数理解析研究所講究録
511巻 1984年 74-96

74

Symposium on Software Science
and Engineering
in Kyoto Sep.1983

On Equivalence Transformations for Term Rewriting Systems

Yoshihito TOYAMA
外山 芳人

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

Abstract

This paper proposes some simple methods, based on the Church-Rosser property, for testing equivalence on a limited domain for two reduction systems. Using the Church-Rosser property, some equivalence conditions of abstract reduction systems are first proved. It is then shown that these conditions can be effectively applied to test equivalence of term rewriting systems. Finally, the correctness of transformation rules for term rewriting systems is discussed.

1. Introduction

The equivalence of two term rewriting systems is considered. Equivalence treated in this paper means that the equational relation (or the reduction relation) generated by one system is equal to that in another system on some limited domain. This equivalence concept on a limited domain plays an important role in transforming recursive programs [2][12] and proving an equation on abstract data types [3][5][6][9]. For example, consider a recursive program computing the factorial function on natural numbers;

F(x)=IF equal(x,0) THEN 1 ELSE x*F(x-1).

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

F(0)=1,

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

Regarding equations as rewriting rules over terms, we can obtain two term rewriting systems [4][5] from the above two definitions. Then the second term rewriting system is weaker than the first, since the reduction from "F(M)" to "IF equal(M,0) THEN 1 ELSE M*F(M-1)" for any term M in the first system can not be obtained in the second system by only the rewriting rules, although, they give the same function for natural numbers. Thus, the equivalence for recursive programs must be regarded as the equivalence on a limited domain such as natural numbers for term rewriting systems.

The equivalence concept on a limited domain for term

rewriting systems first appeared implicitly in automated theorem provers for abstract data types discussed by Goguen[3], Huet and Hullot[6], and Musser[9]. In these methods, an equation M=N whose proof usually requires induction on some data types is proved by cleverly using the Church-Rosser property, without explicit induction.

We shall here formalize and extend this idea, introducing the concept of equivalence on a limited domain for abstract reduction systems. Sufficient conditions for the equivalence on a limited domain are given. It is shown how one can formally validate the transformations for term rewriting systems by using these conditions. Finally, we discuss the problems related to rules for transforming programs described by Burstall and Darlington [2], and Scherlis [12].

2. Reduction System

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

2.1. Definitions

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. The identity of elements of A (or syntactical equality) is denoted by $\equiv$. $\overset{*}{\rightarrow}$ is the

transitive reflexive closure, $\xrightarrow{=}$ is the reflexive closure 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 $NF_{\rightarrow}$ or NF be the set of normal forms. If $x\xrightarrow{*}y\in NF$ then we say x has a normal form y and y is a normal form of x.

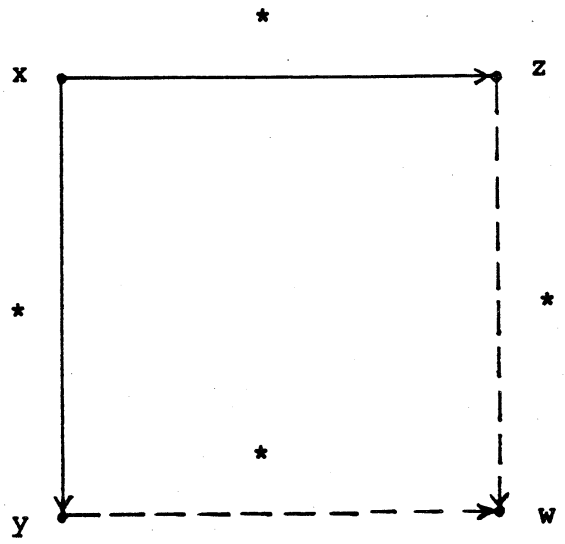
$R=\langle A,\rightarrow\rangle$ is strongly normalizing (denoted by SN(R) or $SN(\rightarrow)$) iff every reduction in R terminates, i.e., there is no infinite sequence $x_0\rightarrow x_1\rightarrow x_2\rightarrow\ldots$.

R is weakly normalizing (denoted by WN(R) or $WN(\rightarrow)$) iff any $x\in A$ has a normal form.

2.2. Church-Rosser

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

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



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

2.3. Property

Let CR(R), 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 $\xrightarrow{*}_i$, $=_i$ and NF_i be the transitive closure, the equivalence relation and the set of normal forms in R_i respectively (i=1,2). Let B, C be any subset of object set A. We write $=_1 = =_2$ (on B) for $\forall x,y\in B[x=_1 y \Leftrightarrow x=_2 y]$, and it means that two equivalence relations $=_1$ and $=_2$ are the same on the limited object set B. We first consider sufficient conditions for $=_1 = =_2$ (on B).

3.1. Lemma

Let R_1, R_2 have the following conditions:

(1) $=_1 \subset =_2$,

(2) $=_1 = =_2$ (on C),

(3) $\forall x\in B, \exists y\in C[x=_1 y]$.

Then $=_1 = =_2$ (on B).

(proof)

Prove $\forall x,y\in B[x=_1 y \Leftrightarrow x=_2 y]$. $\Rightarrow$ is trivial from condition(1), hence we show $\Leftarrow$. Assume $x=_2 y$ 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 to the following condition.

3.2. Theorem

Let's assume the conditions:

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

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

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

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

(proof)

Show condition(2) of Lemma 3.1, $\forall x,y \in C[x \underset{1}{=} y \iff x \underset{2}{=} y]$, from the above conditions. $\Rightarrow$ is trivial from condition(1). We prove $\Leftarrow$. By using property 2.3(2) and condition(2), $x \underset{2}{=} y \Rightarrow x \equiv y$. Therefore $x \underset{1}{=} y$. □

3.3. Corollary

Assume the conditions:

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

(2) $WN(R_1)$ and $CR(R_2)$,

(3) $NF_1 = NF_2$.

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

(proof)

Let B=A and $NF_1 = NF_2 = C$. By using Theorem 3.2, we can

easily prove the corollary. □

Next the equivalence for reduction relations is considered. We write $\xrightarrow[1]{*} = \xrightarrow[2]{*}$ (from B to C) for $\forall x\in B, \forall y\in C[x\xrightarrow[1]{*}y \iff x\xrightarrow[2]{*}y]$.

3.4. Theorem

Assume the following conditions:

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

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

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

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

(proof)

It is sufficient to show that for any $x\in B, y\in C$, $x\xrightarrow[2]{*}y \Longrightarrow x\xrightarrow[1]{*}y$. 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 2.3(2) and condition(2). Therefore $x\xrightarrow[1]{*}y$. □

3.5. Corollary

Assume the conditions:

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

(2) $WN(R_1)$ and $CR(R_2)$,

(3) $NF_1=NF_2$.

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

(proof)

This is obvious from Theorem 3.4. □

4. Term Rewriting System

Next, we will explain term rewriting systems that are reduction systems having term structure.

4.1. Term

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 $F \cap V = \emptyset$. An arity function ρ is a mapping from F to natural number $\mathbb{N}$, 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 definded 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) $f(M_1,...M_n) \in T(F \cup V)$ if $f \in F$, $\rho(f)=n>0$, and $M_1,...,M_n \in T(F \cup V)$.

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 $T(F \cup V)$ when F and V are clear in the context.

4.2. Substitution

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) $\theta(f(M_1,...,M_n)) \equiv f(\theta(M_1),...,\theta(M_n))$

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 $\theta(M)$.

4.3. Context

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 the 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.

4.4. Subterm

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

4.5. Rewriting rule

A rewriting rule on T is a binary relation $\triangleright$ on T, written as $M_1\triangleright M_r$ for $\langle M_1,M_r\rangle\in\triangleright$, such that if $M_1\triangleright M_r$ then any variable in M_r also occurs in M_1. A $\rightarrow$redex, or redex, is a term $M_1\theta$ where $M_1\triangleright M_r$, and in this case $M_r\theta$ is called a $\rightarrow$contractum, or contractum, of $M_1\theta$. The rewriting rule $\triangleright$ on T defines a reduction relation $\rightarrow$ on T as follows:

$M\rightarrow N$ iff $M\equiv C[M_1\theta]$, $N\equiv C[M_r\theta]$, and $M_1\triangleright M_r$

for some M_1, M_r, C[ ], and θ.

4.6. Term Rewriting System

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 rewriting rule $\triangleright$ on T. If R has $M_1 \triangleright M_r$, then we write $M_1 \triangleright M_r \in R$.

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.

4.7. Critical Pair

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 N share no variables. Assume S ($S \notin V$) is a subterm occurrence in $M \equiv C[S]$ such that S and P are unifiable, with minimal unifier θ. 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 [4][5][8][13].

The critical pair to two term rewriting systems can be defined in the same way. Let $M \triangleright N$ and $P \triangleright Q$ be in R_1 and 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 between them [13].

4.8. Conditions for Church-Rosser

The following sufficient conditions for the

Church-Rosser property are well known [4][5][8]:

(1) Let SN(R). If for any critical pair $\langle P,Q\rangle$ in R, P and Q have the same normal form, then CR(R).

(2) Let R be linear and non-overlapping. Then CR(R).

The next condition is described in [13] by using the commutativity between two term rewriting systems R_1 and R_2.

(3) Consider two linear term rewriting systems R_1, R_2, being non-overlapping between them and $CR(R_1)$, $CR(R_2)$. Let $R=R_1 \cup R_2$, then CR(R).

5. Equivalence Transformation

In this section, equivalence transformations for term rewriting systems are considered. The basic results in section 3 are effectively applied to test the equivalence on some limited domain for two systems.

First, useful lemmas are given for showing condition(3) in Theorem 3.2 and Theorem 3.4 on term rewriting systems. Let R be a term rewriting system on $T(F \cup V)$, and $G \subset F$.

5.1. Lemma

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

(proof)

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

5.2. Lemma

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

(proof)

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

5.3. Example

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

R_1: $\begin{cases} x+0 \triangleright x, \\ x+S(y) \triangleright S(x+y), \end{cases}$

and

R_2: $\begin{cases} x+0 \triangleright x, \\ 0+x \triangleright x, \\ x+S(y) \triangleright S(x+y). \end{cases}$

We shall prove that $\underset{1}{=} = \underset{2}{=}$ (on $T(F)$) by using Theorem 3.2. It must be shown that R_1 and R_2 hold condition(1), (2), (3) in Theorem 3.2. Since $R_1 \subset R_2$, condition(1), $\underset{1}{=} \subset \underset{2}{=}$, is obvious. From condition 4.8(1), $CR(R_2)$ is obtained. Let $G=\{S,0\}$, then $T(G) \subset NF_2$, thus condition(2) holds. Finally, we can prove condition(3), $^{\forall} M \in T(F), {}^{\exists} N \in T(G)[M \underset{1}{=} N]$, by Lemma 5.1. Therefore $\underset{1}{=} = \underset{2}{=}$ (on $T(F)$).

It is also possible to prove $\xrightarrow[1]{*} = \xrightarrow[2]{*}$ (from $T(F)$ to $T(G)$) by using Theorem 3.4. Hence we may say that R_2 equals R_1 on $T(F)$ for the equivalence relation and the reduction

relation. R_2 is, however, faster than R_1 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: $0+S(S(S(0))) \underset{2}{\rightarrow} S(S(S(0)))$.

Thus the number of reduction steps to obtain a normal form can be improved by transforming R_1 to R_2.

5.4. Example

We show an other example of the equivalence transformation improving the number of reduction steps to obtain a normal form. 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: $\begin{cases} h(0) \triangleright 0, \\ h(S(0)) \triangleright 0, \\ h(S(S(x))) \triangleright x, \\ d(0) \triangleright 0, \\ d(S(x)) \triangleright S(S(d(x))), \end{cases}$

and

$R_2 = R_1 \cup \{h(d(x)) \triangleright x\}$,

where h and d mean the 'half' function $h(n)=\lfloor n/2 \rfloor$ and the 'double' function $d(n)=2*n$. Let $G=\{S,0\}$. Then, by using Theorem 3.2 and 3.4 in the same way as in Example 5.3, we can obtain,

$\underset{1}{=} = \underset{2}{=}$ (on $T(F)$),

$\underset{1}{\overset{*}{\rightarrow}} = \underset{2}{\overset{*}{\rightarrow}}$ (from $T(F)$ to $T(G)$).

R_2 improves the number of reduction steps, since n can be

obtained from h(d(n)) with one step.

Looking at the examples in new light, the above equivalence transformations can also be used to prove equation P=Q on term rewriting system R_1. Consider the proof of some equation $P \underset{1}{=} Q$ on T(F). First, let $R_2 = R_1 \cup \{P \triangleright Q\}$. Second, Prove $\underset{1}{=} = \underset{2}{=}$ (on T(F)) by using Theorem 3.2. Then, since $P \underset{2}{=} Q$ on T(F) from $P \triangleright Q \in R_2$, $P \underset{1}{=} Q$ on T(F) can be obtained. For instance, we attempt this method for Example 5.4 to prove $\forall N \in T(F)[h(d(N)) \underset{1}{=} N]$. From $h(d(x)) \triangleright x \in R_2$, $\forall N \in T(F)[h(d(N)) \underset{2}{=} N]$. Moreover, $\underset{1}{=} = \underset{2}{=}$ (on T(F)). Therefore, it can be said that $\forall N \in T(F)[h(d(N)) \underset{1}{=} N]$. These ideas were first discussed by Musser [9], Goguen [3], Huet and Hullot [6], in studies of proofs with induction on some data types in equational systems. Huet and Hullot showed that by using a simple extension of the Knuth-Bendix completion algorithm [8], equations can be proved without directly using induction. Their method has many limitations, however. In particular, the requirement of the strongly normalizing property limits its application, since many recursive definitions , such as recursive programs, do not satisfy these requirements. On the other hand, our basic results in Section 3 do not require the strongly normalizing property. We next show an example transforming R_1 which does not have the strongly normalizing property.

5.5. Example

Let F={if, eq, -, d, S, true, false, 0} be a 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 system is considered for computing the 'double' function d:

R_1: $\begin{cases} d(x) \triangleright \text{if}(\text{eq}(x,0),0,S(S(d(x\text{-}S(0))))), \\ \text{if}(\text{true},x,y) \triangleright x, \\ \text{if}(\text{false},x,y) \triangleright y, \\ \text{eq}(0,0) \triangleright \text{true}, \\ \text{eq}(S(x),0) \triangleright \text{false}, \\ x\text{-}0 \triangleright x, \\ S(x)\text{-}S(y) \triangleright x\text{-}y. \end{cases}$

This term rewriting system does not have the strongly normalizing property, since the first rewriting rule can be applied infinitely to function symbol d. By using Condition 4.8(2), The Church-Rosser property for R_1 can be easily shown. Let R_2 have the following rules:

R_2: $\begin{cases} d(0) \triangleright 0, \\ d(S(x)) \triangleright S(S(d(x))). \end{cases}$

It is obvious that R_2 has the strongly normalizing property. Let H={d,S,0} and G={S,0}. It will be shown that the function d of R_2 equals R_1 on natural numbers, that is, $\underset{1}{=} = \underset{2}{=}$ (on T(H)). For this purpose, Theorem 3.2 is used. We show condition(1), (2), (3) in the Theorem 3.2. Since $d(0) \underset{1}{=} 0$ and $d(S(x)) \underset{1}{=} S(S(d(x)))$, condition(1), $\underset{2}{=} \subset \underset{1}{=}$, is obtained. Moreover condition(2), $CR(R_1)$ and $\sim T(G) \subset NF_1$, hold. By using Lemma 5.1, condition(3) is obtained, i.e., $\forall M \in T(H), \exists N \in T(G)[M \underset{2}{=} N]$. Therefore, $\underset{1}{=} = \underset{2}{=}$ (on T(H)) holds.

6. Transformation Rules

In this section we consider the correctness of 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 correctness was considered 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 operational semantics, for the correctness of transformation rules.

6.1. Rules

For program transformations, Scherlis suggested the following rules: Abstraction, Composition, Application, and Elimination. Using these rules, a recursive program, described by a set of equations, can be transformed to an improved one.

This transformation method can be considered from the standpoint of term rewriting systems. Let R_1 be a term rewriting system on $T(F \cup V)$. Then, we use the following three rules for transforming R_1 to new system R_2:

(1) Definition: Add the new rewriting rule $g(x_1,\ldots,x_n) \triangleright Q$ to R_1, where $g \notin F$ is new function symbol, $g(x_1,\ldots,x_n)$ is linear, and $Q \in T(F \cup V)$. Thus,

$$R_2 = R_1 \cup \{g(x_1,\ldots,x_n) \triangleright Q\}.$$

Hence R_2 is on $T(F \cup \{g\} \cup V)$.

(2) Addition: Add the new rule $P \triangleright Q$ to R_1, where $P =_1 Q$. Thus,

$$R_2 = R_1 \cup \{P \triangleright Q\}.$$

(3) Elimination: Remove the rule $P \triangleright Q$ from R_1. Thus,

$$R_2 = R_1 - \{P \triangleright Q\}.$$

The above three rules include the transformation rules suggested by Scherlis. Abstraction is represented by using Definition, Addition, and Elimination. Composition is represented by Addition, and Application by Addition and Elimination. The main difference between his rules and the rules presented here is that his rules preserve strong equivalence of recursive programs, while our rules preserve only the equivalence on some domain of term rewriting systems.

$R_1 \underset{i}{\Rightarrow} R_2$ shows that R_1 is transformed to R_2 by rule(i). $R_1 \Rightarrow R_2$ shows that R_1 is transformed to R_2 by rule(1), (2), or (3). $R_1 \overset{*}{\underset{i}{\Rightarrow}} R_2$ and $R_1 \overset{*}{\Rightarrow} R_2$ are the transitive reflexive closure of these two relations.

6.2. Theorem

Let $R_1 \overset{*}{\Rightarrow} R_2$, where R_1 is a linear system on $T(F \cup V)$ and $CR(R_1)$. Let $G \subset F$ and $T(G) \subset NF_1$. Assume the following property for R_1 and R_2:

$$\forall M \in T(F)\ \exists N \in T(G)\ [M \underset{i}{=} N] \qquad (i=1,2).$$

Then $=_1 = =_2$ (on $T(F)$).

(proof)

By exchanging these rule application order, the above transformation sequence can be replaced by,

$R_1 \overset{*}{\Rightarrow}_1 R_a \overset{*}{\Rightarrow}_2 R_b \overset{*}{\Rightarrow}_3 R_2$.

From Condition 4.8.(3), we easily obtain $CR(R_a)$. Moreover, $T(G) \subset NF_a$. Therefore, by using Theorem 3.2, it can be proved that $\underset{1}{=} = \underset{a}{=}$ (on $T(F)$).

It is trivial that $\underset{a}{=} = \underset{b}{=}$ from the definition of rule(2). Since $\underset{2}{=} \subset \underset{b}{=}$, $\underset{2}{=} \subset \underset{a}{=}$. Hence, using theorem 3.2 again, $\underset{2}{=} = \underset{a}{=}$ (on $T(F)$).

Therefore $\underset{1}{=} = \underset{2}{=}$ (on $T(F)$) holds. □

6.3. Theorem

Let $R_1 \overset{*}{\Rightarrow} R_2$, where R_1 is a linear system on $T(F \cup V)$ and $CR(R_1)$. Let $G \subset F$ and $T(G) \subset NF_1$. Assume the following property for R_1 and R_2:

$\forall M \in T(F)\ \exists N \in T(G)\ [M \overset{*}{\underset{i}{\rightarrow}} N] \quad (i=1,2)$.

Then $\overset{*}{\underset{1}{\rightarrow}} = \overset{*}{\underset{2}{\rightarrow}}$ (from $T(F)$ to $T(G)$).

(proof)

By using Theorem 6.2, we obtain $\underset{1}{=} = \underset{2}{=}$ (on $T(F)$). Let $M \in T(F)$, $N,P \in T(G)$, and $M \overset{*}{\underset{1}{\rightarrow}} N$, $M \overset{*}{\underset{2}{\rightarrow}} P$. Then, by $M \underset{2}{=} P$, $M \underset{1}{=} P$ can be obtained, therefore, $N \underset{1}{=} P$. Since $N,P \in NF_1$ and $CR(R_1)$, $N \equiv P$ holds. From this fact, it can be easily proved that $\forall M \in T(F)\ \forall N \in T(G)\ [M \overset{*}{\underset{1}{\rightarrow}} N \iff M \overset{*}{\underset{2}{\rightarrow}} N]$. □

We will prove the correctness of transformation for the following examples discussed in [2][12].

6.4. Example (List reverse)

Let G={cons, nil}, where ρ(cons)=2 and ρ(nil)=0. Then we can see T(G) as List. The append function on List is proposed 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 R_1={(1),(2),(3),(4)} and F=G $\cup$ {append,rev}. We first show that for M,N,P $\in$ T(F),

append(append(M,N),P) $=_1$ append(M,append(N,P)), and

append(M,nil) $=_1$ M. For this purpose two rules are added to R_1:

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

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

Let $R_2=R_1 \cup \{(5),(6)\}$. Note that $\forall M \in T(F)\ \exists N \in T(G)[M =_1 N]$ and $T(G) \subset NF_2$. Using Condition 4.8.(1), it can be shown that $CR(R_2)$. Hence $=_1 = =_2$ (on T(F)) by Theorem 3.2.

Next we will transform R_2 to an improved version by using the transformation rules. Using Abstraction, we introduce a new function f,

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

Let R_3 be the union of R_2 and the above rule. 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))

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

By using Addition, we obtain R_4 which is the union of R_3 and the following:

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

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

Then, rev(cons(x,y))$\underset{4}{=}$f(y,cons(x,nil)) holds. Hence we obtain R_5 from R_4, by adding:

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

Finally, removing unnecessary rules from R_5, R_6 is obtained which is 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,append(cons(x,nil),z)).

By using Lemma 5.1, it can be proved that $\forall M\in T(F)\ \exists N\in T(G)[M\underset{6}{=}N]$. Therefore, $\underset{1}{=} = \underset{6}{=}$ (on T(F)) is obtained by Theorem 6.2.

Note that it is also possible to prove $\xrightarrow[1]{*} = \xrightarrow[6]{*}$ (from T(F) to T(G)) by using Theorem 6.3.

6.5. Example (List reverse-append)

Let the set of function symbols G and the rewriting rule(1),...,(6) be the same as in Example 6.4. Let F=G$\cup${append,rev,h} , where h is defined by the following rule:

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

Let R_1={(1),(2),(3),(4),(7)} and $R_2=R_1\cup$ {(5),(6)}. Note that $\forall M\in T(F)\ \exists N\in T(G)[M\underset{1}{=}N]$. Then, $\underset{1}{=} = \underset{2}{=}$ (on T(F)) can be

proved in the same way as in Example 6.4. Here we obtain $h(nil,y) \underset{2}{=} y$, 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 two rules can be added by using Addition:

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

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

Since $rev(x) \underset{2}{=} h(x,nil)$, using Addition, we can add rule(10):

(10) $rev(x) \triangleright h(x,nil)$.

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

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

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

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

By using Lemma 5.1, it is possible to obtain $\forall M \in T(F) \; \exists N \in T(G) [M \underset{3}{=} N]$. Therefore $\underset{1}{=} = \underset{3}{=}$ (on T(F)) holds for Theorem 6.2.

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 helpful

comments.

References

[1] Barendrect,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 ", to appear (1983), in Japanese.