# 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 is proposed for testing equivalence in a restricted domain of two given term rewriting systems. By using the Church-Rosser property and the reachability of term rewriting systems, the method allows us to prove equivalence of these systems without the explicit use of induction; this proof usually requires some kind of induction. The method proposed is a general extension of *inductionless induction* methods developed by Musser, Goguen, Huet and Hullot, and allows us to extend *inductionless induction* concepts to not only term rewriting systems with the termination property, but also various reduction systems: term rewriting systems without the termination property, string rewriting systems, graph rewriting systems, combinatory reduction systems, and resolution systems. This method is applied to test equivalence of term rewriting systems, to prove the inductive theorems, and to derive a new term rewriting system from a given system by using equivalence transformation rules.

## 1. Introduction

We consider how to prove equivalence in a restricted domain of two term rewriting systems [5][6] without induction. 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 another system.

We first explain the concept of equivalence in a restricted domain through simple examples. Consider the term rewriting system $R_1$ computing the addition on the set $N$ of natural numbers represented by $0, s(0), s(s(0)), \ldots$;

$$\begin{aligned} R_1 : \; & x + 0 \triangleright x, \\ & x + s(y) \triangleright s(x + y). \end{aligned}$$

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

$$\begin{aligned} R_2 : \; & x + 0 \triangleright x, \\ & x + s(y) \triangleright s(x + y), \\ & x + (y + z) \triangleright (x + y) + z. \end{aligned}$$

Then, $R_2$ can reduce $(M + N) + P$ and $M + (N + P)$ to the same normal form for any terms $M, N, P$, but $R_1$ cannot reduce them unless $M, N, P$ can be reduced to natural numbers. Thus, equivalence of $R_1$ and $R_2$ must be regarded as equivalence in the domain in which terms can be reduced into natural numbers.

We show another example concerning equivalence of recursive programs. Assuming rules for primitive functions, we define the factorial function $f(n) = n!$ by using the term rewriting systems $R_1$ and $R_2$;

$R_1$ : $f(x) \triangleright$ if equal$(x,0)$ then $s(0)$ else $x * f(x - s(0))$,

$R_2$ : $f(0) \triangleright s(0)$,
$f(s(x)) \triangleright s(x) * f(x)$.

Since the rewriting rule of $R_1$ can be infinitely applied to the function symbol $f$, $R_1$ has an infinite reduction sequence starting with the term $f(M)$ for any term $M$. On the other hand, $R_2$ cannot reduce $f(M)$ unless $M$ can be reduced to a natural number. Thus, $R_1$ and $R_2$ generally produce different reduction sequences, although they can reduce the term $f(M)$ to the same result if $M$ can be reduced to a natural number. Therefore, equivalence for the recursive programs may be regarded as equivalence in the restricted domain of term rewriting systems.

Thus, 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, in general, be proved by mere equational reasoning: some kind of induction on the domain is necessary.

In this paper, we present a new very simple method for proving equivalence in a restricted domain of two term rewriting systems without explicit induction. Our approach to this problem was inspired by *inductionless induction* methods developed by Musser [13], Goguen [4], Huet and Hullot [7]. However, our method is more general than their *inductionless induction* methods, and allows us to extend *inductionless induction* concepts to not only term rewriting systems with the termination property [4][7][13], but also various reduction systems: term rewriting systems without the termination property [5][6], string rewriting systems (Thue systems) [2], graph rewriting systems [17], combinatory reduction systems [10], and resolution systems [8][16], etc.

The key idea of our method is that equivalence in a restricted domain can be easily proved by using the Church-Rosser property and the reachability of reduction systems. We first explain this idea in an abstract framework. Simple sufficient conditions for equivalence in a restricted domain of two given abstract reduction systems are shown. Our results are carefully partitioned between abstract properties depending solely on the reduction relation and properties depending on the term structure. We show how one can formally validate equivalence for term rewriting systems by using these abstract results, and how *inductionless induction* methods by [4][7][13] can be naturally extended to our method. Finally, we propose an equivalence transformation technique for term rewriting systems.

## 2. Reduction Systems

We explain notions of reduction systems and give definitions for the following sections. These reduction systems have only an abstract structure, thus they are called abstract reduction systems [5][10].

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., $\rightarrow \subset A \times A$), called a reduction relation. A reduction (starting with $x_0$) in $R$ is a finite or infinite sequence $x_0 \rightarrow x_1 \rightarrow x_2 \rightarrow \ldots$. 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; 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 $R$ has the termination property, 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 $WN(R)$) iff any $x \in A$ has a normal form.

**Definition.** $R = \langle A, \rightarrow \rangle$ has the Church-Rosser property (denoted by $CR(R)$) 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][5][10].

**Proposition 2.1.** Let $R$ have the Church-Rosser property, then,
(1) $\forall x, y \in A[x = y \Rightarrow \exists w \in A, x \xrightarrow{*} w \wedge y \xrightarrow{*} w]$,
(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 $R_1 = \langle A, \underset{1}{\rightarrow} \rangle$ and $R_2 = \langle A, \underset{2}{\rightarrow} \rangle$ be two abstract reduction systems having the same object set $A$, and let $\xrightarrow[i]{*}$, $\underset{i}{=}$ 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, $\underset{1}{=} \subset \underset{2}{=}$ 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 $\underset{1}{\sim} = \underset{2}{\sim}$ in $A' \times A''$ iff $\forall x \in A' \forall y \in A''[x \underset{1}{\sim} y \iff x \underset{2}{\sim} y]$. We write this as $\underset{1}{\sim} = \underset{2}{\sim}$ in $A'$ if $A' = A''$. $A'$ is reachable to $A''$ under $\underset{1}{\sim}$ iff $\forall x \in A' \exists y \in A''[x \underset{1}{\sim} y]$. $A'$ 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 $R_1$, $R_2$ satisfy the following conditions:
(1) $\underset{1}{=} \subset \underset{2}{=}$,
(2) $\underset{1}{=} = \underset{2}{=}$ in $A''$,
(3) $A'$ is reachable to $A''$ 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 $x \underset{1}{=} z \underset{1}{=} w \underset{1}{=} y$. □

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) $\underset{1}{=} \subset \underset{2}{=}$,
(2) $CR(R_2)$, $\underset{1}{\rightarrow} = \underset{2}{\rightarrow}$ in $A''$, and $A''$ is closed under $\underset{2}{\rightarrow}$,
(3) $A'$ is reachable to $A''$ under $\underset{1}{=}$.
Then $\underset{1}{=} = \underset{2}{=}$ in $A'$.

**Proof.** Show 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 $\underset{2}{\rightarrow}$, and Proposition 2.1(1), there exists some $z \in A''$ such that $x \xrightarrow[2]{*} z$ and $y \xrightarrow[2]{*} z$. By using $\underset{1}{\rightarrow} = \underset{2}{\rightarrow}$ in $A''$ and the closed property of $A''$ under $\underset{2}{\rightarrow}$, $x \xrightarrow[1]{*} z$ and $y \xrightarrow[1]{*} z$ can be derived. Therefore $x \underset{1}{=} y$. □

**Theorem 3.2.** Assume the following conditions:

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

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

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

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

**Proof.** Show 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 $x \underset{1}{=} y$. □

**Corollary 3.1.** Assume the conditions:

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

(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 $\underset{1}{\overset{*}{\rightarrow}} = \underset{2}{\overset{*}{\rightarrow}}$ in $A' \times A''$.

**Theorem 3.3.** Assume the following conditions:

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

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

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

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

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

**Corollary 3.2.** Assume the conditions:

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

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

(3) $WN(R_1)$.

Then $\underset{1}{\overset{*}{\rightarrow}} = \underset{2}{\overset{*}{\rightarrow}}$ 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 $A$. However, note that the above abstract results can be applied to not only term rewriting systems, but also 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 [5][6].

Let $F$ be an enumerable set of function symbols denoted by $f, g, h, \ldots$, and let $V$ be an enumerable set of variable symbols denoted by $x, y, z, \ldots$ where $F \cap V = \phi$. By $T(F, V)$ (abbreviated by $T$) 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 substitution $\theta$ is a mapping from a term set $T$ to $T$ 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 $\Box$ called a hole and the set $T(F \cup \{\Box\}, V)$. Then $C \in T(F \cup \{\Box\}, V)$ is called a context on $F$. We use the notation $C[\ ]$ for the context containing precisely one hole, and if $N \in T(F, V)$ then $C[N]$ denotes the result of placing $N$ in the hole of $C[\ ]$. $N$ is called a subterm of $M \equiv C[N]$. Let $N$ be a subterm occurrence of $M$, then, write $N \subset M$.

A rewriting rule on $T$ is a pair $\langle M_l, M_r \rangle$ of terms in $T$ such that $M_l \notin V$ and any variable in $M_r$ also occurs in $M_l$. The notation $\triangleright$ denotes a set of rewriting rules on $T$ and we write $M_l \triangleright M_r$ for $\langle M_l, M_r \rangle \in \triangleright$. The set $\triangleright$ of rewriting rules on $T$ defines a reduction relation $\rightarrow$ on $T$ as follows:
$M \rightarrow N$ iff $M \equiv C[M_l\theta]$, $N \equiv C[M_r\theta]$, and $M_l \triangleright M_r$ for some $M_l, M_r, C[\ ]$, and $\theta$.

**Definition.** A term rewriting system $R$ on $T$ is a reduction system $R = \langle T, \rightarrow \rangle$ such that the reduction relation $\rightarrow$ is defined by a set $\triangleright$ of rewriting rules on $T$.

$R \cup \{M \triangleright N\}$ (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 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 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 $\theta$ [5][11]. Since $M\theta \equiv C[S]\theta \equiv C\theta[P\theta]$, two reductions starting with $M\theta$, i.e., $M\theta \rightarrow C\theta[Q\theta] \equiv C[Q]\theta$ and $M\theta \rightarrow N\theta$, can be obtained by using $P \triangleright Q$ and $M \triangleright N$. Then we say that the pair $\langle C[Q]\theta, N\theta \rangle$ of terms is critical in $R$ [5][6]. 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 pair $\langle N, N \rangle$. If $R$ has no critical pair, then we say that $R$ is nonoverlapping [5][6][11][19].

The following sufficient conditions for the Church-Rosser property are well known [5][6][11].

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

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

For more discussions concerning the Church-Rosser property of term rewriting systems having overlapping or nonlinear rules, see [5][19][21].

There are several sufficient conditions for the reachability of term rewriting systems to hold. However, we will omit all proofs of the reachability in the following examples, because they are mostly technical. For discussions of techniques for proving the reachability, see [7][12][15][18].

## 5. Examples

We now illustrate how to prove 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.

**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 $N$:

$$R_1 : \begin{array}{l} x + 0 \triangleright x, \\ x + s(y) \triangleright s(x + y), \end{array} \qquad R_2 : \begin{array}{l} x + 0 \triangleright x, \\ x + s(y) \triangleright s(x + y), \end{array}$$

$$x + (y + z) \triangleright (x + y) + z.$$

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 must show conditions (1), (2), (3) of Theorem 3.2 for $R_1$ and $R_2$. Since $\underset{1}{\triangleright} \subset \underset{2}{\triangleright}$, condition (1), i.e., $\underset{1}{=} \subset \underset{2}{=}$, is obvious. By using $SN(R_2)$ and Proposition 4.1, $CR(R_2)$ is obtained. Condition (2) holds, since $T(F'') \subset NF_2$. $T(F')$ is reachable to $T(F'')$ under $\underset{1}{=}$. Therefore, $\underset{1}{=} = \underset{2}{=}$ in $T(F')$.

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

**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 $Z_3$:

$$\begin{array}{ll} R_1 : s(s(s(x))) \triangleright x, & R_2 : s(s(s(x))) \triangleright x, \\ \quad x + 0 \triangleright x, & \quad x + 0 \triangleright x, \\ \quad x + s(y) \triangleright s(x + y), & \quad x + s(y) \triangleright s(x + y), \\ & \quad x + (y + z) \triangleright (x + y) + z. \end{array}$$

We will prove that $\underset{1}{=} = \underset{2}{=}$ in $T(F')$ by using Theorem 3.1. Let $A' = T(F'), A'' = T(F'')$. We must show conditions (1), (2), (3) of Theorem 3.1 for $R_1$ and $R_2$. Condition (1), i.e., $\underset{1}{=} \subset \underset{2}{=}$, 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 $T(F'')$ is closed under $\underset{2}{\rightarrow}$. $T(F')$ is reachable to $T(F'')$ 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 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 $d(n) = 2 * n$:

$$\begin{array}{ll} R_1 : d(0) \triangleright 0, & R_2 : d(x) \triangleright \mathrm{if}(x, 0, s(s(d(x - s(0))))), \\ \quad d(s(x)) \triangleright s(s(d(x))), & \quad \mathrm{if}(0, y, z) \triangleright y, \\ & \quad \mathrm{if}(s(x), y, z) \triangleright z, \\ & \quad x - 0 \triangleright x, \\ & \quad s(x) - s(y) \triangleright x - y. \end{array}$$

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 $A' = T(F'), A'' = T(F'')$. We must show 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., $\underset{1}{=} \subset \underset{2}{=}$, is obtained. It is obvious that $R_2$ is 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 that $T(F'') \subset NF_2$. Thus, condition (2) holds. $T(F')$ is reachable to $T(F'')$ under $\underset{1}{=}$. Therefore, $\underset{1}{=} = \underset{2}{=}$ in $T(F')$ holds.

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

## 6. Inductionless Induction

By using Theorem 3.2, an equation whose proof usually requires induction on some data

structure can be proved without the explicit use of induction. In this section, we will explain how to prove an equation with *inductionless induction* method [4][7][9][12][13][14][15][16].

Let $R_1$ be a term rewriting system with reachability from $T(F')$ to $T(F'')$ under $\underset{1}{=}$. 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 be proved by merely equational reasoning: some kind of induction on $T(F')$ usually becomes necessary [4][7][13]. However, we can prove that $M \underset{1}{=} N$ in $T(F')$ by using the following theorem, without induction.

**Theorem 6.1.** Let $R_2 = R_1 \cup \{M \triangleright N\}$. If $R_2$ has the Church-Rosser property and $T(F'') \subset NF_2$, then $M \underset{1}{=} N$ in $T(F')$.

**Proof.** It is obvious that $R_1$ and $R_2$ satisfy conditions (1), (2), (3) of Theorem 3.2 by letting $A' = T(F')$, $A'' = T(F'')$. 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 $d(n) = 2 * n$:

$$
\begin{aligned}
R_1 : \; & h(0) \triangleright 0, \\
& h(s(0)) \triangleright 0, \\
& h(s(s(x))) \triangleright s(h(x)), \\
& d(0) \triangleright 0, \\
& d(s(x)) \triangleright s(s(d(x))).
\end{aligned}
$$

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

When $R_2 = R_1 \cup \{M \triangleright N\}$ does not satisfy the conditions in Theorem 6.1, we may find $R_3$ instead of $R_2$ such that $CR(R_3)$, $T(F'') \subset NF_3$, and $\underset{2}{=} = \underset{3}{=}$ in $T(F')$. If term rewriting systems are strongly normalizing, then the effective search for $R_3$ can be done by using the Knuth-Bendix completion algorithm [11]. Thus, this method allows us to automatically prove inductive theorems.

The original idea of this method was proposed by Musser [13], and has been extended by Goguen [4], Huet and Hullot [7], and others [9][12][14][15]. However, their *inductionless induction* methods have many limitations. In particular, the requirement for the strongly normalizing property [4][7][13][14] (or the strongly normalizing property on equivalence classes of terms if there are associative/commutative laws [9][12][15]) 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, since Theorem 6.1 holds under very weak assumptions, our method allows us to overcome these limitations.

We next show an example in which $R_1$ does not have the strongly normalizing property.

**Example 6.2.**

$$
\begin{aligned}
R_1 : \; & d(x) \triangleright \mathrm{if}(x, 0, s(s(d(x - s(0))))), \\
& h(x) \triangleright \mathrm{if}(x, 0, \mathrm{if}(x - s(0), 0, s(h(x - s(s(0)))))), \\
& \mathrm{if}(0, y, z) \triangleright y, \\
& \mathrm{if}(s(x), y, z) \triangleright z, \\
& x - 0 \triangleright x, \\
& s(x) - s(y) \triangleright x - y.
\end{aligned}
$$

Note that the term rewriting system $R_1$ does not have the strongly normalizing property,

since the first and the second rules in $R_1$ can be infinitely applied to the function symbols $d$ and $h$ respectively.

Let $F' = \{d, h, \text{if}, -, s, 0\}$ and $F'' = \{s, 0\}$. $T(F')$ is reachable to $T(F'')$ under $\underset{1}{=}$. Now, we show that $h(d(x)) \underset{1}{=} x$ in $T(F')$. Take $R_2 = R_1 \cup \{h(d(x)) \triangleright x\}$. 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 $R_2$:

$$
\begin{aligned}
R_3 : \; & d(0) \triangleright 0, \\
& d(s(x)) \triangleright s(s(d(x))), \\
& h(0) \triangleright 0, \\
& h(s(0)) \triangleright 0, \\
& h(s(s(x))) \triangleright s(h(x)), \\
& \text{if}(0, y, z) \triangleright y, \\
& \text{if}(s(x), y, z) \triangleright z, \\
& x - 0 \triangleright x, \\
& s(x) - s(y) \triangleright x - y, \\
& h(d(x)) \triangleright x.
\end{aligned}
$$

Then, $\underset{2}{=} = \underset{3}{=}$ in $T(F')$ can be proved 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, $T(F'') \subset NF_3$. Hence, $R_3$ satisfies the conditions in Theorem 6.1. Therefore, $h(d(x)) \underset{1}{=} x$ in $T(F')$. □

## 7. Equivalence Transformation Technique

In this section, we propose the equivalence transformation rules for term rewriting systems. We show that equivalence of term rewriting systems, to which it is difficult to apply Theorem 3.2 and 3.3 directly, can be easily proved by an equivalence transformation technique.

Let $R_0 = \langle T(F,V), \underset{0}{\rightarrow} \rangle$ with $\underset{0}{\triangleright}$, and let $F'$ be a subset of $F$. Now, we give the equivalence transformation rules in $T(F')$ for $R_0$. Let $F_0$ be the union of the set $F'$ and the set of all function symbols appearing in the rewriting rules of $R_0$. Then, we transform $R_n = \langle T(F,V), \underset{n}{\rightarrow} \rangle$ with $\underset{n}{\triangleright}$ $(n \geq 0)$ to $R_{n+1} = \langle T(F,V), \underset{n+1}{\rightarrow} \rangle$ with $\underset{n+1}{\triangleright}$ by using the following rules:

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

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

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

**Remarks.** The above three rules are a natural extension of the program transformation rules suggested by Burstall and Darlington [3]. Note that their program transformations can be seen as special cases of equivalence transformations for term rewriting systems in restricted domains. Thus, we can give formal proofs to correctness of program transformations by the technique developed in this section [20].

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

**Theorem 7.1.** Let $R_0 \overset{*}{\Rightarrow} R_n$, where $R_0$ is a linear system and $CR(R_0)$. Let $F'' \subset F'$ and $T(F'') \subset NF_0$. Assume that $T(F')$ is reachable to $T(F'')$ under $\underset{0}{=}$ and under $\underset{n}{=}$ (resp. under $\underset{0}{\rightarrow}$ and under $\underset{n}{\rightarrow}$). Then $\underset{0}{=} = \underset{n}{=}$ in $T(F')$ (resp. $\overset{*}{\underset{0}{\rightarrow}} = \overset{*}{\underset{n}{\rightarrow}}$ in $T(F') \times T(F'')$).

**Proof.** See [20]. □

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

$$
\begin{array}{ll}
R_1 : f(0) \triangleright 0, & R_2 : f(0) \triangleright 0, \\
\quad f(s(x)) \triangleright s(x) + f(x), & \quad f(s(x)) \triangleright g(x, s(x)), \\
\quad x + 0 \triangleright x, & \quad g(0, y) \triangleright y, \\
\quad x + s(y) \triangleright s(x + y), & \quad g(s(x), y) \triangleright g(x, y + s(x)), \\
 & \quad x + 0 \triangleright x, \\
 & \quad x + s(y) \triangleright s(x + y).
\end{array}
$$

Let $F' = \{f, +, s, 0\}$ and $F'' = \{s, 0\}$. By using the equivalence transformation rules, we will show that $\underset{1}{=} = \underset{2}{=}$ in $T(F')$. To transform $R_1$ to $R_2$, we first add the associative law for $+$ to $R_1$: take $R_3 = R_1 \cup \{x + (y + z) \triangleright (x + y) + z\}$. Then $R_3 \Rightarrow R_1$ by rule$(E)$. From Proposition 4.1, $CR(R_3)$. Clearly $T(F'') \subset NF_3$. $T(F')$ is reachable to $T(F'')$ 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. By using rule$(D)$, 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 $f(s(x)) \underset{4}{=} g(x, s(x))$, $g(0, y) \underset{4}{=} y$, and

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

By using 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 using 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 to $T(F'')$ 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 $T(F')$.

Note that it is also possible to prove $\underset{1}{\overset{*}{\rightarrow}} = \underset{2}{\overset{*}{\rightarrow}}$ in $T(F') \times T(F'')$ by this transformation technique. □

## 8. Conclusion

In this paper, we have proposed a new simple method to prove equivalence in a restricted domain for reduction systems without the explicit use of induction. The key idea is that equivalence in the restricted domain can be easily tested by using the Church-Rosser property and the reachability of reduction systems. We have shown that this technique can be effectively applied to test the equality of term rewriting systems and to prove the inductive theorems without induction. We believe firmly that our method provides us with systematic means of proving equivalence which arises in various formal systems: automated theorem proving, program transformation, program verification, and semantics of abstract data types.

## Acknowledgments

The author is grateful to Hirofumi Katsuno, Shigeki Goto, and other members of the First Research Section for their suggestions.

## References


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

[2] Book,R.“ Confluent and other types of Thue systems”, J.ACM, Vol.29 (1982), pp.171-182.

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

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

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

[6] 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.

[7] 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.

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

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

[10] Klop,J.W.“ Combinatory reduction systems”, Dissertation, Univ. of Utrecht (1980).

[11] 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.

[12] Kounalis,E.“Completeness in data type specifications”, Lecture Notes in Comput. Sci., Vol.204, Springer-Verlag (1985), pp.348-362.

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

[14] Nipkow,T. and Weikum,G.“A decidability results about sufficient-completeness of axiomatically specified abstract data type”, Lecture Notes in Comput. Sci., Vol.145, Springer-Verlag (1983), pp.257-267.

[15] Paul,E.“ Proof by induction in equational theories with relations between constructors”, 9th Colloquium on trees in algebra and programming, Ed. Courcelle,B., Cambridge University Press (1984), pp.211-225.

[16] Paul,E.“On solving the equality problem in theories defined by Horn clauses”, Lecture Notes in Comput. Sci., Vol.204, Springer-Verlag (1985), pp.363-377.

[17] Raoult,J.C.“On graph rewriting”, Theoretical Comput. Sci. Vol.32 (1984), pp.1-24.

[18] Thiel,J.J.“ Stop losing sleep over incomplete data type specifications”, Proc. 11th ACM Sympo. Principles of programming languages (1984), pp.76-82.

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

[20] Toyama,Y.“ On equivalence transformations for term rewriting systems”, RIMS Symposia on Software Science and Engineering, Kyoto (1984), Lecture Notes in Comput. Sci., Vol.220, Springer-Verlag (1986), pp.44-61.

[21] Toyama,Y.“ On the Church-Rosser property for the direct sum of term rewriting systems”, to appear in J.ACM.