How to Prove Equivalence of Term Rewriting Systems without Induction*

Yoshihito TOYAMA
NTT Electrical Communications Laboratories
3-9-11 Midori-cho, Musashino-shi, Tokyo 180, Japan

Abstract

A simple method, based on the Church-Rosser property and reachability, is proposed for proving the equivalence in a restricted domain of two given term rewriting systems without the explicit use of induction; this proof usually requires some kind of induction. The method is applied to proving the inductive theorems and to deriving a new term rewriting system from a given system by using the equivalence transformation rules. The result is a general extension of the *inductionless induction* methods developed by Musser, Goguen, Huet and Hullot.

1 Introduction

We consider how to prove the equivalence in a restricted domain of two term rewriting systems [10, 11] without induction. The equivalence in a restricted domain means that the equational relation (or the transitive reflexive closure) generated by the reduction relation of one system is equal in the restricted domain to that of the other system.

We first explain the concept of the equivalence in a restricted domain through simple examples.

*A preliminary version of this paper was published in the Proceedings of the 8th International Conference on Automated Deduction in July 1986, Lecture Notes in Computer Science 230 (Springer-Verlag, 1986).

$\mathbf{N}$ is the set of natural numbers represented by $0, s(0), s(s(0)), \ldots$. Consider the term rewriting system R_1 computing the addition on the set $\mathbf{N}$:

$$R_1 \quad \begin{cases} x + 0 \triangleright x \\ x + s(y) \triangleright s(x + y). \end{cases}$$

By adding the associative law to R_1, we can obtain another system R_2 computing the same function:

$$R_2 \quad \begin{cases} x + 0 \triangleright x \\ x + s(y) \triangleright s(x + y) \\ x + (y + z) \triangleright (x + y) + z. \end{cases}$$

Then, R_2 can reduce $(M+N)+P$ and $M+(N+P)$ to the same normal form for any terms M, N, P, while R_1 cannot generally do so unless M, N and P are ground terms. The equivalence of R_1 and R_2 must be regarded as the equivalence in the set of ground terms (i.e., in the initial model) of two systems. Thus we can observe the equivalence in a restricted domain of two term rewriting systems R_1 and R_2.

We show another example. Consider the following term rewriting systems $\tilde{R}_1$ and $\tilde{R}_2$ computing the *double* function $d(n) = 2 * n$:

$$\tilde{R}_1 \quad \begin{cases} d(0) \triangleright 0 \\ d(s(x)) \triangleright s(s(d(x))) \end{cases}$$

$$\tilde{R}_2 \quad \begin{cases} d(x) \triangleright \mathit{if}(x, 0, s(s(d(x - s(0))))) \\ \mathit{if}(0, y, z) \triangleright y \\ \mathit{if}(s(x), y, z) \triangleright z \\ x - 0 \triangleright x \\ s(x) - s(y) \triangleright x - y. \end{cases}$$

The term rewriting system $\tilde{R}_1$ has no infinite reduction sequence. On the other hand, $\tilde{R}_2$ has an infinite reduction sequence starting with the term $d(M)$ for any term M, since the first rewriting rule in $\tilde{R}_2$ can be infinitely applied to the function symbol d. Moreover, $\tilde{R}_1$ has no rewriting rules for the functions *if* and $-$. Thus, $\tilde{R}_1$ and $\tilde{R}_2$ generally produce different reduction sequences, although they can reduce the term $d(M)$ to the same result if M can be reduced to a natural number in $\mathbf{N}$. Therefore, the equivalence of $\tilde{R}_1$ and $\tilde{R}_2$ must be regarded as that in the set of ground terms represented by only the function symbols d, s and 0. Thus, we can say that $\tilde{R}_1$ and $\tilde{R}_2$ are equivalent in the restricted domain.

The concept of equivalence in a restricted domain of term rewriting systems frequently appears in computer science: automated theorem proving, semantics of functional programs, program transformation, verification of programs, and specification of abstract data types. However, this equivalence cannot generally be proven by mere equational reasoning; some kind of induction on the domain structure is necessary.

This paper presents a new simple method for proving the equivalence in a restricted domain of two term rewriting systems without the explicit use of induction. Our approach to this problem was inspired by the *inductionless induction* methods developed by Musser [21], Goguen [7], Huet and Hullot [12], and others [6, 13, 15, 17, 20, 23, 24, 27]. We generalize the *inductionless induction* methods within a general framework of abstract reduction systems. Some limitations of the *inductionless induction* methods are removed: in particular, the strongly normalizing restriction. The *sufficient completeness* limitation is replaced with a more general concept of *reachability*.

The key idea behind our method is that the equivalence in a restricted domain can be easily proven by the Church-Rosser property and reachability. We first explain this idea in an abstract framework. Section 2 introduces preliminary concepts of abstract reduction systems. In Section 3, we indicate simple sufficient conditions for the equivalence in a restricted domain of two given abstract reduction systems. Our results are carefully partitioned between abstract properties depending solely on the reduction relation and properties depending on the term structure. In Section 4, we present preliminary concepts for term rewriting systems. In the next section, we offer some examples of how to prove the equivalence for term rewriting systems by using the abstract results. In Section 6, we explain how our method relates to the *inductionless induction* methods developed by Musser [21], Goguen [7], Huet and Hullot [12]. Based on our framework, we demonstrate that the *inductionless induction* methods can work under a very weak assumption. Finally, in Section 7, we propose an equivalence transformation technique for term rewriting systems.

Furthermore, we discuss some examples confirming that the extended *inductionless induction* concept is a useful tool for proving correctness of program transformations proposed by Burstall and Darlington[3].

2 Reduction Systems

We explain notions of reduction systems and give definitions for the following sections, according to Huet[10] and Klop[18]. Since these reduction systems have only an abstract structure, they are called abstract reduction systems in Klop[18].

A reduction system is a structure $R = \langle A, \rightarrow \rangle$ consisting of some object set A and some binary relation $\rightarrow$ on A (i.e., $\rightarrow \subseteq A \times A$), called a reduction relation. A reduction (starting with x_0) in R is a finite or infinite sequence $x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \cdots$. The identity of elements of A (or syntactical equality) is denoted by $\equiv$. $\stackrel{*}{\rightarrow}$ is the transitive 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, or $\rightarrow$ normal form; let NF be the set of normal forms. If $x \stackrel{*}{\rightarrow} y$ 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 R is terminating, iff every reduction in R terminates, i.e., there is no infinite sequence $x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \cdots$. R is weakly normalizing (denoted by $WN(R)$) iff any $x \in A$ has a normal form.

Definition. $R = \langle A, \rightarrow \rangle$ has the Church-Rosser property (denoted by $CR(R)$), or R is confluent, iff $\forall x, y, z \in A[x \stackrel{*}{\rightarrow} y \wedge x \stackrel{*}{\rightarrow} z \Rightarrow \exists w \in A, y \stackrel{*}{\rightarrow} w \wedge z \stackrel{*}{\rightarrow} w]$.

The following proposition is well known [1, 10, 18] .

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

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

(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 $R_1 = \langle A, \underset{1}{\rightarrow}\rangle$ and $R_2 = \langle A, \underset{2}{\rightarrow}\rangle$ be two abstract reduction systems having the same object set A, and let $\overset{*}{\underset{i}{\rightarrow}}$, $\underset{i}{=}$ and NF_i (i=1,2) be the transitive reflexive closure, the equivalence relation and the set of normal forms in R_i respectively. Note that $\xrightarrow{*}$ and $=$ are subsets of $A \times A$; for example, $\underset{1}{=} \subseteq \underset{2}{=}$ means that $\forall x, y \in A[x \underset{1}{=} y \Rightarrow x \underset{2}{=} y]$.

Definition. Let A' and A'' be any nonempty subsets of the object set A. Let $\underset{i}{\sim}$ $(i = 1, 2)$ be any two binary relations on A. Then we give the following definitions:

(1) $\underset{1}{\sim} = \underset{2}{\sim}$ in $A' \times A''$ iff $\forall x \in A' \forall y \in A''[x \underset{1}{\sim} y \iff x \underset{2}{\sim} y]$, and $\underset{1}{\sim} \subseteq \underset{2}{\sim}$ in $A' \times A''$ iff $\forall x \in A' \forall y \in A''[x \underset{1}{\sim} y \Rightarrow x \underset{2}{\sim} y]$. We write these relations as $\underset{1}{\sim} = \underset{2}{\sim}$ in A' and $\underset{1}{\sim} \subseteq \underset{2}{\sim}$ in A' respectively if $A' = A''$.

(2) A'' is reachable from A' under $\underset{1}{\sim}$ iff $\forall x \in A' \exists y \in A''[x \underset{1}{\sim} y]$.

(3) A' is closed under $\underset{1}{\sim}$ iff $\forall x \in A' \forall y \in A[x \underset{1}{\sim} y \Rightarrow y \in A']$.

We first show sufficient conditions for $\underset{1}{=} = \underset{2}{=}$ in A'.

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

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

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

(3) A'' is reachable from A' under $\underset{1}{=}$.

Then $\underset{1}{=} = \underset{2}{=}$ in A'.

Proof. Prove $\forall x, y \in A'[x \underset{1}{=} y \iff x \underset{2}{=} y]$. $\Rightarrow$ is trivial from condition (1), hence, we will show $\Leftarrow$. Assume $x \underset{2}{=} y$, where $x, y \in A'$. By using condition (3), there are some elements $z, w \in A''$ 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 $x \underset{1}{=} z \underset{1}{=} w \underset{1}{=} y$. $\square$

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

Theorem 3.1. Assume the following conditions:

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

(2) $CR(R_2)$, $\overset{*}{\underset{2}{\rightarrow}} \subseteq \overset{*}{\underset{1}{\rightarrow}}$ in A'', and A'' is closed under $\overset{*}{\underset{2}{\rightarrow}}$,

(3) A'' is reachable from A' under $\underset{1}{=}$.

Then $\underset{1}{=} = \underset{2}{=}$ in A'.

Proof. Prove condition (2) of Lemma 3.1, i.e., $\forall x, y \in A''[x \underset{1}{=} y \iff x \underset{2}{=} y]$. $\Rightarrow$ is trivial from condition (1), hence we will prove $\Leftarrow$. Assume $x \underset{2}{=} y$, where $x, y \in A''$. From $CR(R_2)$, the closed property of A'' under $\overset{*}{\underset{2}{\rightarrow}}$, and Proposition 2.1(1); there exists some $z \in A''$ such that $x \overset{*}{\underset{2}{\rightarrow}} z$ and $y \overset{*}{\underset{2}{\rightarrow}} z$. By using $\overset{*}{\underset{2}{\rightarrow}} \subseteq \overset{*}{\underset{1}{\rightarrow}}$ in A'', $x \overset{*}{\underset{1}{\rightarrow}} z$ and $y \overset{*}{\underset{1}{\rightarrow}} z$ can be derived. Therefore $x \underset{1}{=} y$. $\square$

Theorem 3.2. Assume the following conditions:

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

(2) $CR(R_2)$ and $A'' \subseteq NF_2$,

(3) A'' is reachable from A' under $\underset{1}{=}$.

Then $\underset{1}{=} = \underset{2}{=}$ in A'.

Proof. Prove condition (2) of Lemma 3.1, i.e., $\forall x, y \in A''[x \underset{1}{=} y \iff x \underset{2}{=} y]$. $\Rightarrow$ is trivial. $\Leftarrow$: By using condition (2) of this theorem and Proposition 2.1(2), $x \underset{2}{=} y \Rightarrow x \equiv y$ for any $x, y \in A''$. Therefore $x \underset{1}{=} y$. $\square$

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 $A' = A$ and $A'' = NF_1 = NF_2$ in Theorem 3.2. □

Next, we consider sufficient conditions for $\overset{*}{\underset{1}{\to}} = \overset{*}{\underset{2}{\to}}$ in $A' \times A''$.

Theorem 3.3. Assume the following conditions:

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

(2) $CR(R_2)$ and $A'' \subseteq NF_2$,

(3) A'' is reachable from A' under $\overset{*}{\underset{1}{\to}}$.

Then $\overset{*}{\underset{1}{\to}} = \overset{*}{\underset{2}{\to}}$ in $A' \times A''$.

Proof. Prove $\forall x \in A' \forall y \in A''[x \overset{*}{\underset{1}{\to}} y \iff x \overset{*}{\underset{2}{\to}} y]$. $\Rightarrow$: Let $x \overset{*}{\underset{1}{\to}} y$. Then $x \underset{2}{=} y$ from condition (1). Thus $x \overset{*}{\underset{2}{\to}} y$ is obtained from condition (2) and Proposition 2.1(3). $\Leftarrow$: Let $x \overset{*}{\underset{2}{\to}} y$. Then, from condition (3), there exists some $z \in A''$ such that $x \overset{*}{\underset{1}{\to}} z$. By condition (1), $x \underset{2}{=} z$; hence, $y \underset{2}{=} z$ can be derived from $y \underset{2}{=} x \underset{2}{=} z$. Thus, $y \equiv z$ is obtained from condition (2) and Proposition 2.1(2). Therefore $x \overset{*}{\underset{1}{\to}} y$. □

Corollary 3.2. Assume the conditions:

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

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

(3) $WN(R_1)$.

Then $\overset{*}{\underset{1}{\to}} = \overset{*}{\underset{2}{\to}}$ in $A \times NF_1$.

Proof. Set $A' = A$ and $A'' = NF_1 = NF_2$ in Theorem 3.3. □

In the following sections, we will explain how to apply the above abstract results to term rewriting systems that are reduction systems having a term set as the object set. However, note that the above abstract results can be applied not only to term rewriting systems, but also to various reduction systems.

4 Term Rewriting Systems

Assuming that the reader is familiar with the basic concepts concerning term rewriting systems, we briefly summarize the important notions below, according to Huet[10], Huet and Oppen[11].

Let F be an enumerable set of function symbols denoted by $f, g, h, \cdots$, and let V be an enumerable set of variable symbols denoted by $x, y, z, \cdots$ where $F \cap V = \phi$. By $T(F, V)$, we denote the set of terms constructed from F and V. If V is empty, $T(F, V)$, denoted as $T(F)$, is the set of ground terms. A term set is sometimes denoted by T.

A substitution θ is a mapping from a term set $T(F, V)$ to $T(F, V)$ such that for term M, $\theta(M)$ is completely 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 $\square$ 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. We use the notation $C[\ ,\ldots,\ ]$ for the context containing n holes $(n \geq 0)$, and if $N_1, \ldots, N_n \in T(F, V)$, then $C[N_1, \ldots, N_n]$ denotes the result of placing $N_1, \ldots, N_n$ in the holes of $C[\ ,\ldots,\ ]$ 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 $N \subseteq M$.

A rewriting rule is a pair $\langle M_l, M_r \rangle$ of terms 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 and we write $M_l \triangleright M_r$ for $\langle M_l, M_r \rangle \in \triangleright$. The set $\triangleright$ of rewriting rules defines a reduction relation $\rightarrow$ on T as follows:

$$M \rightarrow N \text{ iff } M \equiv C[M_l\theta],\ N \equiv C[M_r\theta], \text{ and } M_l \triangleright M_r \quad \text{for some } M_l, M_r, C[\ ], \text{ and } \theta.$$

Definition. A term rewriting system R on T is a reduction system $R = \langle T, \rightarrow \rangle$ such that the reduction relation $\rightarrow$ on T is defined by a set $\triangleright$ of rewriting rules. We usually omit 'on T' if T is $T(F, V)$. If R has $M_l \triangleright M_r$, then we write $M_l \triangleright M_r \in R$.

A redex, or $\rightarrow$ redex, is a term $M_l\theta$ where $M_l \triangleright M_r$. $R \cup \{M \triangleright N\}$ (resp. $R - \{M \triangleright N\}$) denotes the term rewriting system obtained by adding the rule $M \triangleright N$ to R (resp. by removing the rule $M \triangleright N$ from R). If every variable in term M occurs only once, then M is called linear. We say that R is left-linear iff for any $M_l \triangleright M_r \in R$, M_l is linear.

Let $M \triangleright N$ and $P \triangleright Q$ be two rules in R. We assume that we have renamed the 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 [10, 19] . 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 $P \triangleright Q$ and $M \triangleright N$ are overlapping, and that the pair $\langle C[Q]\theta, N\theta\rangle$ of terms is critical in R [10, 11] . 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 the trivial pair $\langle N, N\rangle$. If R has no critical pair, then we say that R is nonoverlapping [10, 11, 19, 28] .

The critical pair for two term rewriting systems R_1 and R_2 can be defined in the same way. Let $M \triangleright N \in R_1$ and $P \triangleright Q \in R_2$ (or $M \triangleright N \in R_2$ and $P \triangleright Q \in R_1$). We then 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, we then say that R_1 and R_2 are nonoverlapping with each other [28] .

The following sufficient conditions for the Church-Rosser property are well known [10, 11, 19] .

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

Proposition 4.2. Let R be left-linear and nonoverlapping. Then R has the Church-Rosser property.

Let $R_1 = \langle T, \underset{1}{\rightarrow}\rangle$ with $\underset{1}{\triangleright}$ and $R_2 = \langle T, \underset{2}{\rightarrow}\rangle$ with $\underset{2}{\triangleright}$. Then their union is defined by $R_1 \cup R_2 = \langle T, \rightarrow\rangle$ with $\underset{1}{\triangleright} \cup \underset{2}{\triangleright}$. The next proposition described in Toyama[28] plays an essential role in Section 7.

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

Remark. The Church-Rosser property on $T(F)$ follows from that on $T(F, V)$ since $T(F)$ is closed under the reduction relation. Hence, the following results and examples, based on the above propositions, concerning term rewriting systems on $T(F, V)$ also hold for term rewriting systems on $T(F)$. This remark is the basis of the comment and work [6] by Fribourg.

For more discussions concerning the Church-Rosser property of term rewriting systems having overlapping or nonlinear rules, see [10, 28, 29] .

5 Examples

We now illustrate how to prove the equivalence in a restricted domain of two term rewriting systems R_1 and R_2 by using Theorems 3.1, 3.2, and 3.3. We will omit here all proofs of reachability in the following examples. For the proofs of reachability for the examples in this and the following sections, see Appendix.

Example 5.1. Let $F' = \{+, s, 0\}$ and $F'' = \{s, 0\}$. Consider the term rewriting systems R_1 and R_2 computing the addition on the set $\mathbf{N}$:

$$R_1 \quad \begin{cases} x + 0 \triangleright x \\ x + s(y) \triangleright s(x + y) \end{cases}$$

$$R_2 \quad \begin{cases} x + 0 \triangleright x \\ x + s(y) \triangleright s(x + y) \\ x + (y + z) \triangleright (x + y) + z \end{cases}$$

We will prove that $\underset{1}{=} = \underset{2}{=}$ in $T(F')$ by using Theorem 3.2. Let $A' = T(F')$, $A'' = T(F'')$ in Theorem 3.2. We need to verify conditions (1), (2), (3) of Theorem 3.2 for

R_1 and R_2. Since $\underset{1}{\triangleright} \subseteq \underset{2}{\triangleright}$, condition (1), i.e., $\underset{1}{=} \subseteq \underset{2}{=}$, is obvious. By using $SN(R_2)$ and Proposition 4.1, $CR(R_2)$ is obtained. Condition (2) holds, since $T(F'') \subseteq NF_2$. $T(F'')$ is reachable from $T(F')$ under $\underset{1}{=}$. Therefore, $\underset{1}{=} = \underset{2}{=}$ in $T(F')$.

It is also possible to prove $\overset{*}{\underset{1}{\rightarrow}} = \overset{*}{\underset{2}{\rightarrow}}$ in $T(F') \times T(F'')$ by using Theorem 3.3. □

The next example shows that our result can be easily applied to the reduction systems having the relation between constructors such as $s^3(x) = x$.

Example 5.2. Let $F' = \{+, s, 0\}$ and $F'' = \{s, 0\}$. Consider the term rewriting systems R_1 and R_2 computing the addition on Z_3:

$$R_1 \quad \begin{cases} s(s(s(x))) \triangleright x \\ x + 0 \triangleright x \\ x + s(y) \triangleright s(x + y) \end{cases}$$

$$R_2 \quad \begin{cases} s(s(s(x))) \triangleright x \\ x + 0 \triangleright x \\ x + s(y) \triangleright s(x + y) \\ x + (y + z) \triangleright (x + y) + z \end{cases}$$

We will prove that $\underset{1}{=} = \underset{2}{=}$ in $T(F')$ by using Theorem 3.1. Let $A' = T(F')$, $A'' = T(F'')$. We need to verify conditions (1), (2), (3) of Theorem 3.1 for R_1 and R_2. Condition (1), i.e., $\underset{1}{=} \subseteq \underset{2}{=}$, is obvious. By using $SN(R_2)$ and Proposition 4.1, $CR(R_2)$ is obtained. Condition (2) holds since $\underset{1}{\rightarrow} = \underset{2}{\rightarrow}$ in $T(F'')$, and since $T(F'')$ is closed under $\underset{2}{\rightarrow}$. $T(F'')$ is reachable from $T(F')$ under $\underset{1}{=}$. Therefore, $\underset{1}{=} = \underset{2}{=}$ in $T(F')$.

Note that it is also possible to prove $\underset{1}{=} = \underset{2}{=}$ in $T(F')$ by letting $A'' = \{0, s(0), s(s(0))\}$ and by using Theorem 3.2. □

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

Example 5.3. Consider the following term rewriting systems R_1 and R_2 computing the *double* function $d(n) = 2 * n$:

$$R_1 \quad \begin{cases} d(0) \rhd 0 \\ d(s(x)) \rhd s(s(d(x))) \end{cases}$$

$$R_2 \quad \begin{cases} d(x) \rhd \mathit{if}(x, 0, s(s(d(x - s(0))))) \\ \mathit{if}(0, y, z) \rhd y \\ \mathit{if}(s(x), y, z) \rhd z \\ x - 0 \rhd x \\ s(x) - s(y) \rhd x - y \end{cases}$$

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 the function symbol d.

Let $F' = \{d, s, 0\}$ and $F'' = \{s, 0\}$. We will show that the function d of R_1 equals that of R_2 in the restricted domain $T(F')$, that is, $\underset{1}{=} = \underset{2}{=}$ in $T(F')$. For this purpose, Theorem 3.2 is used. Let $A' = T(F')$ and $A'' = T(F'')$. We must verify conditions (1), (2), (3) of Theorem 3.2. Since $d(0) \underset{2}{=} 0$ and $d(s(x)) \underset{2}{=} s(s(d(x)))$, condition (1), i.e., $\underset{1}{=} \subseteq \underset{2}{=}$, is obtained. It is obvious that R_2 is left-linear and nonoverlapping. Hence, by using Proposition 4.2, R_2 has the Church-Rosser property. Since some function symbol not in F'' appears in the left-hand side of any rewriting rule in R_2, we can obtain $T(F'') \subseteq NF_2$. Thus, condition (2) holds. $T(F'')$ is reachable from $T(F')$ under $\underset{1}{=}$. Therefore, $\underset{1}{=} = \underset{2}{=}$ in $T(F')$ holds.

Note that $T(F'')$ is also reachable from $T(F')$ under $\underset{1}{\overset{*}{\rightarrow}}$. Hence, by Theorem 3.3, we can prove $\underset{1}{\overset{*}{\rightarrow}} = \underset{2}{\overset{*}{\rightarrow}}$ in $T(F') \times T(F'')$ in the same way as the above proof. □

Remark. Reachability from $T(F)$ to $T(C)$ under $=$ has been called sufficient completeness [4, 8, 13, 15, 16, 17, 20, 22, 23] where C is the set of constructors. Clearly, sufficient completeness is a very particular case of reachability.

It is known that sufficient completeness is undecidable in general[8]. The decidability of sufficient completeness under certain conditions is discussed in [4, 13, 15, 16, 17, 20, 22, 23]. Hence, reachability is also undecidable in general, and the test of reachability under certain conditions has the same difficulty as that of sufficient completeness.

Remark. In the above examples, it is sufficient to consider the term rewriting systems on the set $T(F)$ of ground terms. Hence, it is not necessary to verify the Church-Rosser property on $T(F,V)$ but only on $T(F)$ (i.e., *ground confluence* [6, 25]).

6 Inductionless Induction

By using Theorem 3.2, an equation whose proof usually requires induction on some data structure can be proven without the explicit use of induction. In this section, we will explain how to prove an equation with the *inductionless induction* method[6, 7, 12, 13, 15, 17, 20, 21, 23] based on a very general framework. The framework makes it clear that the *inductionless induction* methods can work under a very weak assumption: the Church-Rosser property and reachability.

Let R_1 be a term rewriting system on $T(F,V)$ (or on $T(F)$). For a term set T, let $M \underset{1}{=} N$ in T denote $\forall\theta[M\theta, N\theta \in T \Rightarrow M\theta \underset{1}{=} N\theta]$. Now, for given terms $M, N \in T(F', V)$ such that any variable in N also occurs in M, consider the validity of $M \underset{1}{=} N$ in $T(F')$. Note that this validity cannot, in general, be proven by merely equational reasoning, that is, some kind of induction on $T(F')$ usually becomes necessary [7, 12, 21] . However, we can prove that $M \underset{1}{=} N$ in $T(F')$ by using the following theorem without induction.

Theorem 6.1. Let R_1 be a term rewriting system on $T(F,V)$ (or on $T(F)$) with reachability from $T(F')$ to $T(F'')$ under $\underset{1}{=}$. Let $R_2 = R_1 \cup \{M \triangleright N\}$. If R_2 has the Church-Rosser property and $T(F'') \subseteq NF_2$, then $M \underset{1}{=} N$ in $T(F')$.

Proof. It is obvious that R_1 and R_2 satisfy conditions (1), (2) and (3) of Theorem 3.2 by letting $A' = T(F')$ and $A'' = T(F'')$. Thus, $\underset{1}{=} = \underset{2}{=}$ in $T(F')$. Since $M \triangleright N \in R_2$, we can show that $M \underset{2}{=} N$ in $T(F')$. Therefore, $M \underset{1}{=} N$ in $T(F')$. □

Example 6.1. Consider R_1 defining the *half* function $h(n) = n/2$ and the *double* function $d(n) = 2 * n$:

$$R_1 \quad \begin{cases} h(0) \rhd 0 \\ h(s(0)) \rhd 0 \\ h(s(s(x))) \rhd s(h(x)) \\ d(0) \rhd 0 \\ d(s(x)) \rhd s(s(d(x))) \end{cases}$$

Let $F' = \{h, d, s, 0\}$ and $F'' = \{s, 0\}$. Now, let us prove $h(d(x)) \underset{1}{=} x$ in $T(F')$ by using Theorem 6.1. $T(F'')$ is reachable from $T(F')$ under $\underset{1}{=}$. Take $R_2 = R_1 \cup \{h(d(x)) \rhd x\}$. Then, $CR(R_2)$ by Proposition 4.1. Clearly, $T(F'') \subseteq NF_2$. Therefore, $h(d(x)) \underset{1}{=} x$ in $T(F')$. □

When $R_2 = R_1 \cup \{M \rhd N\}$ does not satisfy the conditions in Theorem 6.1, we may find R_3 instead of R_2 such that $CR(R_3)$, $T(F'') \subseteq NF_3$, and $\underset{2}{=} = \underset{3}{=}$ in $T(F')$.

Corollary 6.1. Let R_1 be a term rewriting system on $T(F, V)$ (or on $T(F)$) with reachability from $T(F')$ to $T(F'')$ under $\underset{1}{=}$. Let $R_2 = R_1 \cup \{M \rhd N\}$. If there exists a term rewriting system R_3 satisfying the Church-Rosser property, $T(F'') \subseteq NF_3$, and $\underset{2}{=} = \underset{3}{=}$ in $T(F')$; then $M \underset{1}{=} N$ in $T(F')$.

In Corollary 6.1 if the term rewriting system R_2 is strongly normalizing, then an effective search for R_3 can be executed by applying the Knuth-Bendix completion algorithm[19] to R_2. Thus, by using a modified Knuth-Bendix completion algorithm, we can prove automatically inductive theorems without the explicit use of induction. Hence, it has been called *inductionless induction*.

The original idea of the *inductionless induction* method was proposed by Musser [21] , and has been extended by Goguen [7] , Huet and Hullot [12] , and others [6, 13, 15, 17, 20, 23, 24, 27]. However, their *inductionless induction* methods have many limitations. In particular, the requirement for the strongly normalizing property of R_2[6, 7, 12, 21, 27] (or the strongly normalizing property on equivalence classes of terms if there are non-oriented equations such as associative/commutative laws [13, 15, 17, 20, 23]) restricts its application, since most term rewriting systems in which functions are denoted by recursive definitions, such as recursive programs, do not satisfy this property.

On the other hand, Corollary 6.1 have clarified that we can treat the *inductionless induction* concept itself apart from the Knuth-Bendix completion algorithm. Thus, in the *inductionless induction* proof technique if we do not use the Knuth-Bendix completion algorithm to find R_3, the strong normalizing limitation can be removed from R_2. We next show an example in which R_2 is not strongly normalizing.

Example 6.2.

$$R_1 \quad \begin{cases} d(x) \triangleright \mathit{if}(x, 0, s(s(d(x - s(0))))) \\ h(x) \triangleright \mathit{if}(x, 0, \mathit{if}(x - s(0), 0, s(h(x - s(s(0)))))) \\ \mathit{if}(0, y, z) \triangleright y \\ \mathit{if}(s(x), y, z) \triangleright z \\ x - 0 \triangleright x \\ s(x) - s(y) \triangleright x - y \end{cases}$$

Note that the term rewriting system R_1 does not have the strongly normalizing property, since the first and second rules in R_1 can be infinitely applied to the function symbols d and h respectively.

Let $F' = \{d, h, s, 0\}$ and $F'' = \{s, 0\}$. Now, we show that $h(d(x)) \underset{1}{=} x$ in $T(F')$. $T(F'')$ is reachable from $T(F')$ under $\underset{1}{=}$. Take $R_2 = R_1 \cup \{h(d(x)) \triangleright x\}$. R_2 is not strongly normalizing since R_1 is so. To easily show the Church-Rosser property of the term rewriting system obtained by adding the rule $h(d(x)) \triangleright x$, we consider R_3 instead of R_2:

$$R_3 \quad \begin{cases} d(0) \triangleright 0 \\ d(s(x)) \triangleright s(s(d(x))) \\ h(0) \triangleright 0 \\ h(s(0)) \triangleright 0 \\ h(s(s(x))) \triangleright s(h(x)) \\ \mathit{if}(0, y, z) \triangleright y \\ \mathit{if}(s(x), y, z) \triangleright z \\ x - 0 \triangleright x \\ s(x) - s(y) \triangleright x - y \\ h(d(x)) \triangleright x \end{cases}$$

Then, $\underset{2}{=} \; = \; \underset{3}{=}$ in $T(F')$ can be proven in the same way as for Example 5.3. It is shown from Proposition 4.1 that R_3 has the Church-Rosser property. Clearly, $T(F'') \subseteq NF_3$. Hence, R_3 satisfies the conditions in Corollary 6.1. Therefore, $h(d(x)) \underset{1}{=} x$ in $T(F')$. $\square$

Note. There is a practical difficulty in directly applying the above method to automated deduction. The basis of the method is an equivalence transformation from R_2 to R_3. However, at this stage we know no automatic transformation method in which the strong normalizing limitation on R_2 is unnecessary. Thus, an automatic transformation method must still be found.

7 Equivalence Transformation Technique

In this section, we propose the equivalence transformation rules for term rewriting systems. We show that the equivalence of term rewriting systems, to which it is difficult to apply Theorem 3.2 and 3.3 directly, can be easily proven by an equivalence transformation technique. The results are effectively applied to proving the correctness of program transformations proposed by Burstall and Darlington[3].

Let $R_0 = \langle T, \underset{0}{\rightarrow} \rangle$ with $\triangleright$ be a left-linear term rewriting system having the Church-Rosser property. Here, T is $T(F, V)$ or $T(F)$. Let F' and F'' be the subsets of F and let $T(F'') \subseteq NF_0$. Let $S_0 = \triangleright$. Now, we give the equivalence transformation rules from $R_n = \langle T, \underset{n}{\rightarrow} \rangle \ \ (n \geq 0)$ to $R_{n+1} = \langle T, \underset{n+1}{\rightarrow} \rangle$:

I. **Introduction**: Introduce a new relation by adding a new rewriting rule $P \triangleright Q$ to R_n; where $P \triangleright Q$ is not overlapping with any rule in S_n, and P is linear and has at least one function symbol not in F''. Thus, $R_{n+1} = R_n \cup \{P \triangleright Q\}$. Set $S_{n+1} = S_n \cup \{P \triangleright Q\}$.

A. **Addition**: Add an extra rule $P \triangleright Q$ to R_n, where $P \underset{n}{=} Q$. Thus, $R_{n+1} = R_n \cup \{P \triangleright Q\}$. Set $S_{n+1} = S_n$.

E. **Elimination**: Remove a rule $P \triangleright Q$ from R_n. Thus, $R_{n+1} = R_n - \{P \triangleright Q\}$. Set $S_{n+1} = S_n$.

Remark. The above three rules are a natural extension of the program transformation rules suggested by Burstall and Darlington [3] : *Definition, Instantiation, Unfolding, Folding, Abstraction*, and *Laws.* We can easily show that *Instantiation, Unfolding, Folding*, and *Laws* can be obtained directly from rule A; and *Definition* from rule I. *Abstraction* can be also obtained by combining the above three rules. Hence, their program transformations can be seen as a particular case of our equivalence transformations for term rewriting systems in restricted domains. Indeed, we can give a formal proof to the correctness of the program transformations by the technique developed in this section: see Examples 7.1 and 7.2.

$R_n \underset{i}{\Rightarrow} R_{n+1}$ shows that R_n is transformed to R_{n+1} by rule i ($i = I, A$, or E). $R_n \Rightarrow R_{n+1}$ shows that R_n is transformed to R_{n+1} by rule I, A, or E. $\overset{*}{\underset{i}{\Rightarrow}}$ and $\overset{*}{\Rightarrow}$ denote the transitive reflexive closure of $\underset{i}{\Rightarrow}$ and $\Rightarrow$, respectively.

Lemma 7.1. If $R \underset{i}{\Rightarrow} R' \underset{j}{\Rightarrow} \tilde{R}$ ($i = E$ and $j = I$, $i = E$ and $j = A$, or $i = A$ and $j = I$), then there is some R'' such that $R \underset{j}{\Rightarrow} R'' \underset{i}{\Rightarrow} \tilde{R}$.

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

Lemma 7.2. Let $R \overset{*}{\Rightarrow} \tilde{R}$. Then, there exists a transformation sequence from R to $\tilde{R}$ such that $R \overset{*}{\underset{I}{\Rightarrow}} R' \overset{*}{\underset{A}{\Rightarrow}} R'' \overset{*}{\underset{E}{\Rightarrow}} \tilde{R}$.

Proof. By using Lemma 7.1 repeatedly, we can construct a sequence $R \overset{*}{\underset{I}{\Rightarrow}} R' \overset{*}{\underset{A}{\Rightarrow}} R'' \overset{*}{\underset{E}{\Rightarrow}} \tilde{R}$ from $R \overset{*}{\Rightarrow} \tilde{R}$. □

Theorem 7.1. Let R_0 be a left-linear term rewriting system on $T(F, V)$ (or on $T(F)$) having the Church-Rosser property. Let F' and F'' be the subsets of F and let $T(F'') \subseteq NF_0$. Let $R_0 \overset{*}{\Rightarrow} R_m$, and let $T(F'')$ be reachable from $T(F')$ under $\underset{0}{=}$ and under $\underset{m}{=}$ (resp. under $\overset{*}{\underset{0}{\rightarrow}}$ and under $\overset{*}{\underset{m}{\rightarrow}}$). Then, $\underset{0}{=} = \underset{m}{=}$ in $T(F')$ (resp. $\overset{*}{\underset{0}{\rightarrow}} = \overset{*}{\underset{m}{\rightarrow}}$ in $T(F') \times T(F'')$).

Proof. We prove here only $\underset{0}{=} = \underset{m}{=}$ in $T(F')$, since the proof of $\overset{*}{\underset{0}{\rightarrow}} = \overset{*}{\underset{m}{\rightarrow}}$ in $T(F') \times T(F'')$ can be obtained in an analogous way. From Lemma 7.2, we may assume that $R_0 \overset{*}{\underset{I}{\Rightarrow}} R_p \overset{*}{\underset{A}{\Rightarrow}} R_q \overset{*}{\underset{E}{\Rightarrow}} R_m$. To prove the theorem we will show that $\underset{0}{=} = \underset{p}{=}$ in $T(F')$ and $\underset{p}{=} = \underset{m}{=}$ in $T(F')$.

Consider $R_0 \overset{*}{\underset{I}{\Rightarrow}} R_p$. It is clear that $\underset{0}{=} \subseteq \underset{p}{=}$. Let R' be the term rewriting system defined by $S_p - S_0$, i.e., the set of new rules introduced through $R_0 \overset{*}{\underset{I}{\Rightarrow}} R_p$. Then R_p is the union of R_0 and R'. Since R' is left-linear and nonoverlapping, $CR(R')$ can be proven by using Proposition 4.2. It is obvious that R_0 and R' are nonoverlapping with each other. Hence, by Proposition 4.3, $CR(R_p)$ is obtained. Since the left-hand side of each introduced rule has at least one function symbol not in F'' and since $T(F'') \subseteq NF_0$, we can obtain $T(F'') \subseteq NF_p$. It has been assumed that $T(F'')$ is reachable from $T(F')$ under $\underset{0}{=}$. Hence, by using Theorem 3.2, we can obtain $\underset{0}{=} = \underset{p}{=}$ in $T(F')$.

From $R_p \overset{*}{\underset{A}{\Rightarrow}} R_q$, $\underset{p}{=} = \underset{q}{=}$ is trivial.

Now consider $R_q \overset{*}{\underset{E}{\Rightarrow}} R_m$. Since $\underset{m}{=} \subseteq \underset{q}{=}$ and $\underset{q}{=} = \underset{p}{=}$, it is obvious that $\underset{m}{=} \subseteq \underset{p}{=}$. It has been shown that $CR(R_p)$ and $T(F'') \subseteq NF_p$. It has been assumed that $T(F'')$ is reachable from $T(F')$ under $\underset{m}{=}$. Hence, by Theorem 3.2 for R_m and R_p, it can be proven that $\underset{p}{=} = \underset{m}{=}$ in $T(F')$.

Therefore, it follows that $\underset{0}{=} = \underset{m}{=}$ in $T(F')$. $\square$

We now explain how to show the equivalence of two term rewriting systems by using the equivalence transformation technique. We take the examples discussed in [3] as examples of program transformations. Thus, the following examples make it clear that the extended *inductionless induction* concept is a useful tool for proving the correctness of the program transformations.

Example 7.1 (Summation). Consider the following term rewriting systems R_1 and R_2 computing the summation function $f(n) = n + \cdots + 1 + 0$:

$$R_1 \quad \begin{cases} f(0) \triangleright 0 \\ f(s(x)) \triangleright s(x) + f(x) \\ x + 0 \triangleright x \\ x + s(y) \triangleright s(x + y) \end{cases}$$

$$R_2 \quad \begin{cases} f(0) \triangleright 0 \\ f(s(x)) \triangleright g(x, s(x)) \\ g(0, y) \triangleright y \\ g(s(x), y) \triangleright g(x, y + s(x)) \\ x + 0 \triangleright x \\ x + s(y) \triangleright s(x + y) \end{cases}$$

Let $F' = \{f, +, s, 0\}$ and $F'' = \{s, 0\}$. By using the equivalence transformation rules, we will show that $\underset{1}{=} = \underset{2}{=}$ in $T(F')$. Note that R_2 is in *iterative form* [3]. Hence, this transformation technique can be used to convert *recursive form* into *iterative form* for functional programs.

To transform R_1 to R_2, we first add the associative law for $+$ to R_1: take $R_3 = R_1 \cup \{x + (y + z) \triangleright (x + y) + z\}$. Here, $R_3 \Rightarrow R_1$ by rule E. From Proposition 4.1, $CR(R_3)$ is proven. Clearly, $T(F'') \subseteq NF_3$. $T(F'')$ is reachable from $T(F')$ under $\underset{1}{=}$ (and also under $\underset{3}{=}$). By Theorem 7.1, $\underset{1}{=} = \underset{3}{=}$ in $T(F')$ is obtained.

Now let us transform R_3 to R_2 by using the transformation rules. Through rule I, we introduce a new function g:

(1) $g(x, y) \triangleright y + f(x)$.

Let $R_4 = R_3 \cup \{(1)\}$, then we can prove

$$f(s(x)) \underset{4}{=} g(x, s(x)),$$

$$g(0, y) \underset{4}{=} y,$$

$$g(s(x), y) \underset{4}{=} y + f(s(x)) \underset{4}{=} y + (s(x) + f(x)) \underset{4}{=} (y + s(x)) + f(x) \underset{4}{=} g(x, y + s(x)).$$

Utilizing rule A, we can obtain $R_5 = R_4 \cup \{(2), (3), (4)\}$:

(2) $f(s(x)) \triangleright g(x, s(x))$,

(3) $g(0, y) \triangleright y$,

(4) $g(s(x), y) \triangleright g(x, y + s(x))$.

Finally, by employing rule E, remove unnecessary rules $x + (y + z) \triangleright (x + y) + z$, $f(s(x)) \triangleright s(x) + f(x)$, and $g(x, y) \triangleright y + f(x)$ from R_5. Thus, we can obtain R_2. $T(F'')$ is reachable from $T(F')$ under $\underset{2}{=}$. Hence, $\underset{3}{=} = \underset{2}{=}$ in $T(F')$ is obtained by Theorem 7.1.

Now, we obtain an equivalence transformation sequence from R_1 to R_2: $R_1 \Leftarrow R_3 \Rightarrow R_4 \overset{*}{\Rightarrow} R_5 \overset{*}{\Rightarrow} R_2$. Therefore, $\underset{1}{=} = \underset{2}{=}$ in $T(F')$. It is also possible to prove $\underset{1}{\overset{*}{\rightarrow}} = \underset{2}{\overset{*}{\rightarrow}}$ in $T(F') \times T(F'')$ by analogous transformation. $\square$

Example 7.2 (Fibonacci). Consider the following term rewriting systems R_1 and R_2 computing the *Fibonacci* function $f(n + 2) = f(n + 1) + f(n)$ where $f(0) = 1, f(1) = 1$:

$$R_1 \quad \begin{cases} f(0) \rhd s(0) \\ f(s(0)) \rhd s(0) \\ f(s(s(x))) \rhd f(s(x)) + f(x) \\ x + 0 \rhd x \\ x + s(y) \rhd s(x + y) \end{cases}$$

$$R_2 \quad \begin{cases} f(0) \rhd s(0) \\ f(s(0)) \rhd s(0) \\ f(s(s(x))) \rhd p(h(g(x))) \\ g(0) \rhd \langle s(0), s(0) \rangle \\ g(s(x)) \rhd h(g(x)) \\ h(x) \rhd \langle p(x) + q(x), p(x) \rangle \\ p(\langle x, y \rangle) \rhd x \\ q(\langle x, y \rangle) \rhd y \\ x + 0 \rhd x \\ x + s(y) \rhd s(x + y) \end{cases}$$

Let $F' = \{f, +, s, 0\}$ and $F'' = \{s, 0\}$. By using the equivalence transformation rules, we will show that $\underset{1}{=} = \underset{2}{=}$ in $T(F')$. Note that if we ignore the computation time for the function $+$ as a primitive function, R_2 computes the *Fibonacci* function in linear time instead of exponential time [3].

From Proposition 4.1, $CR(R_1)$ is proven. Clearly, $T(F'') \subseteq NF_1$. $T(F'')$ is reachable from $T(F')$ under $\underset{1}{=}$. To transform R_1 to R_2, we first introduce the pairing $\langle\ ,\ \rangle$ and the projection functions p, q by rule I:

(1) $p(\langle x, y \rangle) \rhd x$,

(2) $q(\langle x, y \rangle) \rhd y$.

Through rule I, we introduce a new function g:

(3) $g(x) \rhd \langle f(s(x)), f(x) \rangle$.

Let $R_3 = R_1 \cup \{(1), (2), (3)\}$. Then we can prove

$g(0) \underset{3}{=} \langle s(0), s(0) \rangle$,

$g(s(x)) \underset{3}{=} \langle f(s(x)) + f(x), f(s(x)) \rangle \underset{3}{=} \langle p(g(x)) + q(g(x)), p(g(x)) \rangle.$

Hence, we can add the following rules by using rule A:

(4) $g(0) \triangleright \langle s(0), s(0) \rangle$,

(5) $g(s(x)) \triangleright \langle p(g(x)) + q(g(x)), p(g(x)) \rangle$.

Here, we introduce a new function h through rule I to avoid computing $g(x)$ three times in the right-hand side of (5),

(6) $h(x) \triangleright \langle p(x) + q(x), p(x) \rangle$.

Let $R_4 = R_3 \cup \{(4), (5), (6)\}$. Then we can prove

$g(s(x)) \underset{4}{=} h(g(x))$,

$f(s(s(x))) \underset{4}{=} p(g(s(x))) \underset{4}{=} p(h(g(x)))$.

By employing rule A, we can obtain $R_5 = R_4 \cup \{(7), (8)\}$:

(7) $g(s(x))) \triangleright h(g(x))$,

(8) $f(s(s(x))) \triangleright p(h(g(x)))$.

Finally, by using rule E, remove unnecessary rules $f(s(s(x))) \triangleright f(s(x)) + f(x)$, $g(x) \triangleright \langle f(s(x)), f(x) \rangle$ and $g(s(x)) \triangleright \langle p(g(x)) + q(g(x)), p(g(x)) \rangle$ form R_5. Thus, we can obtain R_2. $T(F'')$ is reachable from $T(F')$ under $\underset{2}{=}$. Hence, $\underset{1}{=} = \underset{2}{=}$ in $T(F')$ is obtained by Theorem 7.1. We can also prove $\overset{*}{\underset{1}{\rightarrow}} = \overset{*}{\underset{2}{\rightarrow}}$ in $T(F') \times T(F'')$ by analogous method. □

8 Conclusion

In this paper, we have proposed a new simple method to prove the equivalence in a restricted domain for reduction systems without the explicit use of induction. The key idea is that the equivalence in the restricted domain can be easily tested by using the Church-Rosser property and reachability of reduction systems. Our method has extended the *inductionless induction* methods developed by Musser [21], Goguen [7], Huet and Hullot [12], and others [6, 13, 15, 17, 20, 23, 24, 27] as follows:

(1) The *inductionless induction* methods are based on the framework of term rewriting systems. Our method is essentially based on a more general framework of abstract reduction systems in which we assume only abstract structures. Hence, our method can be applied directly not only to term rewriting systems, but also to various reduction systems: *Thue* systems [2], graph rewriting systems [26], lambda calculus [1], combinatory reduction systems [18], resolution systems [9, 14, 24], and so on.

(2) The *inductionless induction* methods deal with only the inductive equality of two equational theories. On the other hand, our method extends the *inductionless induction* concept to a computational aspect, i.e., *reduction*. The method can deal with the inductive equality of two reduction systems, i.e., the inductive equality of two *reductions*, in the same way as that of two equational theories. This extension is very important in practice; for instance, we have shown that our results can be effectively applied to proving the correctness of program transformations proposed by Burstall and Darlington[3].

(3) The *inductionless induction* methods have many limitations [12] for term rewriting systems: the Church-Rosser property, the strongly normalizing property, the partition of the function symbols into constructors and non-constructors, and sufficient completeness. These limitations were partially relaxed by several authors [6, 13, 15, 17, 20, 23, 24, 27]; for example, the second restriction can be replaced with the strongly normalizing property on equivalence classes of terms if there are non-oriented equations such as associative/commutative laws [13, 17, 20, 23], and the third restriction can be completely removed [13, 15]. However, our method has made it clear that the *inductionless induction* concept essentially requires only two limitations: the Church-Rosser property and reachability which is an abstract extension of sufficient completeness.

We believe that our method provides a very useful means of proving the equivalence which arises in various formal systems: automated theorem proving, semantics of functional programs, program transformation, program verification, and specification of abstract data types.

Appendix

We present a simple method for verifying reachability by using the idea of the test set [4, 12, 13, 15, 16, 25, 27]. From here on we assume that term rewriting system R is left-linear.

We first explain the notations. Let $C[\ ,\dots,\ ]$ be a context and let $T_1,\cdots,T_n$ $(n \geq 0)$ be the non-empty sets of terms. Then the set of terms $C[T_1,\dots,T_n] = \{C[M_1,\dots M_n] \mid \ M_i \in T_i\}$. A substitution φ is a mapping from a term set T to a power set 2^T such that for a term $M \in T$, $\varphi(M) \subseteq T$ is completely determined by its values $\varphi(x),\dots,\varphi(z) \subseteq T$ on the variable symbols $x,\dots,z$ occurring in M. We define $\varphi(M) = \{M\}$ if no variable symbol occurs in M. $\varphi(C)[T_1,\dots,T_n]$ denotes the term set obtained by replacing the variable symbols $x,\dots,z$ and the holes $\Box$ occurring in $C[\ ,\dots,\ ]$ with $\varphi(x),\dots,\varphi(z)$ and $T_1,\dots,T_n$, respectively. A finite set of linear terms $S = \{M_1,\dots,M_s\}$ is a test set for a term set T with φ if $T = \bigcup_i \varphi(M_i)$. For example, $S = \{x+0, x+s(y)\}$ is a test set for the term set $\mathbf{N}+\mathbf{N}$ with $\varphi(x) = \varphi(y) = \mathbf{N}$. Here, $\mathbf{N} = T(\{0,s\})$. For more discussions concerning the test set, see [4, 12, 13, 15, 16, 25, 27].

We say a well-founded partial ordering $>$ on a term set T is stable under substitution if $M\theta > N\theta$ for any θ and each $M > N$. Here, θ is a substitution from T to T, and $M,N \in T$. For example, let l be a fixed positive number and let $\underset{l}{>}$ be the well-founded ordering on a term set T defined by $f(M_1,\dots,M_n) \underset{l}{>} f(N_1,\dots,N_n)$ iff $N_l \subseteq M_l$ and $N_l \not\equiv M_l$ (i.e., N_l is a proper subterm of M_l). Then, it is obvious that $\underset{l}{>}$ is stable under substitution.

We often write $\vec{T}$ for a direct product of sets $T \times \cdots \times T$ and $\vec{M}$ for an ordered n-tuples $\langle M_1,\dots,M_n\rangle \in T_1 \times \cdots \times T_n$. We write $f(\vec{M})$ for $f(M_1,\dots,M_n)$ and $f(\vec{T})$ for $f(T,\dots,T)$.

$T \Rightarrow T'$ denotes that T' is reachable from T under $\xrightarrow{*}$ (and also $=$).

The following properties can be easily demonstrated from the above definitions.

Properties.

A.1. If $T \subseteq T'$ then $T \Rightarrow T'$.

A.2. If $T \Rightarrow T'$ and $T' \Rightarrow T''$ then $T \Rightarrow T''$.

A.3. If $T_i \Rightarrow T'_i$ $(i = 1,\dots,n)$ then $C[T_1,\dots,T_n] \Rightarrow C[T'_1,\dots,T'_n]$.

A.4. If $f(\overrightarrow{T(F)}) \Rightarrow T(F)$ then $T(F \cup \{f\}) \Rightarrow T(F)$.

A.5. If $T(F \cup \{f\}) \Rightarrow T$ and $T(F \cup \{f'\}) \Rightarrow T$ then $T(F \cup \{f, f'\}) \Rightarrow T$.

A.6. If $M \xrightarrow{*} N$ then $\varphi(M) \Rightarrow \varphi(N)$.

We can prove reachability for the examples in this paper by using the properties and the following theorem. Note that our method does not assume the strongly normalizing property for term rewriting systems.

Theorem A.1. Let R be a left-linear term rewriting system and let $>$ be a well-founded partial ordering on a term set that is stable under substitution. Let $S = \{f(\vec{P_1}), \ldots, f(\vec{P_p})\}$ be a test set for $f(\overrightarrow{T(F)})$ with $\varphi(x) = T(F)$ for all $x \in V$. Then $f(\overrightarrow{T(F)}) \Rightarrow T$ if R satisfies the following conditions: For any $f(\vec{P_i}) \in S$ there exists a term Q_i such that

$f(\vec{P_i}) \xrightarrow{*} Q_i \equiv C[f(\vec{A_1}), \ldots, f(\vec{A_q})]$ $(q \geq 0)$

where C contains no function symbol f and

(1) $f(\vec{P_i}) > f(\vec{A_j})$ for all j,

(2) $f(\vec{A_j}) \in f(\overrightarrow{T(F, V)})$ for all j,

(3) $\varphi(C)[T, \ldots, T] \Rightarrow T$.

Proof. Make the rewriting rules $f(\vec{P_i}) \triangleright Q_i$ from each pair of $f(\vec{P_i})$ and Q_i $(i = 1, \ldots, p)$. Let R_S be the term rewriting system defined by these rules. Then, $\underset{S}{\rightarrow} \subseteq \xrightarrow{*}$. Note that any $f(\vec{M}) \in f(\overrightarrow{T(F)})$ is a $\underset{S}{\rightarrow}$ redex since S is a test set for $f(\overrightarrow{T(F)})$.

Let $M_0 \in f(\overrightarrow{T(F)})$. First, we show that there is no infinite reduction sequence $M_0 \underset{S}{\rightarrow} M_1 \underset{S}{\rightarrow} M_2 \underset{S}{\rightarrow} \cdots$. Let $OC(M_i)$ denote the multiset of the $\underset{S}{\rightarrow}$ redex occurrences in M_i. From the stability under substitution of $>$ and condition (1) in the theorem, $OC(M_i) \gg OC(M_{i+1})$ is obtained. Here, $\gg$ is the multiset ordering extended from $>$. $\gg$ is well-founded since $>$ is so [5]. Thus, the above reduction sequence must terminate eventually into a $\underset{S}{\rightarrow}$ normal form. Note that the $\underset{S}{\rightarrow}$ normal form contains no term in $f(\overrightarrow{T(F)})$ as a subterm occurrence. We write $d(M_0)$ for the maximal length of the $\underset{S}{\rightarrow}$ reduction sequences starting with M_0 into a $\underset{S}{\rightarrow}$ normal form of M_0.

Now, we show $\forall f(\vec{M}) \in f(\overrightarrow{T(F)}), \exists N \in T, f(\vec{M}) \xrightarrow{*} N$ by induction on $d(f(\vec{M}))$.

Basis ($d(f(\vec{M})) = 1$). There exist some $f(\vec{P_i}) \triangleright Q_i \in R_S$ and some θ such that $f(\vec{M}) \equiv f(\vec{P_i})\theta \underset{S}{\rightarrow} Q_i\theta$, where $Q_i\theta$ is a $\underset{S}{\rightarrow}$ normal form. Since $Q_i\theta$ contains no term in $f(\overrightarrow{T(F)})$, $\varphi(Q_i) \Rightarrow T$ follows from condition (3) in the theorem. From $f(\vec{P_i})\theta \in f(\overrightarrow{T(F)})$ and $f(\vec{P_i}) \xrightarrow{*} Q_i$, we can show $Q_i\theta \in \varphi(Q_i)$. Therefore, there exists $N \in T$ such that $f(\vec{M}) \xrightarrow{*} Q_i\theta \xrightarrow{*} N$.

Induction. Assume that the theorem is true for $d(f(\vec{M})) \leq k$ $(k \geq 1)$. Now, we will show the case for $d(f(\vec{M})) = k+1$. There exist some $f(\vec{P_i}) \triangleright Q_i \in R_S$ and some θ such that $f(\vec{M}) \equiv f(\vec{P_i})\theta \underset{S}{\rightarrow} Q_i\theta \equiv C[f(\vec{A_1}), \ldots, f(\vec{A_q})]\theta \equiv C\theta[f(\vec{A_1})\theta, \ldots, f(\vec{A_q})\theta]$ $(q \geq 1)$. From condition (2) in the theorem, $f(\vec{A_j})\theta \in f(\overrightarrow{T(F)})$ for all j. From the inductive hypothesis, we can obtain $N_j \in T$ $(j = 1, \ldots, q)$ such that $f(\vec{A_j})\theta \xrightarrow{*} N_j$ since $d(f(\vec{M})) > d(f(\vec{A_j})\theta)$.

Thus, it follows that $f(\vec{M}) \xrightarrow{*} C\theta[f(\vec{A_1})\theta, \ldots, f(\vec{A_q})\theta] \xrightarrow{*} C\theta[N_1, \ldots, N_q] \in \varphi(C)[T, \ldots, T]$. Therefore, by condition (3) in the theorem, there exists $N \in T$ such that $C\theta[N_1, \ldots, N_q]$ $\xrightarrow{*} N$. □

We now illustrate how to prove reachability by using the above theorem.

Example A.1. Consider the term rewriting system R_1 in Example 5.1 to be R.

$$R \quad \begin{cases} x + 0 \triangleright x \\ x + s(y) \triangleright s(x + y) \end{cases}$$

We will prove $T(\{+, s, 0\}) \Rightarrow \mathbf{N}$. From Property A.4, it is sufficient to prove $\mathbf{N} + \mathbf{N} \Rightarrow \mathbf{N}$ by using the theorem. Take the test set $S = \{x + 0, x + s(y)\}$ for $\mathbf{N} + \mathbf{N}$ with $\varphi(x) = \varphi(y) = \mathbf{N}$ and the well-founded ordering $\underset{2}{>}$. Note that $\underset{2}{>}$ is stable under substitution. We must verify the conditions in the theorem for $x + 0$ and $x + s(y)$ respectively.

For $x + 0$, $x + 0 \xrightarrow{*} x$. Consider the pair of $x + 0$ and x. Then, the pair trivially satisfies conditions (1) and (2) in the theorem. Condition (3) is also satisfied since

$\varphi(x) = \mathbf{N}$.

For $x + s(y)$, $x + s(y) \xrightarrow{*} s(x + y)$. We must verify the conditions in the theorem for the pair of $x + s(y)$ and $s(x + y) \equiv s(\Box)[x + y]$. Condition (1) is satisfied since $x + s(y) \underset{2}{>} x + y$. Condition (2) is satisfied since $x + y \in T(\{s, 0\}, V) + T(\{s, 0\}, V)$. Since $\varphi(s(\Box))[\mathbf{N}] = s(\mathbf{N})$, condition (3) is satisfied. Therefore, $\mathbf{N} + \mathbf{N} \Rightarrow \mathbf{N}$ follows from the theorem. $\Box$

Example A.2. Consider the term rewriting system R_1 in Example 6.2 to be R. Note that R does not have the strongly normalizing property.

$$R \quad \begin{cases} d(x) \rhd \mathit{if}(x, 0, s(s(d(x - s(0))))) \\ h(x) \rhd \mathit{if}(x, 0, \mathit{if}(x - s(0), 0, s(h(x - s(s(0)))))) \\ \mathit{if}(0, y, z) \rhd y \\ \mathit{if}(s(x), y, z) \rhd z \\ x - 0 \rhd x \\ s(x) - s(y) \rhd x - y \end{cases}$$

We will prove $T(\{d, h, s, 0\}) \Rightarrow \mathbf{N}$. From Property A.5, it is sufficient to prove $T(\{d, s, 0\}) \Rightarrow \mathbf{N}$ and $T(\{h, s, 0\}) \Rightarrow \mathbf{N}$.

Let us prove $T(\{d, s, 0\}) \Rightarrow \mathbf{N}$. From Property A.4, it is sufficient to prove $d(\mathbf{N}) \Rightarrow \mathbf{N}$ by using the theorem. Take the test set $S = \{d(0), d(s(x))\}$ for $d(\mathbf{N})$ with $\varphi(x) = \mathbf{N}$ and the substitution preserved well-founded ordering $\underset{1}{>}$. We must verify the conditions in the theorem for $d(0)$ and $d(s(x))$, respectively.

For $d(0)$, $d(0) \xrightarrow{*} 0$. Consider the pair of $d(0)$ and 0. Then, the conditions are trivially satisfied.

For $d(s(x))$, $d(s(x)) \xrightarrow{*} s(s(d(x)))$. Consider the pair of $d(s(x))$ and $s(s(d(x))) \equiv s(s(\Box))[d(x)]$. Condition (1) is satisfied since $d(s(x)) \underset{1}{>} d(x)$. Condition (2) is satisfied since $d(x) \in d(T(\{s, 0\}, V))$. Since $\varphi(s(s(\Box)))[\mathbf{N}] = s(s(\mathbf{N}))$, condition (3) is also satisfied. Hence, $T(\{d, s, 0\}) \Rightarrow \mathbf{N}$ follows from the theorem.

Next, let us show $T(\{h, s, 0\}) \Rightarrow \mathbf{N}$. From property A.4, it is sufficient to prove $h(\mathbf{N}) \Rightarrow \mathbf{N}$ by using the theorem. Take the test set $S = \{h(0), h(s(0)), h(s(s(x)))\}$ for $h(\mathbf{N})$ with $\varphi(x) = \mathbf{N}$ and the well-founded ordering $\underset{1}{>}$. Note that $\underset{1}{>}$ is stable under substitution. We need to verify the conditions in the theorem for $h(0)$, $h(s(0))$, and $h(s(s(x)))$, respectively. For $h(0)$ and $h(s(0))$, it can be easily shown that the conditions are satisfied.

For $h(s(s(x)))$, $h(s(s(x))) \xrightarrow{*} s(h(x))$. Consider the pair of $h(s(s(x)))$ and $s(h(x)) \equiv s(\Box)[h(x)]$. Condition (1) is satisfied since $h(s(s(x))) \underset{1}{>} h(x)$. Condition (2) is satisfied since $h(x) \in h(T(\{s, 0\}, V))$. Since $\varphi(s(\Box))[\mathbf{N}] = s(\mathbf{N})$, condition (3) is satisfied. Hence, $T(\{h, s, 0\}) \Rightarrow \mathbf{N}$ follows from the theorem. $\Box$

Example A.3. Consider the term rewriting system R_2 in Example 7.2 to be R.

$$R \quad \begin{cases} f(0) \triangleright s(0) \\ f(s(0)) \triangleright s(0) \\ f(s(s(x))) \triangleright p(h(g(x))) \\ g(0) \triangleright \langle s(0), s(0) \rangle \\ g(s(x)) \triangleright h(g(x)) \\ h(x) \triangleright \langle p(x) + q(x), p(x) \rangle \\ p(\langle x, y \rangle) \triangleright x \\ q(\langle x, y \rangle) \triangleright y \\ x + 0 \triangleright x \\ x + s(y) \triangleright s(x + y) \end{cases}$$

We will prove $T(\{f, +, s, 0\}) \Rightarrow \mathbf{N}$. From $T(\{+, s, 0\}) \Rightarrow \mathbf{N}$ and Property A.5, it is sufficient to prove $T(\{f, s, 0\}) \Rightarrow \mathbf{N}$. From Property A.4, it is sufficient to prove $f(\mathbf{N}) \Rightarrow \mathbf{N}$. Take the test set $S = \{f(0), f(s(0)), f(s(s(x)))\}$ for $f(\mathbf{N})$ with $\varphi(x) = \mathbf{N}$ and the well-founded ordering $\underset{1}{>}$. Note that $\underset{1}{>}$ is stable under substitution. We must verify the conditions in the theorem for $f(0)$, $f(s(0))$ and $f(s(s(x)))$, respectively. For $f(0), f(s(0))$, it is obvious that the conditions are satisfied.

For $f(s(s(x)))$, $f(s(s(x))) \xrightarrow{*} p(h(g(x)))$. Consider the pair of $f(s(s(x)))$ and $p(h(g(x)))$. Then, conditions (1) and (2) are trivially satisfied. For condition (3), we must prove $\varphi(p(h(g(x)))) = p(h(g(\mathbf{N}))) \Rightarrow \mathbf{N}$. If $g(\mathbf{N}) \Rightarrow \langle \mathbf{N}, \mathbf{N} \rangle$ then the condition is satisfied since $p(h(g(\mathbf{N}))) \Rightarrow p(h(\langle \mathbf{N}, \mathbf{N} \rangle)) \Rightarrow p(\langle p(\langle \mathbf{N}, \mathbf{N} \rangle) + q(\langle \mathbf{N}, \mathbf{N} \rangle), p(\langle \mathbf{N}, \mathbf{N} \rangle) \rangle)$ $\Rightarrow p(\langle \mathbf{N} + \mathbf{N}, \mathbf{N} \rangle) \Rightarrow \mathbf{N} + \mathbf{N} \Rightarrow \mathbf{N}$.

Now, we prove $g(\mathbf{N}) \Rightarrow \langle \mathbf{N}, \mathbf{N} \rangle$ by using the theorem. Take the test set $S = \{g(0), g(s(x))\}$ for $g(\mathbf{N})$ with $\varphi(x) = \mathbf{N}$ and the well-founded ordering $\underset{1}{>}$. Note that $\underset{1}{>}$ is stable under substitution. We must verify the conditions in the theorem for $g(0)$ and $g(s(x))$, respectively.

For $g(0)$, $g(0) \xrightarrow{*} \langle s(0), s(0) \rangle$. Consider the pair of $g(0)$ and $\langle s(0), s(0) \rangle$. Then, conditions (1) and (2) are trivially satisfied. Condition (3) follows from $\varphi(\langle s(0), s(0) \rangle)$ $= \{\langle s(0), s(0) \rangle\} \Rightarrow \langle \mathbf{N}, \mathbf{N} \rangle$.

For $g(s(x))$, $g(s(x)) \xrightarrow{*} h(g(x))$. Consider the pair of $g(s(x))$ and $h(g(x)) \equiv h(\Box)[g(x)]$. Then, condition (1) is satisfied since $g(s(x)) \underset{1}{>} g(x)$. Condition (2) is satisfied since $g(x) \in g(T(\{s, 0\}, V))$. Condition (3) follows from $\varphi(h(\Box))[\langle \mathbf{N}, \mathbf{N} \rangle] = h(\langle \mathbf{N}, \mathbf{N} \rangle) \Rightarrow \langle p(\langle \mathbf{N}, \mathbf{N} \rangle) + q(\langle \mathbf{N}, \mathbf{N} \rangle), p(\langle \mathbf{N}, \mathbf{N} \rangle) \rangle \Rightarrow \langle \mathbf{N} + \mathbf{N}, \mathbf{N} \rangle \Rightarrow \langle \mathbf{N}, \mathbf{N} \rangle$. $\Box$

Acknowledgments

The author would like to thank Kokichi Futatsugi, Hirofumi Katsuno, and Shigeki Goto for their helpful suggestions, and the referees for their useful comments, which improved the presentation of our results.

References

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

[2] R.V. Book, Confluent and other types of Thue systems, *J. ACM 29* (1982) 171-182.

[3] R.M. Burstall and J. Darlington, A transformation system for developing recursive programs, *J. ACM 24* (1977) 44-67.

[4] H. Comon, Sufficient completeness, term rewriting systems and "anti-unification", *Lecture Notes in Comput. Sci. 230* (Springer-Verlag, 1986) 128-140.

[5] N. Dershowitz and Z. Manna, Proving termination with multiset orderings, *C. ACM 22* (1979) 465-476.

[6] L.Fribourg, A strong restriction of the inductive completion procedure, *Lecture Notes in Comp. Sci. 226* (Springer-Verlag, 1986) 105-115.

[7] J.A. Goguen, How to prove algebraic inductive hypotheses without induction, with applications to the correctness of data type implementation, *Lecture Notes in Comput. Sci. 87* (Springer-Verlag, 1980) 356-373.

[8] J.V. Guttag and J.J. Horning, The algebraic specification of abstract data types, *Acta Informatica 10* (1978) 27-52.

[9] J. Hsiang, Refutational theorem proving using term-rewriting systems, *Artificial Intelligence 25* (1985) 255-300.

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

[11] G. Huet and D.C. Oppen, Equations and rewrite rules: a survey, in: R.V. Book, ed., *Formal languages: perspectives and open problems*, (Academic Press, 1980) 349-405.

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

[13] J.P. Jouannaud and E. Kounalis, Automatic proofs by induction in theories without constructors, *L.R.I. Rapport de Recherch No 295* (1986).

[14] D. Kapur and P. Narendran, An equational approach to theorem proving in first-order predicate calculus, *General Electric Corporate Research Development Report, No.84CRD322* (1985).

[15] D. Kapur, P. Narendran, and H. Zhang, Proof by induction using test sets, *Lecture Notes in Comput. Sci. 230* (Springer-Verlag, 1986) 99-117.

[16] D. Kapur, P. Narendran, and H. Zhang, Complexity of sufficient-completeness, *Lecture Notes in Comput. Sci. 241* (Springer-Verlag, 1986) 426-442.

[17] H. Kirchner, A general inductive completion algorithm and application to abstract data types, *Lecture Notes in Comput. Sci. 170* (Springer-Verlag, 1985) 282-302.

[18] J.W. Klop, Combinatory reduction systems, Dissertation, Univ. of Utrecht, 1980.

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

[20] E. Kounalis, Completeness in data type specifications, *Lecture Notes in Comput. Sci. 204* (Springer-Verlag, 1985) 348-362.

[21] D.R. Musser, On proving inductive properties of abstract data types, *Proc. 7th ACM Sympo. Principles of Programming Languages* (1980) 154-162.

[22] T. Nipkow and G. Weikum, A decidability result about sufficient-completeness of axiomatically specified abstract data type, *Lecture Notes in Comput. Sci. 145* (Springer-Verlag, 1983) 257-267.

[23] E. Paul, Proof by induction in equational theories with relations between constructors, in: B. Courcelle, ed., *9th Colloquium on trees in algebra and programming* (Cambridge University Press, 1984) 211-225.

[24] E. Paul, On solving the equality problem in theories defined by Horn clauses, *Theoretical Comput. Sci. 44* (1986) 127-153.

[25] D.A. Plaisted, Semantic confluence tests and completion methods, *Information and Control 65* (1985) 182-215.

[26] J.C. Raoult, On graph rewriting, *Theoretical Comput. Sci. 32* (1984) 1-24.

[27] J.J. Thiel, Stop losing sleep over incomplete data type specifications, *Proc. 11th ACM Sympo. Principles of Programming Languages* (1984) 76-82.

[28] Y. Toyama, On commutativity of term rewriting systems, *Trans. IECE Japan, J66-D, 12* (1983) 1370-1375, in Japanese.

[29] Y. Toyama, On the Church-Rosser property for the direct sum of term rewriting systems, *J. ACM 34* (1987) 128-143.