On Equivalence Transformations for Term Rewriting Systems

Yoshihito TOYAMA

Musashino Electrical Communication Laboratory, N.T.T.
Midori-cho, Musashino-shi, 180 Japan

Abstract

This paper proposes some simple methods, based on the Church-Rosser property, for testing the equivalence in a restricted domain of two reduction systems. Using the Church-Rosser property, sufficient conditions for the equivalence of abstract reduction systems are proved. These conditions can be effectively applied to test the equivalence in a restricted domain of term rewriting systems. In addition, equivalence transformation rules for term rewriting systems are proposed.

1. Introduction

The concept of the equivalence in a restricted domain of two term rewriting systems is presented here. The equivalence in a restricted domain means that the equational relation (or the transitive reflexive closure) generated by a reduction relation of one system is equal in a restricted domain to that of another system.

This concept 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, and equivalence transformation rules in a restricted domain for term rewriting systems are proposed.

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. $R=\langle A,\rightarrow\rangle$ is strongly normalizing (denoted by SN(R) or SN($\rightarrow$)) iff every reduction in R terminates, i.e., there is no infinite sequence $x_0\rightarrow x_1\rightarrow x_2\rightarrow\ldots$. R is weakly normalizing (denoted by WN(R) or WN($\rightarrow$)) iff any $x\in A$ has a normal form.

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 $=_i$ and $\xrightarrow{*}_i$ 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 $\forall x,y\in B[x=_1 y \Leftrightarrow x=_2 y]$, and say R_1 and R_2 are equivalent in the restricted domain B for the equivalence relation. We write $\xrightarrow{*}_1 = \xrightarrow{*}_2$ (in $B\times C$) for $\forall x\in B\,\forall y\in C[x\xrightarrow{*}_1 y \Leftrightarrow x\xrightarrow{*}_2 y]$, and say R_1 and R_2 are equivalent in the restricted domain $B\times C$ for the transitive reflexive closure.

We first show sufficient conditions for $= = \underset{2}{=}$(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 $\forall x,y \in B[x \underset{1}{=} y \Longleftrightarrow x \underset{2}{=} y]$. $\Longrightarrow$ is trivial from condition(1), hence we show $\Longleftarrow$. 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., $\forall x,y \in C[x \underset{1}{=} y \Longleftrightarrow x \underset{2}{=} y]$, from the above conditions. $\Longrightarrow$ is trivial from condition(1), hence we prove $\Longleftarrow$. By using property 2.1(2) and condition(2), $x \underset{2}{=} y \Longrightarrow x \equiv y$ for any $x,y \in C$. Therefore $x \underset{1}{=} y$. □

Corollary 3.1. Assume the conditions:

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

(2) $CR(R_2)$ and $NF_1 = NF_2$,

(3) $WN(R_1)$.

Then $\underset{1}{=} = \underset{2}{=}$ is obtained.

Proof. Set B=A and $NF_1 = NF_2 = C$ in Theorem 3.1. □

Next we will consider sufficient conditions for the equivalence in $B \times C$ for the transitive reflexive closure, i.e., $\overset{*}{\underset{1}{\rightarrow}} = \overset{*}{\underset{2}{\rightarrow}}$ (in $B \times C$).

Theorem 3.2. Assume the following conditions:

(1) $\xrightarrow[1]{*} \subseteq \xrightarrow[2]{*}$,

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

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

Then $\xrightarrow[1]{*} = \xrightarrow[2]{*}$ (in $B \times C$).

Proof. It is sufficient to show that for any $x \in B, y \in C$, $x \xrightarrow[2]{*} y \Rightarrow x \xrightarrow[1]{*} y$. Let $x \xrightarrow[2]{*} y$. Then, by condition(3), there is some $z \in C$ such that $x \xrightarrow[1]{*} z$. By condition(1), $x \xrightarrow[2]{*} z$, hence, $y \equiv z$ is obtained from property 2.1(2) and condition(2). Therefore $x \xrightarrow[1]{*} y$. □

Corollary 3.2. Assume the conditions:

(1) $\xrightarrow[1]{*} \subseteq \xrightarrow[2]{*}$,

(2) $CR(R_2)$ and $NF_1 = NF_2$,

(3) $WN(R_1)$.

Then $\xrightarrow[1]{*} = \xrightarrow[2]{*}$.

Proof. This is obvious from Theorem 3.2. □

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 $F \cap V = \phi$. Let N be the set of natural numbers. An arity function ρ is a mapping from F to N, and if $\rho(f)=n$ then f is called an n-ary function symbol. In particular, a 0-ary function symbol is called a constant.

The set $T(F \cup V)$ of terms on a function symbol set F and a variable symbol set V is inductively defined as follows:

(1) $x \in T(F \cup V)$ if $x \in V$,

(2) $f \in T(F \cup V)$ if $f \in F$ and $\rho(f)=0$,

(3) $f(M_1,\ldots M_n) \in T(F \cup V)$ if $f \in F$, $\rho(f)=n>0$, and $M_1,\ldots,M_n \in T(F \cup V)$.

We may write MfN, i.e., infix notation, instead of f(M,N). 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.

A substitution θ is a mapping from a term set T to T such that

(1) $\theta(f)\equiv f$ if $f\in F$ and $\rho(f)=0$,

(2) $\theta(f(M_1,\ldots,M_n))\equiv f(\theta(M_1),\ldots,\theta(M_n))$
if $f(M_1,\ldots,M_n)\in T$.

Thus, for term M, $\theta(M)$ is determined by its values on the variable symbols occurring in M. Following common usage, we write this as $M\theta$ instead of $\theta(M)$.

Consider an extra constant $\Box$ called a hole and the set $T(F\cup V\cup\{\Box\})$. Then $C\in T(F\cup V\cup\{\Box\})$ is called a context on F. We use the notation C[,...,] for the context containing n holes ($n\geq 0$), and if $N_1,\ldots,N_n\in T(F\cup V)$ then $C[N_1,\ldots,N_n]$ denotes the result of placing $N_1,\ldots,N_n$ in the holes of C[,...,] from left to right. In particular, C[ ] denotes a context containing precisely one hole.

N is called a subterm of $M\equiv C[N]$. Let N be a subterm occurrence of M, then, write $N\subseteq M$, and if $N\neq M$ then write $N\subset M$.

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

$S \notin V$ 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 $M\theta \rightarrow N\theta$, 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 $M \triangleright_1 N$ and $P \triangleright_2 Q$ 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 $\triangleright_1$ and $R_2 = \langle T, \xrightarrow{}_2 \rangle$ with $\triangleright_2$. Then their union $R_1 \cup R_2$ is defined by $R_1 \cup R_2 = \langle T, \rightarrow \rangle$ with $\triangleright = \triangleright_1 \cup \triangleright_2$. 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 in Restricted Domain

In this section, equivalence for term rewriting systems is

discussed. The basic results in Section 3 are effectively applied to test the equivalence in a restricted domain of two systems.

First, useful lemmas are given for showing condition(3) in Theorem 3.1 and Theorem 3.2 in term rewriting systems. Let R be a term rewriting system on $T(F \cup V)$, and $G \subseteq H \subseteq F$.

Lemma 5.1. Let every term of the form $M \equiv f(M_1,\ldots,M_n)$, with $f \in H-G$ and $M_1,\ldots,M_n$ in $T(G)$, have some term N in $T(G)$ such that $M=N$. Then $\forall M \in T(H), \exists N \in T(G)[M=N]$.

Proof. By structural induction on nesting levels of function symbols in H-G occurring in a term, it is easy to show that for any term M in T(H), there is some term N in T(G) such that $M=N$. □

Lemma 5.2. Let every term of the form $M \equiv f(M_1,\ldots,M_n)$, with $f \in H-G$ and $M_1,\ldots,M_n$ in $T(G)$, have some term N in $T(G)$ such that $M \xrightarrow{*} N$. Then $\forall M \in T(H), \exists N \in T(G)[M \xrightarrow{*} N]$.

Proof. The Lemma can be proved in the same way as for Lemma 5.1. □

Here, examples of the equivalence in a restricted domain of $R_1=\langle T(F), \xrightarrow[1]{}\rangle$ with $\rhd_1$ and $R_2=\langle T(F), \xrightarrow[2]{}\rangle$ with $\rhd_2$ will be shown.

Example 5.1. Let $F=\{+,S,0\}$ be a set of function symbols, where $\rho(+)=2$, $\rho(S)=1$, $\rho(0)=0$. Consider the term rewriting systems R_1 and R_2 having the following rewriting rules:

R_1: $x+0 \rhd x$,
$x+S(y) \rhd S(x+y)$,

and

R_2: $x+0 \rhd x$,
$0+x \rhd x$,
$x+S(y) \rhd S(x+y)$.

We shall prove that $=_1 = =_2$ (in T(F)) by using Theorem 3.1. It must be shown that R_1 and R_2 hold conditions(1), (2), (3) in Theorem 3.1. Since $\rhd_1 \subseteq \rhd_2$, condition(1), i.e., $=_1 \subseteq =_2$, is obvious. By using $SN(R_2)$ and condition 4.1, $CR(R_2)$ is obtained. Let $G=\{S,0\}$, then $T(G) \subseteq NF_2$,

thus condition(2) holds. Finally, we can prove condition(3), i.e., $\forall M \in T(F), \exists N \in T(G)[M \underset{1}{=} N]$, by using Lemma 5.1. Therefore $\underset{1}{=} = \underset{2}{=}$ (in $T(F)$).

It is also possible to prove $\xrightarrow[1]{*} = \xrightarrow[2]{*}$ (in $T(F) \times T(G)$) by using Theorem 3.2. Hence we may say that R_1 equals R_2 in $T(F)$ for the equivalence relation and in $T(F) \times T(G)$ for the transitive reflexive closure. However, the number of reduction steps required to obtain a normal form can be reduced by transforming R_1 to R_2 for the following computation:

R_1: $0+S(S(S(0))) \underset{1}{\rightarrow} S(0+S(S(0))) \underset{1}{\rightarrow} S(S(0+S(0)))$
$\underset{1}{\rightarrow} S(S(S(0+0))) \underset{1}{\rightarrow} S(S(S(0)))$,

R_2: $0+S(S(S(0))) \underset{2}{\rightarrow} S(S(S(0)))$. □

Let us examine another example of the equivalence in a restricted domain such that R_2 reduces the number of reduction steps required to obtain a normal form.

Example 5.2. Let $F=\{h,d,S,0\}$ be a function symbol set, where $\rho(h)=\rho(d)=\rho(S)=1$, $\rho(0)=0$. Consider the following R_1 and R_2:

R_1: $h(0) \triangleright 0$,
$h(S(0)) \triangleright 0$,
$h(S(S(x))) \triangleright S(h(x))$,
$d(0) \triangleright 0$,
$d(S(x)) \triangleright S(S(d(x)))$,

and

R_2: $\underset{2}{\triangleright} = \underset{1}{\triangleright} \cup \{h(d(x)) \triangleright x\}$,

where h and d mean the 'half' function $h(n)=\lfloor n/2 \rfloor$ (i.e., the greatest integer less than or equal to n/2), and the 'double' function $d(n)=2*n$. Let $G=\{S,0\}$. Then, by using Theorem 3.1 and 3.2 in the same way as in Example 5.1, we can obtain

$\underset{1}{=} = \underset{2}{=}$ (in $T(F)$),
$\xrightarrow[1]{*} = \xrightarrow[2]{*}$ (in $T(F) \times T(G)$).

R_2 reduces the number of reduction steps more than R_1, since n can be obtained from h(d(n)) in one step by using the rule $h(d(x)) \triangleright x$ in R_2. □

Looking at these examples from another viewpoint, the equivalence in the restricted domain T(F) can be used to prove an equation

$$\forall P_1,..,P_n \in T(F)\ [E_1(P_1,..,P_n) \underset{1}{=} E_2(P_1,..,P_n)]$$

in R_1. Let us assume that

$$E_1(P_1,..,P_n), E_2(P_1,..,P_n) \in T(F)$$

for any $P_1,..,P_n \in T(F)$. By regarding the equation

$$E_1(x_1,..,x_n) = E_2(x_1,..,x_n),$$

that must be proved, as the rewriting rule and adding it to R_1, we obtain R_2 with the set

$$\triangleright_2 = \triangleright_1 \cup \{E_1(x_1,..,x_n) \triangleright E_2(x_1,..,x_n)\}$$

of rewriting rules. By $E_1(x_1,..,x_n) \triangleright E_2(x_1,..,x_n) \in R_2$,

$$\forall P_1,..,P_n \in T(F)\ [E_1(P_1,..,P_n) \underset{2}{=} E_2(P_1,..,P_n)]$$

is trivial. Hence, if we can prove $\underset{1}{=} = \underset{2}{=}$ (in T(F)) by using Theorem 3.1, then

$$\forall P_1,..,P_n \in T(F)\ [E_1(P_1,..,P_n) \underset{1}{=} E_2(P_1,..,P_n)]$$

can be obtained.

For instance, let R_1 be the same as in Example 5.2 and let us prove the equation $\forall P \in T(F)[h(d(P)) \underset{1}{=} P]$ by the above method. We obtain R_2 in Example 5.2 by adding $h(d(x)) \triangleright x$ to R_1. $\underset{1}{=} = \underset{2}{=}$ (in T(F)) has been shown in Example 5.2. Therefore, it can be proved that $\forall P \in T(F)[h(d(P)) \underset{1}{=} P]$.

This idea for proving an equation has been proposed by Musser [9], Goguen [3], Huet and Hullot [6], in studies of the validity of equations in abstract data types. Huet and Hullot showed that by using the above method in a simple extension of the Knuth-Bendix completion algorithm [8], an equation whose proof usually requires induction on some data types can be proved without the direct use of induction. Their method of proving the equation has many restrictions, however. In particular, the requirement of the strongly normalizing property of R_1 and R_2 restricts its application, since most term rewriting systems obtained from recursive definitions, such as recursive programs, do not satisfy these requirements. On the other hand, the basic results proposed

in Section 3 do not require the strongly normalizing property, hence, this difficulty can be overcome by using these results.

We next show an example in which R_2 does not have the strongly normalizing property.

Example 5.3. Let F={if, eq, -, d, S, true, false, 0} be the set of function symbols, where ρ(if)=3, ρ(eq)=ρ(-)=2, ρ(d)=ρ(S)=1, and ρ(true)=ρ(false)=ρ(0)=0. The following term rewriting systems R_1 and R_2 are considered for computing the 'double' function d:

R_1: d(0) $\triangleright$ 0,
d(S(x)) $\triangleright$ S(S(d(x))).

R_2: d(x) $\triangleright$ if(eq(x,0),0,S(S(d(x-S(0))))),
if(true,x,y) $\triangleright$ x,
if(false,x,y) $\triangleright$ y,
eq(0,0) $\triangleright$ true,
eq(S(x),0) $\triangleright$ false,
x-0 $\triangleright$ x,
S(x)-S(y) $\triangleright$ x-y.

The term rewriting system R_2 does not have the strongly normalizing property, since the first rewriting rule in R_2 can be applied infinitely to function symbol d.

Let H={d,S,0} and G={S,0}. It will be shown that the function d of R_1 equals that of R_2 in the restricted domain T(H), that is, $=_1 = =_2$ (in T(H)). For this purpose, Theorem 3.1 is used. We must show conditions(1), (2), (3) in Theorem 3.1. Since $d(0) =_2 0$ and $d(S(x)) =_2 S(S(d(x)))$, condition(1), i.e., $=_1 \subseteq =_2$, is obtained. It is obvious that R_2 is linear and non-overlapping. Hence, by using condition 4.2, R_2 has the Church-Rosser property. Since some function symbol in F-G appears in the left hand side of any rewriting rule in R_2, it is trivial that $T(G) \subseteq NF_2$. Thus, condition(2) holds. By using Lemma 5.1, condition(3) is obtained, i.e., $\forall M \in T(H), \exists N \in T(G)[M =_1 N]$. Therefore, $=_1 = =_2$ (in T(H)) holds. □

6. 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. Moreover, formal proof of the correctness of the transformation was discussed in [12].

This problem can be seen as one of equivalence transformations for term rewriting systems. In this section, an attempt is made to give a formal proof, based on the equivalence in the restricted domain, for the correctness of transformation rules.

Let $R=\langle T(F\cup V),\rightarrow\rangle$ 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 $R_n=\langle T(F\cup V),\underset{n}{\rightarrow}\rangle$ with $\underset{n}{\rhd}$ to $R_{n+1}=\langle T(F\cup V),\underset{n+1}{\rightarrow}\rangle$ with $\underset{n+1}{\rhd}$ 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 $Q\in T(F_n\cup V)$. Thus, $\underset{n+1}{\rhd}=\underset{n}{\rhd}\cup\{g(x_1,\ldots,x_k)\rhd Q\}$. 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 $P,Q\in T(F_n\cup V)$. Thus, $\underset{n+1}{\rhd}=\underset{n}{\rhd}\cup\{P\rhd Q\}$. Set $F_{n+1}=F_n$.

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

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

Lemma 6.1. If $R_1 \underset{i}{\Rightarrow} R_2 \underset{j}{\Rightarrow} R_3$ $(i>j)$, then there is some R_2' such that $R_1 \underset{j}{\Rightarrow} R_2' \underset{i}{\Rightarrow} R_3$.

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

Lemma 6.2. Let $R \overset{*}{\underset{1}{\Rightarrow}} R'$. Then there exists a transformation sequence from R to R' such that $R \overset{*}{\underset{1}{\Rightarrow}} R_a \overset{*}{\underset{2}{\Rightarrow}} R_b \overset{*}{\underset{3}{\Rightarrow}} R'$.

Proof. By using Lemma 6.1 repeatedly, we can construct a sequence $R \overset{*}{\underset{1}{\Rightarrow}} R_a \overset{*}{\underset{2}{\Rightarrow}} R_b \overset{*}{\underset{3}{\Rightarrow}} R'$ from $R \overset{*}{\Rightarrow} R'$. □

Theorem 6.1. Let $R_0 \overset{*}{\Rightarrow} R_n$, 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 6.2, we may assume that $R_0 \overset{*}{\underset{1}{\Rightarrow}} R_a \overset{*}{\underset{2}{\Rightarrow}} R_b \overset{*}{\underset{3}{\Rightarrow}} R_n$. To prove the theorem we will show that $\underset{0}{=} = \underset{a}{=}$ (in $T(H)$) and $\underset{a}{=} = \underset{n}{=}$ (in $T(H)$).

Consider $R_0 \overset{*}{\underset{1}{\Rightarrow}} R_a$. 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 $R_0 \overset{*}{\underset{1}{\Rightarrow}} R_a$. 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 $R_a \overset{*}{\underset{2}{\Rightarrow}} R_b$ and the definition of rule (2), $\underset{a}{=} = \underset{b}{=}$ is trivial.

Now, consider $R_b \overset{*}{\underset{3}{\Rightarrow}} R_n$. 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. $\forall M \in T(H)\ \exists N \in T(G)[M \underset{n}{=} N]$ 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 $T(H)$). □

Theorem 6.2. Let $R_0 \overset{*}{\Rightarrow} R_n$, 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:

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

Then $\overset{*}{\underset{0}{\rightarrow}} = \overset{*}{\underset{n}{\rightarrow}}$ (in $T(H) \times T(G)$).

Proof. By the assumption, it holds that $\forall M \in T(H)\ \exists N \in T(G)[M \underset{i}{=} N]\ (i=0,n)$. Hence, by using Theorem 6.1, we obtain $\underset{0}{=} = \underset{n}{=}$ (in $T(H)$). Now, we will show that $\forall M \in T(H)\ \forall N \in T(G)[M \overset{*}{\underset{0}{\rightarrow}} N \Leftrightarrow M \overset{*}{\underset{n}{\rightarrow}} N]$. $\Rightarrow$: Let $M \overset{*}{\underset{0}{\rightarrow}} N$ and $M \in T(H), N \in T(G)$. Then, by the assumption in the theorem, there is a term $P \in T(G)$ such that $M \overset{*}{\underset{n}{\rightarrow}} P$. By $\underset{0}{=} = \underset{n}{=}$ (in $T(H)$), $M \underset{0}{=} N$, and $M \underset{n}{=} P$, we can obtain $N \underset{?}{=} P$. Since $N, P \in NF_0$ and $CR(R_0)$, $N \equiv P$ holds. Therefore $M \overset{*}{\underset{n}{\rightarrow}} N$. $\Leftarrow$: It can be proved in the same way. □

By using the above theorems, we will show the correctness of the equivalence transformation for the examples discussed in [2][12]. Note that the transformation $R \overset{*}{\Rightarrow} R'$ can be used in the reverse direction to obtain R from R'. We write $R' \overset{*}{\Leftarrow} R$ if $R \overset{*}{\Rightarrow} R'$ is used to obtain R from R'. Hence, if $R_1 \overset{*}{\Leftarrow} R_2$, $R_2 \overset{*}{\Rightarrow} R_3$, $R_3 \overset{*}{\Leftarrow} R_4, \ldots, R_{n-1} \overset{*}{\Rightarrow} R_n$ each are the equivalence transformation on the restricted domain $T(H)$, i.e., $\underset{i}{=} = \underset{i+1}{=}$ (in $T(H)$) for $1 \leq i \leq n-1$, then we can obtain R_n from R_1 by this sequence.

Example 6.1.(List Reverse) Let H={append, cons, rev, nil} and G={cons, nil}, where ρ(append)=ρ(cons)=2 and ρ(rev)=1, ρ(nil)=0. Note that T(G) can be regarded as the set of lists. Then the append function is defined by;

(1) append(nil,y) $\triangleright$ y,
(2) append(cons(x,y),z) $\triangleright$ 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 $\underset{2}{\triangleright} = \underset{1}{\triangleright} \cup \{(5),(6)\}$. Note that $R_1 \overset{*}{\underset{3}{\Leftarrow}} R_2$, i.e., $\underset{1}{=} \subseteq \underset{2}{=}$. By using Lemma 5.1, it can be proved that $\forall M \in T(H)\ \exists N \in T(G)[M \underset{1}{=} N]$. $T(G) \subseteq NF_2$ 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 $\underset{1}{=} = \underset{2}{=}$ (in T(H)) holds by Theorem 6.1.

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 $\underset{2}{\triangleright}$ and rule(7). Then,

f(nil,y) $\underset{3}{=}$ y,

and,

f(cons(x,y),z)
$\underset{3}{=}$ append(append(rev(y),cons(x,nil)),z)
$\underset{3}{=}$ append(rev(y),append(cons(x,nil),z))
$\underset{3}{=}$ f(y,append(cons(x,nil),z))
$\underset{3}{=}$ f(y,cons(x,z)).

By using addition, we obtain R_4 which is defined by the union of $\underset{3}{\triangleright}$ 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)) $\underset{4}{=}$ 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 using Lemma 5.1, it can be proved that $\forall M\in T(H)\ \exists N\in T(G)[M \underset{6}{=} N]$. Thus, $\underset{1}{=} = \underset{6}{=}$ (in T(H)) is obtained by Theorem 6.1. Therefore, $\underset{1}{=} = \underset{2}{=}$ (in T(H)).

Note that it is also possible to prove $\xrightarrow[1]{*} = \xrightarrow[6]{*}$ (in $T(H)\times T(G)$) by using Theorem 6.2. □

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 $R_1 \underset{3}{\overset{*}{\Leftarrow}} R_2$, $\underset{1}{=} = \underset{2}{=}$ (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 using Lemma 5.1, it is possible to obtain $\forall M \in T(H) \exists N \in T(G)[M \underset{3}{=} N]$. Thus, by Theorem 6.1, it can be proved that $\underset{2}{=} = \underset{3}{=}$ (in $T(H)$). Therefore $\underset{1}{=} = \underset{3}{=}$ (in $T(H)$).

By using Theorem 6.2, we can also obtain $\xrightarrow[1]{*} = \xrightarrow[3]{*}$ (in $T(H) \times T(G)$). □

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 in Sections 5 and 6 that the concept can be effectively applied to test the equivalence of term rewriting systems and to prove the correctness of equivalence transformation rules for these systems. 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).