holds. Therefore $x \underset{1}{=} y$ from $x \underset{1}{=} z \underset{1}{=} w \underset{1}{=} y$. □

If R_2 has the Church-Rosser property, we can modify condition(2) in Lemma 3.1 as follows.

Theorem 3.1. Assume the following conditions:

(1) $\underset{1}{=} \subseteq \underset{2}{=}$,

(2) $CR(R_2)$ and $C\subseteq NF_2$,

(3) $\forall x\in B, \exists y\in C[x \underset{1}{=} y]$.

Then $\underset{1}{=} = \underset{2}{=}$ (in B).

Proof. Show condition(2) of Lemma 3.1, i.e., $\forall x,y\in C[x \underset{1}{=} y \iff x \underset{2}{=} y]$, from the above conditions. $\Rightarrow$ is trivial from condition(1), hence we prove $\Leftarrow$. By using property 2.1(2) and condition(2), $x \underset{2}{=} y \Rightarrow x \equiv y$ for any $x,y\in C$. Therefore $x \underset{1}{=} y$. □

4. Term Rewriting Systems

In this section we will explain term rewriting systems that are reduction systems having a term set as an object set A.

Let V be a set of variable symbols denoted by x,y,z,..., and let F be a set of function symbols denoted by f,g,h,..., where $F\cap V=\emptyset$. T(F,V) is the set of terms on F and V. Let T(F) be the set of terms having no variable symbols. T is used for T(F,V) when F and V are clear from the context.

If M is a term and θ is a substitution, then $M\theta$ is the result of applying θ to M; that is, each variable of M is replaced by the term specified in θ.

Consider an extra constant □ called a hole and the set $T(F\cup\{\square\},V)$. Then $C\in T(F\cup\{\square\},V)$ is called a context on F. The notation C[ ] denotes a context

containing precisely one hole, and C[N] denotes the result of placing N in the hole of C[ ].

A rewriting rule on T is a pair $\langle M_l, M_r \rangle$ of terms in T such that $M_l \notin V$ and any variable in M_r also occurs in M_l. The notation $\triangleright$ denotes a set of rewriting rules on T and we write $M_l \triangleright M_r$ for $\langle M_l, M_r \rangle \in \triangleright$. A $\rightarrow$redex, or redex, is a term $M_l\theta$ where $M_l \triangleright M_r$, and in this case $M_r\theta$ is called a $\rightarrow$contractum, or contractum, of $M_l\theta$. The set $\triangleright$ of rewriting rules on T defines a reduction relation $\rightarrow$ on T as follows:

$M \rightarrow N$ iff $M \equiv C[M_l\theta]$, $N \equiv C[M_r\theta]$, and $M_l \triangleright M_r$
for some M_l, M_r, C[ ], and θ.

Definition. A term rewriting system R on T is a reduction system $R = \langle T, \rightarrow \rangle$ such that the reduction relation $\rightarrow$ is defined by a set $\triangleright$ of rewriting rules on T. If R has $M_l \triangleright M_r$, then we write $M_l \triangleright M_r \in R$.

If every variable in term M occurs only once, then M is called linear. We say that R is linear iff $\forall M \triangleright N \in R$, M is linear.

Let $M \triangleright N$ and $P \triangleright Q$ be two rules in R. We assume that we have renamed variables appropriately, so that M and P share no variables. Assume $S \neq V$ is a subterm occurrence in M, i.e., $M \equiv C[S]$, such that S and P are unifiable, i.e., $S\theta \equiv P\theta$, with a minimal unifier θ [4][8]. Since $M\theta \equiv C[S]\theta \equiv C\theta[P\theta]$, two reductions starting with $M\theta$, i.e., $M\theta \rightarrow C\theta[Q\theta] \equiv C[Q]\theta$ and $M\theta \rightarrow\!\!\!\!\rightarrow N\theta$, can be obtained by using $P \triangleright Q$ and $M \triangleright N$. Then we say that the pair $\langle C[Q]\theta, N\theta \rangle$ of terms is critical in R [4][5]. We may choose $M \triangleright N$ and $P \triangleright Q$ to be the same rule, but in this case we shall not consider the case $S \equiv M$, which gives trivial pairs $\langle N, N \rangle$. If R has no critical pair, then we say that R is non-overlapping (with itself) [4][5][8][13].

The critical pair for two term rewriting systems R_1 and R_2 can be defined in the same way. Let $M_1 \triangleright N$ and $P_2 \triangleright Q$ be in R_1 and in R_2 respectively. Then we say that the above pair $\langle C[Q]\theta, N\theta \rangle$ is critical between R_1 and R_2. If there is no critical pair between R_1 and R_2, then we say that R_1 and R_2 are non-overlapping with each other [13].

The following sufficient conditions for the Church-Rosser property are well known [4][5][8].

Condition 4.1. Let R be strongly normalizing. If for any critical pair $\langle P, Q \rangle$ in R, P and Q have the same normal form, then R has the Church-Rosser property.

Condition 4.2. Let R be linear and non-overlapping. Then R has the Church-Rosser property.

Let $R_1=\langle T,\xrightarrow[1]{}\rangle$ with $\rhd_1$ and $R_2=\langle T,\xrightarrow[2]{}\rangle$ with $\rhd_2$. Then their union $R_1\cup R_2$ is defined by $R_1\cup R_2=\langle T,\longrightarrow\rangle$ with $\rhd=\rhd_1\cup\rhd_2$. The next condition is described in [13] by using the commutativity of R_1 and R_2.

Condition 4.3. Let the two linear term rewriting systems R_1 and R_2 have the Church-Rosser property and let them be non-overlapping with each other. Then $R_1\cup R_2$ has the Church-Rosser property.

5. Equivalence Transformation Rules

In this section, let us consider the correctness of the program transformation rules discussed by Burstall and Darlington [2], and Scherlis [12]. They showed in many examples that by using their rules, a recursive program can be transformed to an improved one computing the same function. This problem can be seen as one of equivalence transformations for term rewriting systems. We will give a formal proof, based on the equivalence in a restricted domain, for the correctness of transformation rules.

Let $R=\langle T(F,V),\longrightarrow\rangle$ with $\rhd$, and let H be a subset of F such that H contains all function symbols appearing in the rewriting rules of R. We propose the equivalence transformation rules in the restricted domain T(H) for R. Set $R_0=R$ and $F_0=H$, and then we transform $R_n=\langle T(F,V),\xrightarrow[n]{}\rangle$ with $\rhd_n$ to $R_{n+1}=\langle T(F,V),\xrightarrow[n+1]{}\rangle$ with $\rhd_{n+1}$ by using the following rules:

(1) **Definition:** Add a new rewriting rule $g(x_1,\ldots,x_k)\rhd Q$ to R_n, where $g\in F-F_n$, $g(x_1,\ldots,x_k)$ is linear, and $Q\in T(F_n,V)$. Thus, $\rhd_{n+1}=\rhd_n\cup\{g(x_1,\ldots,x_k)\rhd Q\}$. Set $F_{n+1}=F_n\cup\{g\}$.

(2) **Addition:** Add a new rule $P\rhd Q$ to R_n, where $P\underset{n}{=}Q$ and $P,Q\in T(F_n,V)$. Thus, $\rhd_{n+1}=\rhd_n\cup\{P\rhd Q\}$. Set $F_{n+1}=F_n$.

(3) **Elimination:** Remove a rule $P\rhd Q$ from R_n. Thus, $\rhd_{n+1}=\rhd_n-\{P\rhd Q\}$. Set $F_{n+1}=F_n$.

Remark. The above three rules include the transformation rules suggested by Scherlis [12]: we can show easily that transformations by the rules in [12] can be obtained by using the above rules.

$R_n\overset{i}{==>}R_{n+1}$ shows that R_n is transformed to R_{n+1} by rule(i) (i=1,2,or 3). $R_n==>R_{n+1}$ shows that R_n is transformed to R_{n+1} by rule(1), (2), or (3). $R_m\overset{*}{\underset{i}{==>}}R_n$ and $R_m\overset{*}{==>}R_n$ $(m\leq n)$ are the transitive reflexive closure of the two relations.

Lemma 5.1. If $R_1 \overset{i}{\Longrightarrow} R_2 \overset{j}{\Longrightarrow} R_3$ $(i>j)$, then there is some R_2' such that $R_1 \overset{j}{\Longrightarrow} R_2' \overset{i}{\Longrightarrow} R_3$.

Proof. From the definition of the rules, it is obvious. □

Lemma 5.2. Let $R \overset{*}{\Longrightarrow} R'$. Then there exists a transformation sequence from R to R' such that $R \overset{*}{\underset{1}{\Longrightarrow}} R_a \overset{*}{\underset{2}{\Longrightarrow}} R_b \overset{*}{\underset{3}{\Longrightarrow}} R'$.

Proof. By using Lemma 5.1 repeatedly, we can construct a sequence $R \overset{*}{\underset{1}{\Longrightarrow}} R_a \overset{*}{\underset{2}{\Longrightarrow}} R_b \overset{*}{\underset{3}{\Longrightarrow}} R'$ from $R \overset{*}{\Longrightarrow} R'$. □

Theorem 5.1. Let $R_0 \overset{*}{\Longrightarrow} R_n$, where R_0 is a linear system and $CR(R_0)$. Let $G \subseteq H$ and $T(G) \subseteq NF_0$. Assume the following property for R_0 and R_n:

$\forall M \in T(H)\ \exists N \in T(G)[M \underset{i}{=} N]$ $\quad (i=0,n)$.

Then $\underset{0}{=} = \underset{n}{=}$ (in $T(H)$).

Proof. By Lemma 5.2, we may assume that $R_0 \overset{*}{\underset{1}{\Longrightarrow}} R_a \overset{*}{\underset{2}{\Longrightarrow}} R_b \overset{*}{\underset{3}{\Longrightarrow}} R_n$. To prove the theorem we will show that $\underset{0}{=} = \underset{a}{=}$ (in $T(H)$) and $\underset{a}{=} = \underset{n}{=}$ (in $T(H)$). Consider $R_0 \overset{*}{\underset{1}{\Longrightarrow}} R_a$. It is clear that $\underset{0}{=} \subseteq \underset{a}{=}$. Let $\triangleright' = \{g_1(x_1,..,x_{n1}) \triangleright Q_1, .., g_a(x_1,..,x_{na}) \triangleright Q_a\}$ be the set of new rules added to R_0 through $R_0 \overset{*}{\underset{1}{\Longrightarrow}} R_a$. Define R' by $\triangleright'$. Then R_a is the union of R_0 and R'. Since R' is linear and non-overlapping, by using condition 4.2, CR(R') can be proved. R_0 and R' are non-overlapping with each other since the function symbols $g_0,..,g_a$ do not appear in the rewriting rules in R_0. Hence, by condition 4.3, $CR(R_a)$ is obtained. From the definition of rule (1), $T(G) \subseteq NF_a$. $\forall M \in T(H)\ \exists N \in T(G)[M \underset{0}{=} N]$ has been assumed. Hence, by using Theorem 3.1, we can obtain $\underset{0}{=} = \underset{a}{=}$ (in $T(H)$). By $R_a \overset{*}{\underset{2}{\Longrightarrow}} R_b$ and the definition of rule (2), $\underset{a}{=} = \underset{b}{=}$ is trivial. Now, consider $R_b \overset{*}{\underset{3}{\Longrightarrow}} R_n$. By $\underset{a}{=} = \underset{b}{=}$ and $\underset{n}{=} \subseteq \underset{b}{=}$, we can prove $\underset{n}{=} \subseteq \underset{a}{=}$. It has been shown that $CR(R_a)$ and $T(G) \subseteq NF_a$ hold. $\forall M \in T(H)\ \exists N \in T(G)[M=N]$ has been assumed. Hence, by using Theorem 3.1 for R_a and R_n, it can be proved that $\underset{a}{=} = \underset{n}{=}$ (in $T(H)$). Therefore it follows that $\underset{0}{=} = \underset{n}{=}$ (in $T(H)$). □

6. Applications to Program Transformations

By using Theorem 6.1, we will show the correctness of program transformations discussed in [2][12]. Note that the transformation R==>R' can be used in the reverse direction to obtain R from R'.

Example 6.1.(List Reverse) Let H={append, cons, rev, nil} and G={cons, nil}. Note that T(G) can be regarded as the set of lists. Then the append function is defined by;

(1) append(nil,y) ▷ y,

(2) append(cons(x,y),z) ▷ cons(x,append(y,z)).

The reverse function is given by the following rules:

(3) rev(nil) $\triangleright$ nil,

(4) rev(cons(x,y)) $\triangleright$ append(rev(y),cons(x,nil)).

Let us define R_1 by $\triangleright_1 = \{(1),(2),(3),(4)\}$. We will transform R_1 to an improved version R_6 which equals R_1 in the restricted domain T(H). We first add two rules(5),(6) to R_1:

(5) append(append(x,y),z) $\triangleright$ append(x,append(y,z)),

(6) append(x,nil) $\triangleright$ x.

Let us define R_2 by $\triangleright_2 = \triangleright_1 \cup \{(5),(6)\}$. Note that $R_2 \overset{*}{=\!=_3}> R_1$, i.e., $=\!=_1 \subseteq =\!=_2$. By structural induction on nesting levels of function symbols occurring in a term, it can be proved that $\forall M \in T(H)\ \exists N \in T(G)[M =\!=_1 N]$. $T(G) \subseteq NF_2$ is obvious from the definition of R_2. Since R_2 is strongly normalizing, by using Condition 4.1, it can be shown that $CR(R_2)$. Hence $=\!=_1 = =\!=_2$ (in T(H)) holds by Theorem 5.1.

Now, let us transform R_2 to R_6 by using the transformation rules: definition, addition, and elimination. By using definition, we introduce a new function f,

(7) f(x,y) $\triangleright$ append(rev(x),y).

Define R_3 by the union of $\triangleright_2$ and rule(7). Then,

$f(nil,y) =\!=_3 y$,

and,

f(cons(x,y),z)
$=\!=_3$ append(append(rev(y),cons(x,nil)),z)
$=\!=_3$ append(rev(y),append(cons(x,nil),z))
$=\!=_3$ f(y,append(cons(x,nil),z))
$=\!=_3$ f(y,cons(x,z)).

By using addition, we obtain R_4 which is defined by the union of $\triangleright_3$ and the following:

(8) f(nil,y) $\triangleright$ y,

(9) f(cons(x,y),z) $\triangleright$ f(y,cons(x,z)).

Then, $rev(cons(x,y)) =\!=_4 f(y,cons(x,nil))$ holds. Hence we obtain R_5 from R_4, by addition:

(10) rev(cons(x,y)) $\triangleright$ f(y,cons(x,nil)).

Finally, by using elimination, remove unnecessary rules from R_5. Thus, we obtain R_6 defined by the union of {(1),(2)} and the rules:

(3) rev(nil) $\triangleright$ nil,

(10) rev(cons(x,y)) $\triangleright$ f(y,cons(x,nil)),

(8) f(nil,y) $\triangleright$ y,

(9) f(cons(x,y),z) $\triangleright$ f(y,cons(x,z)).

By structural induction, it can be proved that $\forall M \in T(H)\ \exists N \in T(G)[M =\!=_6 N]$. Thus, $=\!=_2 = =\!=_6$ (in T(H)) is obtained by Theorem 5.1. Therefore, $=\!=_1 = =\!=_6$ (in T(H)). □

Example 6.2.(List Reverse-Append) Let the set of function symbols G

and the rewriting rule(1),...,(6) be the same as in Example 6.1. Let $H=G \cup \{append, rev, h\}$, where h is defined by the following rule:

(7) $h(x,y) \triangleright append(rev(x),y)$.

Let us define R_1 by $\triangleright_1 = \{(1),(2),(3),(4),(7)\}$ and R_2 by $\triangleright_2 = \triangleright_1 \cup \{(5),(6)\}$. Then, since $R_2 \overset{*}{\underset{3}{=}}> R_1$, $\underset{1}{=} = \underset{2}{=}$ (in T(H)) can be proved in the same way as in Example 6.1. Here we obtain

$rev(x) \underset{2}{=} h(x,nil)$,

$h(nil,y) \underset{2}{=} y$,

and

$h(cons(x,y),z)$

$\underset{2}{=} append(rev(cons(x,y),z)$

$\underset{2}{=} append(append(rev(y),cons(x,nil)),z)$

$\underset{2}{=} append(rev(y),append(cons(x,nil),z))$

$\underset{2}{=} append(rev(y),cons(x,z))$

$\underset{2}{=} h(y,cons(x,z))$.

Hence the following three rules can be added to R_2 by using addition:

(8) $rev(x) \triangleright h(x,nil)$,

(9) $h(nil,y) \triangleright y$,

(10) $h(cons(x,y),z) \triangleright h(y,cons(x,z))$.

Finally, using elimination, we can obtain R_3 which is defined by the union of {(1),(2)} and,

(8) $rev(x) \triangleright h(x,nil)$,

(9) $h(nil,y) \triangleright y$,

(10) $h(cons(x,y),z) \triangleright h(y,cons(x,z))$.

By structural induction, it is possible to obtain $\forall M \in T(H)\ \exists N \in T(G)[M \underset{3}{=} N]$. Thus, by Theorem 5.1, it can be proved that $\underset{2}{=} = \underset{3}{=}$ (in T(H)). Therefore $\underset{1}{=} = \underset{3}{=}$ (in T(H)). □

7. Conclusion

In this paper we have proposed the concept of the equivalence in a restricted domain for reduction systems. The key point of this concept is that the equivalence in the restricted domain can be tested easily by using the Church-Rosser property of reduction systems. We have shown that the concept can be effectively applied to test the equivalence of term rewriting systems and to prove the correctness of program transformations. We believe firmly that these methods provide us with systematic means of proving the equivalence which arises in various formal systems: program transformation, program verification, semantics of abstract data types, and automated theorem proving.

Acknowledgments

The author is grateful to Hirofumi Katsuno and other members of the First Research Section for their suggestions. The author also wishes to thank Taisuke Sato for his comments.

References

[1] Barendregt,H.P.:" The lambda calculus, its syntax and semantics", North-Holland (1981).

[2] Burstall,R.M. and Darlington,J.:" A transformation system for developing recursive programs", J.ACM, Vol.24 (1977), pp.44-67.

[3] Goguen,J.A.:" How to prove algebraic inductive hypotheses without induction, with applications to the correctness of data type implementation", Proc. 5th Conf. Automated deduction, Les Arcs (1980).

[4] Huet,G.:" Confluent reductions: abstract properties and applications to term rewriting systems", J.ACM, Vol.27 (1980), pp.797-821.

[5] Huet,G. and Oppen,D.C.:" Equations and rewrite rules: a survey", Formal languages: perspectives and open problems, Ed.Book,R., Academic Press (1980), pp.349-393.

[6] Huet,G. and Hullot,J.M.:" Proofs by induction in equational theories with constructors", J. Comput. and Syst.Sci., Vol.25 (1982), pp.239-266.

[7] Klop,J.W.:" Combinatory reduction systems", Dissertation, Univ. of Utrecht (1980).

[8] Knuth,D.E. and Bendix,P.G.:" Simple word problems in universal algebras", Computational problems in abstract algebra, Ed.Leech,J., Pergamon Press (1970), pp.263-297.

[9] Musser,D.R.:" On proving inductive properties of abstract data types", Proc. 7th ACM Sympo. Principles of programming languages (1980), pp.154-162.

[10] O'Donnell,M.:" Computing in systems described by equations", Lecture Notes in Comput. Sci. Vol.58, Springer-Verlag (1977).

[11] Rosen,B.K.:" Tree-manipulating systems and Church-Rosser theorems", J.ACM, Vol 20 (1973), pp.160-187.

[12] Scherlis,W.L.:" Expression procedures and program derivation", Ph.D.thesis, Stanford Computer Science Report STAN-CS-80-818 (1980).

[13] Toyama,Y.:" On commutativity of term rewriting systems ", Trans. IECE Japan, J66-D, 12, pp.1370-1375 (1983).