九州大学学術情報リポジトリ
Kyushu University Institutional Repository

A GRAPH REDUCTION SYSTEM USING GRAPH TERMS

Mizoguchi, Yoshihiro
Dept. of Control Eng. and Sci., Kyushu Inst. of Tech.

https://doi.org/10.5109/13421

出版情報：Bulletin of informatics and cybernetics. 25 (1/2), pp.27-40, 1992-03. Research Association of Statistical Sciences
バージョン：
権利関係：

KYUSHU UNIVERSITY

Bulletin of Informatics and Cybernetics, Vol. 25, No. 1~2, 1992

A GRAPH REDUCTION SYSTEM USING GRAPH TERMS

By

Yoshihiro MIZOGUCHI*

Abstract

A new kind of computing system, a graph reduction system, is introduced using the original notation of graphs. Every graphs and reduction procedures are defined just like as term rewriting systems. Several properties of graphs and graph reductions are studied from the viewpoint of our graph reduction system. Especially, we investigate the ability of parallel reductions and compare parallel reductions with sequential reductions. We formalize the concepts of parallel reductions and sequentially simulatable parallel reductions. Sufficient conditions for reduction rules which guarantees the consistency of parallel reductions and sequential reductions are given. It is also showed that every parallel reduction preserves correctness of solutions for the graph reduction system solving equations of regular expressions.

1. Introduction

There are several researches about graph rewrittings of graphs which express terms sharing subterms [1] [9] [10]. They define expressions of terms sharing subterms and rewriting rules as formal languages. Operational semantics of these languages for actual graph rewritings are defined. These systems have enough power to execute rewritings of terms sharing subterms but are not able to treat complicated looped general graphs.

We consider the general graph rewritings and rewriting rules formally. Our rewritings are based on formal symbolical substitutions which are independent to interpretation of graphs. Using the graph terms, operations such as matching and reducing are formally defined. We are interested in transition of symbolical graphs rather than the languages of graph rewriting system. For example, in our framework, a parallel reduction without any restriction generates an unbelievable graph, but we can formally calculate the rewritings using our rewriting definitions. We analyze that what kind of rewriting rules generates a reasonable graphs without any restriction of application order and parallel execution controls.

As a general framework of graph rewriting, there is a famous Ehrig's specification of graph grammar using fast production rules [7] [8]. But it sometimes inconvenient to simulate term rewritings with sharing subterms. If rewriting rules matches to a graph, Ehrig's rewriting method may not be applicable when it does not satisfy the gluing

* Dept. of Control Eng. and Sci., Kyushu Inst. of Tech., Iizuka 820, Japan.
(Email: ym@ces.kyutech.ac.jp)

conditions. Since our framework is based on the graph rewriting using a different graph structure [15], we can apply our rewriting method for any matching graphs.

When we consider the parallel execution of graph reductions, there are two cases. One is the execution of reductions applying to completely separated parts of a graph. The executions of these reductions are not influenced each other. So we can get the same result by executing each reduction sequentially. We call the execution of these reductions as a sequentially simulatable parallel reduction. The other case is the parallel execution of reductions which have overlapped reading regions. When we apply a rule to a graph, the part of a graph which matches to a rule is called reading region. The part of a graph which is modified by a rule after matching is defined as writing region. Our graph reduction system does not allow the parallel execution of graph reductions which have overlapped writing regions, but our system can execute reductions which have only overlapped reading regions. The result of the execution of overlapped reductions is not always obtained from the sequentially execution of the reductions. In some sense, it is real parallel execution which have overlapped reading regions. The behaviors of parallel execution is so complicated that it is hard to consider the properties of the parallel reductions.

Any parallel reduction based on Ehrig's graph grammar is always sequential simulatable by the Parallelism Theorem [7]. We consider, in some sense, the gluing condition for any reduction is a sufficient condition for sequential simulatable reductions. The known graph rewriting system [1] [9] [10] always execute sequential simulatable parallel reductions using the control of the order of applications.

The main aim of us is to investigate the properties of the real parallel reductions. The symbolical notation of graphs and reduction rules is helpful to define a reading region and a writing region precisely. These facts contribute to clear the concept of sequentially simulatable parallel reductions. We have a sufficient condition that holds parallel reductions are sequentially simulatable. For a meaningful example of non sequentially simulatable reductions, we show a graph reduction system which solves equations of regular expressions. The reduction rules are not sequentially simulatable, but it is shown that every parallel reduction leads to the right solution of the equations.

In Section 2, we define new notations for graphs and graph terms. The interpretation of graph terms is also introduce [15]. In Section 3, we define a graph reduction system. Reduction rules and a matching procedure are defined. We define not only simple reduction but also a parallel reduction. The concept of sequentially simulatable parallel reduction is defined. Some properties about sequentially simulatable reduction are discussed. Turner's SK-reduction machine [18] guarantees every parallel reduction induce a correct result. We define a graph rewriting rules corresponding to SK-reductions, we reconfirm the fact of parallel reductions in our framework.

In Section 4, we define a graph reduction system which solves equations of regular expressions. The ability of parallel execution of the system is discussed. The system has the special property of parallel executions. A parallel reduction, even if which is not sequentially simulatable, guarantees the consistency of a solution of equations.

2. Symbolic Graphs and Graph Terms

We first review the elementary notations of relational calculus. Let A, B and C be sets. When α is a subset of $A \times B$, we call α is a *relation* from A to B and denote it by $\alpha : A \to B$. For relations $\alpha : A \to B$ and $\beta : B \to C$, we define a *composite relation* $\alpha \cdot \beta : A \to C$ by $\alpha \cdot \beta := \{(a, c) \in A \times C \mid (a, b) \in \alpha, (b, c) \in \beta$ for some $b \in B\}$. For relation $\alpha : A \to B$, we define the *inverse relation* $\alpha^{\#} : B \to A$ by $\alpha^{\#} = \{(b, a) \in B \times A \mid (a, b) \in \alpha\}$. We identify a function $f : A \to B$ with a relation $\{(a, f(a)) \in A \times B \mid a \in A\}$ (the graph of f). A function from a set X to one point set $1 = \{*\}$ is denoted by $\Omega_X : X \to 1$. We define a subset $\mathrm{dom}(\alpha)$ of A for a relation $\alpha : A \to B$ by $\mathrm{dom}(\alpha) = \{a \in A \mid \exists b \in B : (a, b) \in \alpha\}$ and a relation $d(\alpha) : A \to A$ by $d(\alpha) = \{(a, a) \in A \times A \mid a \in \mathrm{dom}(\alpha)\}$. For two relations $\alpha, \beta : A \to B$, we define $\alpha \cup \beta$ and $\alpha \cap \beta$ by the set union and intersection, respectively. We denote the category of sets and functions by **Set**, and the category of sets and partial functions by **Pfn**.

Next we summarize some properties related to pushout constructions and graph rewriting squares. We omit details which described in [15] and [16].

LEMMA 2.1. *Let A, B, C and X be sets with $B \subset A$ and $B \subset C$, $i : B \to A$ and $j : B \to C$ inclusion functions, $g : A \to X$ a function. If $gg^{\#}\Omega_A \subset ii^{\#}\Omega_A$, $X \cap (C - B) = \emptyset$ and the square*

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "g"'] & C \arrow[d, "h"] \\ X \arrow[r, "k"'] & Y \end{tikzcd}$$

is a pushout in the category **Pfn**, *then $Y \cong (X - g(A)) \cup g(B) \cup (C - B)$, where $f = i^{\#}j$.*

Let E be a *set of labels for edges. An (E-labeled) graph* $< A, \alpha >$ is a pair of a set A and a collection $\alpha = \{\alpha_e : A \to A \mid e \in E\}$ of relations indexed by E. *A graph morphism* f of a graph $\langle A, \alpha \rangle$ into a graph $\langle B, \beta \rangle$, denoted by $f : \langle A, \alpha \rangle \to \langle B, \beta \rangle$, is a partial function $f : A \to B$ satisfying $d(f)\alpha_e f \subset f\beta_e$ for all $e \in E$.

DEFINITION 2.2. *Let $\langle A, \alpha \rangle$, $\langle B, \beta \rangle$, $\langle X, \xi \rangle$ and $\langle Y, \eta \rangle$ be graphs and $f : A \to B$, $g : A \to X$, $h : B \to Y$ and $k : X \to Y$ partial functions. The square*

$$\begin{tikzcd} \langle A, \alpha \rangle \arrow[r, "f"] \arrow[d, "g"'] & \langle B, \beta \rangle \arrow[d, "h"] \\ \langle X, \xi \rangle \arrow[r, "k"'] & \langle Y, \eta \rangle \end{tikzcd}$$

is called the graph rewriting square *if it satisfies following conditions*:

(1) *$g : A \to X$ is a total graph morphism,*

(2) *The square*

$$\begin{tikzcd} A \arrow[r, "f"] \arrow[d, "g"'] & B \arrow[d, "h"] \\ X \arrow[r, "k"'] & Y \end{tikzcd}$$

is a pushout in the category **Pfn**,

(3) $\eta_e = k^{\#}(\xi_e - g^{\#}\alpha_e g)k \cup h^{\#}\beta_e h$ *for all* $e \in E$.

Next we define the graph as a symbolical objects. A symbolical graph is defined a finite set of symbolical graph terms. The interpretation of these symbolical objects correspond to E-labeled graphs defined above. Let S be a *set of identifiers*.

A *graph term* is defined as follow:

(1) If $s \in S$, then s and $s[]$ are graph terms.

(2) If $s_0 \in S$, $e_1, \ldots, e_n \in E$ and $t_1, \ldots, t_n$ are graph terms, then $s_0[e_1 : t_1, \ldots, e_n : t_n]$ is a graph term.

A graph term is called *simple*, if it is defined by (1) or defined by (2) when every t_i is an element of S.

DEFINITION 2.3. *For a graph term t, we define the root identifier $rt(t)$, the set $Id(t)$ of identifiers, the set $Int(t)$ of internal identifiers, the set ∂t of leaf identifiers, and the incidence relation $\xi_e(t) : Id(t) \to Id(t)$ for each $e \in E$.*

(1) *If $s \in S$, then $rt(s) = s$, $Id(s) = \{s\}$, $Int(s) = \phi$, $\partial s = \{s\}$, $\xi_e(s) = \phi$, $rt(s[]) = s$, $Id(s[]) = \{s\}$, $Int(s[]) = \{s\}$, $\partial s[] = \{s\}$, and $\xi_e(s[]) = \phi$.*

(2) *If $s_0 \in S$, $e_1, \ldots, e_n \in E$ and $t_1, \ldots, t_n$ are graph terms, then*

$$rt(s_0[e_1 : t_1, \ldots, e_n : t_n]) = s_0$$

$$Id(s_0[e_1 : t_1, \ldots, e_n : t_n]) = \{s_0\} \cup \bigcup_{i=1}^{n} Id(t_i),$$

$$Int(s_0[e_1 : t_1, \ldots, e_n : t_n]) = \{s_0\} \cup \bigcup_{i=1}^{n} Int(t_i),$$

$$\partial s_0[e_1 : t_1, \ldots, e_n : t_n] = \bigcup_{i=1}^{n} \partial t_i,$$

$$\xi_e(s_0[e_1 : t_1, \ldots, e_n : t_n]) = \{(s_0, rt(t_i)) \mid e = e_i\} \cup \bigcup_{i=1}^{n} \xi_e(t_i).$$

For a finite set X of a graph terms, the set $\{rt(x) \mid x \in X\}$ of all root identifiers of X is denoted by $Rt(X)$. An interpretation $|X|$ of a finite set X of graph terms is an E-labbeled graph $|X| = \langle Rt(X), \xi(X) \rangle$, where $\xi(X)_e = \cup_{t \in X} \xi_e(t)$.

A finite set G of simple graph terms is a *symbolic graph* if it satisfies following conditions (1), (2) and (3):

(1) If $x \neq y$ then $rt(x) \neq rt(y)$ for any $x, y \in G$,

(2) If $x = s_0[e_1 : s_1, e_2 : s_2, \ldots, e_n : s_n] \in G$, there exists $y_i \in G$ such that $s_i = rt(y_i)$ for all $i = 1, \ldots, n$,

(3) If $x = s_0[e_1 : s_1, e_2 : s_2, \ldots, e_n : s_n] \in G$, then $e_i \neq e_j$ or $s_i \neq s_j$ for any $i \neq j$.

Since every element of a symbolic graph G has a different identifier, we can

uniquely define the simple graph term $\exp_G(s)$ for $s \in G$ satisfying $\exp_G(rt(x)) = x$ for all $x \in G$. A *pointed symbolic graph* is a pair $\langle rt(x), G\rangle$ of a symbolic graph G and an element $rt(x) \in S$ for some $x \in G$.

DEFINITION 2.4. *For a graph term t we define a finite set $G(t)$ of graph terms as follows*:

(1) *If $s \in S$*, then $G(s) = \{s\}$ *and* $G(s[]) = \{s[]\}$.

(2) *If $s_0 \in S$, $e_1, \ldots, e_n \in E$ and $t_1, \ldots, t_n$ are graph terms, then*

$$G(s_0[e_1 : t_1, \ldots, e_n : t_n]) = \{s_0[e_1 : rt(t_1), \ldots, e_n : rt(t_n)]\} \cup \bigcup_{i=1}^{n} G(t_i)$$

For any graph term t, a finite set $G(t)$ of graph terms is not always a symbolic graphs. So we restrict some more conditions for the graph term.

We define a *strict (graph) term* as follows:

(1) If $s \in S$, then s is a strict term,

(2) If $e_1, \ldots, e_n \in E$, $s_0 \in S - \cup_{i=1}^{n} Int(t_i)$ and $t_1, \ldots, t_n$ are strict terms such that $Int(t_i) \cap Int(t_j) = \phi$ for $i \neq j$, then $s_0[(e_1 : t_1), \ldots, (e_n : t_n)]$ is a strict term.

We note that if t is a strict term, then $G(t)$ is a symbolic graph.

DEFINITION 2.5. *For a graph term t and a function $g : A \to B$ such that $Id(t) \subset A \cup S$ and $B \subset S$, we define a graph term $g^*(t)$ as follows*:

(1) *If $s \in S$, then $g^*(s) = g(s)$ and $g^*(s[]) = g(s)[]$.*

(2) *If $s_0 \in S$, $e_1, \ldots, e_n \in E$ and $t_1, \ldots, t_n$ are graph terms, then*

$$g^*(s_0[(e_1 : t_1), \ldots, (e_n : t_n)]) = g(s_0)[(e_1 : g(t_1)), \ldots, (e_n : g(t_n))].$$

The next proposition guarantees that we can construct the rewriting graphs using a symbolical substitution of graph terms. The element of the set ∂t operates a variables in the rule of term rewritings.

PROPOSITION 2.6. *Let t be a strict term and let B, C and X be symbolic graphs with $B \subset G(t)$ and $Rt(B) \subset Rt(C)$, $i : Rt(B) \to Rt(G(t))$, $j : Rt(B) \to Rt(C)$ inclusion functions and $g : Rt(G(t)) \to Rt(X)$ a function. If $\exp_{G(g^*(t))}(g^*(Int(t))) \subset X$, $\exp_{G(t)}(\partial t) \subset B$, $gg^{\#}\Omega_{Rt(G(t))} \subset ii^{\#}\Omega_{Rt(G(t))}$, $g(Rt(X)) \cap (Rt(C) - Rt(B)) = \emptyset$ and the square*

$$\begin{array}{ccc} |G(t)| & \xrightarrow{f} & |C| \\ {\scriptstyle g}\downarrow & & \downarrow{\scriptstyle h} \\ |X| & \xrightarrow[k]{} & \langle H, \eta\rangle \end{array}$$

is a graph rewriting square, then

$$\langle H, \eta\rangle \cong |(G - \exp_{G(g(t))}(Int(t))) \cup \exp_{g(B)}(g(Rt(B) \cap Int(t))) \cup \exp_C(Rt(C) - Rt(B))|,$$

where $f = i^{\#}j$. ∎

We note that if $B = G(t)$ then $gg^{\#}\Omega_{Rt(G(t))} \subset ii^{\#}\Omega_{Rt(G(t))}$. We only treat the case $B = G(t)$ in the graph reduction system defined in Section 3.

DEFINITION 2.7. *Let G be a symbolic graph. We define a subset $P_G(x, y)$ of E^* for $x, y \in Rt(G)$ as follows:*

(1) $\varepsilon \in Path_G(s, s)$ *for any* $s \in Rt(G)$,

(2) $e_i \in Path_G(s, s_i)$ $(i = 1, 2, \ldots, n)$ *for a simple graph term* $s[e_1 : s_1, e_2 : s_2, \ldots, e_n : s_n] \in G$.

(3) *If* $a \in \mathrm{Path}_G(x, s)$ *and* $b \in Path_G(s, y)$ *then* $ab \in Path_G(x, y)$ *for any* $s \in Rt(G)$.

We call an elment of $P_G(x, y)$ a path from x to y.

DEFINITION 2.8. *A pointed symbolic graph $\langle s, G\rangle$ is reachable if $Path_G(s, x) \neq \phi$ for any $x \in Rt(G)$.*

PROPOSITION 2.9. *If a pointed symbolic graph $\langle s, G\rangle$ is reachable then there exists a g-term t satisfying $SG(t) = G$ and $Rt(t) = s$.* ■

EXAMPLE 2.10. *Let $E = \{a, b\}$ and $S = N$. Then $\{1[a:2, b:2], 2[a:1, b:2]\}$ is a symbolic graph (cf. Fig. 1).*

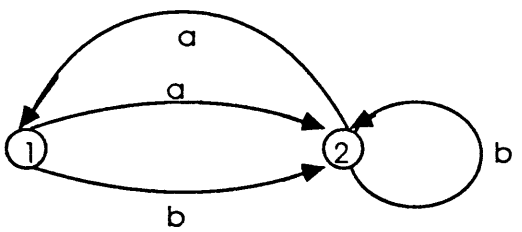

Figure 1: *An example of graph*

EXAMPLE 2.11. *Let E be a one point set $\{*\}$ and $S = N$(a set of natural numbers). We omit to denote an element of E.*

- $G = \{0[1, 2], 1[2], 2[]\}$ *is a symbolic graph.*
- $Path_G(0, 1) = \{*\}$, $Path_G(0, 2) = \{*, **\}$, $Path_G(1, 0) = \emptyset$ *and* $Path_G(1, 2) = \{*\}$.
- $\langle 0, G\rangle$ *is reachable.* $\langle 1, G\rangle$ *is not reachable.*
- $Int(1[2]) = \{1\}$, $Rt(1[2]) = \{1, 2\}$ *and* $G(1[2]) = \{1[2]\}$.
- $t = 0[1[2], 2[]]$ *is a strict term.* $G(t) = G$ *and* $rt(t) = s$.
- *graph term* $t = 0[1[2[]], 2[]]$ *is not strict but* $G(t) = G$.
- *graph term* $u = 0[1[2], 2]$ *is strict and* $|t| = |u|$.
- *A set* $H = \{0[1], 0[2], 1[2]\}$ *of simple graph terms is not a symbolic graph and* $|H| = |\{0[1], 1[]\}|$.

EXAMPLE 2.12. *General terms are expressed by graph term or symbolic graph as follows. We assume $S = N$. Let $E = N + F + C$ where F and C are a set of function symbols and a set of constant symbols, respectively. A constant(function) a is expressed $\cdot \xrightarrow{0} \cdot \xrightarrow{a}$, because it is convenient to share it. We omit denoting a label in N when it is clear from the position of the term in the expressions. For example,* $0[x, y]$ *and* $0[3[f:4],5]$ *means* $0[1:x, 2:y]$ *and* $0[1:3[f:4], 2:5]$, *respectively.*

Term $f(b, g(b, c))$ is expressed by $0[1[f:4], 2[9[b:11]], 3[5[g:8], 6[9], 7[10[c:12]]]]$.

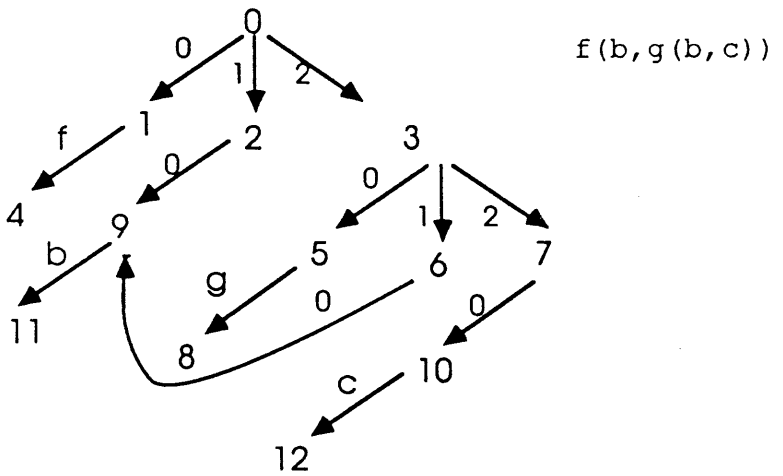

Figure 2: *An element of g-term expressing a term*

3. Graph Reduction System

A function $\sigma: D \to S$ is called an *assignment* for a strict term t if $Rt(G(t)) \subset D \subset S$. A procedure *Match* is defined for an assignment σ for t and a pointed symbolic graph $\langle s, G\rangle$, as follows:

Procedure *Match*
Input: a strict term t,
a pointed graph $\langle s, G\rangle$,
and an assignment σ for t.
Case t **of**
$t = s$ $(s \in S)$:
{**If** $s = \sigma(t)$ **Then** success **Else** failure}
$t = s[]$ $(s \in S)$:
{**If** $\exp_G(s) = \sigma(t)[]$ **Then** success **Else** failure}
$t = s_0[e_1 : t_1, \ldots, e_n : t_n]$:
{**If** $\exp_G(s) = \sigma(t_0)[e_1 : s_1, \ldots, e_n : s_n]$ and
Match$(t_i, \langle s_i, G\rangle, \sigma)$ succeeds
for any $i = 1, \ldots, n$.
Then success **Else** failure}

When the procedure $Match(t, \sigma, \langle s, G\rangle)$ succeeds, we say a strict term t matches to a pointed symbolic graph $\langle s, G\rangle$ by the assignment σ. In this case, we can consider $\sigma(t) \subset G$.

EXAMPLE 3.1. *Let E be a one point set and $S = N$. We omit to denote an element of E. We consider a graph term $t = 0[1[2]]$, a symbolic graph $G = \{0[1], 1[0]\}$ and an assignment $\sigma = \{(0, 0), (1, 1), (2, 0)\}$ for t. Then all of $Match(2, \sigma, \langle 0, G\rangle)$, $Match(1[2], \sigma, \langle 1, G\rangle$ and $Match(t, \sigma, \langle 0, G\rangle)$ succeed.*

DEFINITION 3.2. *A* reduction rule *is a pair $T = \langle t_0, \{t_1, \ldots, t_n\}\rangle$* of a strict term t_0 *and a non empty finite set of simple graph terms $\{t_1, \ldots, t_n\}$ which satisfies $Rt(t_i) \notin \partial t_0$ for $i = 1, \ldots, n$.*

Let $T = \langle t_0, T_0\rangle$ be a reduction rule. For a graph G and an assignment σ, if the conditions

(1) The procedure $Match(t_0, \sigma, \langle s, G\rangle)$ succeeds for an element $s \in G$.

(2) $rt(\sigma(t_i)) \neq rt(\sigma(t_j))$ for any $t_i \neq t_j$ $(t_i, t_j \in T_0)$.

(3) $\sigma(rt(t_i)) \notin Rt(G)$ for any $rt(t_i) \notin Int(t_0)$.

satisfies then we can construct a symbolic graph

$$H = (G - \{\exp_G(Rt(\sigma(t_i)))|i = 1, \ldots, n\}) \cup \{\sigma(t_i)|i = 1, \ldots, n\}.$$

We say the graph G is reduced to the graph H and denote $\langle G, s\rangle \rightarrow_{T/\sigma} H$. We abbreviate $G \rightarrow_{T/\sigma} H$ when we need not concerning s.

Let $t = t_0$, $C = (G(t) - \{\exp_{G(t)}(rt(t_i))|i = 1, \ldots, n\}) \cup \{t_i|i = 1, \ldots, n\}$ and $B = (G(t) - \{\exp_{G(t)}(rt(t_i))|i = 1, \ldots, n\}) \cup \{t_i|rt(t_i) \in Rt(G(t)), i = 1, \ldots, n\}$. The above conditions correspond to that of Proposition 2.6. So the interpretation graph $|H|$ is uniquely determined in the category of graphs.

We define the *reading region* $Rd(G \rightarrow_T H) = \sigma(\mathrm{Int}(t_0))$ and the *writing region* $Wt(G \rightarrow_T H) = \{\sigma(Rt(t_i))|i = 1, \ldots, n\}$.

DEFINITION 3.3. A graph reduction system *is a triple $\langle S, E, R\rangle$, where S is a set of identifiers, E is a label set for edges, and R is a set of reduction rules.*

We note that the set R of reduction rules may be an infinite set.

Next, we define the parallel reductions. Let $T^j = \langle t_0^j, T_0^j\rangle$ $(j = 1, \ldots, m)$ are reduction rules. For a graph G if the conditions

(1) G is reducible to the graph H^j by reduction rules T^j $(\langle G, s^j\rangle \rightarrow_{T^j/\sigma^j} H^j)$.

(2) For any $j_1 \neq j_2$, $Wt(G \rightarrow_{T^{j_1}/\sigma^{j_1}} H^{j_1}) \cap Wt(G \rightarrow_{T^{j_2}/\sigma^{j_2}} H^{j_2}) = \phi$ $(j_1, j_2 = 1, \ldots, m)$.

satisfies then we define a symbolic graph

$$H = G - (\cup_j\{\exp_G(Rt(\sigma^j(t_i^j)))|i = 1, \ldots, n_j\}) \cup (\cup_j\{\sigma(t_i^j)|i = 1, \ldots, n_j\}).$$

We say that the graph G is parallel reduced to the graph H and denote $G \rightarrow_{\{T^1, \ldots, T^m\}} H$.

Roughly speaking, the condition of the parallel reduction is equivalent to the condition of the single reduction by a rule $T = \langle t_0, T_0\rangle$ where $t_0 = \cup_j t_0^j$ and $T_0 = \cup_j T_0^j$. The statement is not perfectly exact, because t_0 may not be a strict term.

DEFINITION 3.4. *A parallel reduction* $G \rightarrow_{\{T^1, \ldots, T^m\}} H$ *is* sequentially simulatable *iff there exists an bijection* $\tau: \{1, \ldots, m\} \rightarrow \{1, \ldots, m\}$ *and graphs* $G_1, \ldots, G_{m-1}$ *such that* $G \rightarrow_{T^{\tau(1)}} G_1 \rightarrow_{T^{\tau(2)}} \ldots \rightarrow_{T^{\tau(m-1)}} G_{m-1} \rightarrow_{T^{\tau(m)}} H$.

PROPOSITION 3.5. *For reduction rules* $T^j = \langle t_0^j, \{t_1^j, \ldots, t_{nj}^j\}\rangle$ $(j = 1, \ldots, m)$, *the parallel reduction* $G \rightarrow_{\{T^1, \ldots, T^m\}} H$ *is sequentially simulatable, if* $Int(t_0^{j1}) \cap Int(t_0^{j2}) = \emptyset$ *for any* $j_1 \neq j_2$.

PROOF. We show the special case of $m = 2$. Let $H_j = \{\sigma^j(t_i^j) | i = 1, \ldots, n_j\}$ and $E_j = \{\exp_G(rt((\sigma^j(t_i^j))) | i = 1, \ldots, n_j\}$. Since $Wt(G \rightarrow_{T^1} H^1) \cap Wt(G \rightarrow_{T^2} H^2) = \phi$, we have $E_2 \cap H_1 = \phi$ then $H_1 - E_2 = H_1$.

$$\begin{aligned} H &= G - (E_1 \cup E_2) \cup (H_1 \cup H_2) \\ &= (G - E_1 - E_2) \cup H_1 \cup H_2 \\ &= (G - E_1 - E_2) \cup (H_1 - E_2) \cup H_2 \\ &= ((G - E_1) \cup H_1) - E_2 \cup H_2 \end{aligned}$$

Let $G_1 = G - E_1 \cup H_1$. Since $Int(t_0^1) \cap Int(t_0^1) = \phi$ and $Match(t_0^2, \sigma^2, \langle s^2, G\rangle)$ succeeds, the procedure $Match\ (t_0^2, \sigma^2, \langle s^2, G_1\rangle)$ succeeds. So we obtain $G \rightarrow_{T^1} G_1 \rightarrow_{T^2} H$.

The cases of $m \geq 2$ are similarly showed. ■

EXAMPLE 3.6. *Let* $E = \{1, 2, S, K, I, B, C, Y, P\}$ *and* $S = N \cup \{f, g, h, x, y\}$. *We omit denoting* 1, 2 *of* E *which it is clear from the position of the term in the expressions. For example,* $0[x, y]$ *and* $0[3[S:4], f]$ *means* $0[1:x, 2:y]$ *and* $0[1:3[S:4], 2:f]$, *respectively. We define rewriting rules as follows. We write* t *and* t_i*'s separately for a reduction rule* $(t, \{t_1, \ldots, t_n\})$.

(1) *Rules for* $Sfgx \rightarrow fx(gx)$:

$$\begin{aligned} t &= 0[1[2[3[S:4], f], g], x] \\ t_1 &= 0[1, 2] \\ t_2 &= 1[f, x] \\ t_3 &= 2[g, x] \end{aligned}$$

(2) *Rules for* $Kxy \rightarrow x$:

$$\begin{aligned} t &= 0[1[2[3[K:4], x], y], z] \\ t_1 &= 0[x, z] \end{aligned}$$

(3) *Rule for* $Ix \rightarrow x$:

$$\begin{aligned} t &= 0[1[2[I:3], x], y] \\ t_1 &= 0[x, y] \end{aligned}$$

(4) *Rules for* $Bfgx \rightarrow f(gx)$:

$$\begin{aligned} t &= 0[1[2[3[B:4], f], g], x] \\ t_1 &= 0[f, 1] \\ t_2 &= 1[g, x] \end{aligned}$$

(5) *Rules for Cfgx → fxg*:

$$\begin{aligned} t &= 0[1[2[3[C:4], f], g], x] \\ t_1 &= 0[1, g] \\ t_2 &= 1[f, x] \end{aligned}$$

(6) *Rule for Yh*:

$$\begin{aligned} t &= 0[1[Y:2], h] \\ t_1 &= 0[h, 0] \end{aligned}$$

PROPOSITION 3.7. *Every parallel reduction of Example 3.6 is sequential simulatable.*

PROOF. It is trivial that for any matching of rules T and T', reading regions are disjoint. So it always holds the condition of Proposition 3.5. ■

4. Calculation of Regular Expressions

In this section, we construct a graph reduction system which solves equations of regular expressions. Let A be an alphabet set, E set of all regular expressions over A and $S = \{F, x, y, x_i, y_i (i = 1, \ldots)\}$. This symbol F express the final state of an automaton graph which is defined later. We put $L_G(s) = Path_G(s, F)$.

DEFINITION 4.1. *A symbolic graph G is an automaton graph iff G contains a simple graph from F.*

PROPOSITION 4.2. *For any element* $s[a_1:s_1, a_2:s_2, \ldots, a_n:s_n] \in G$, $L_G(s) = \cup_i a_i L_G(s_i)$

For an element $s[a_1:s_1, a_2:s_2, \ldots, a_n:s_n] \in G$, we can construct an equation of regular expressions, $W = \cup_i a_i W_i$, where W and W_i $(i = 1, 2, \ldots, n)$ stand for sets of regular expressions. Proposition 4.2 means that $L_G(s)$ and $L_G(s_i)$ $(i = 1, 2, \ldots, n)$ is a solution of the equation. We consider solving these equations constructed by G, using a graph reduction system.

DEFINITION 4.3. *We define three kinds of reduction rules. Assume* $a, a_i, b_i \in E$.

(R) *Reducing rule for x*:

For any n, $1 \le i \le n$ *and any regular expressions* $(a_1, \ldots, a_n)$,

(T_1) $t_0 = x[a_1:x_1, \ldots, a_{i-1}:x, a_i:x, a_{i+1}:x, \ldots, a_n:x_n]$
$t_1 = x[(a_i^* a_1):x_1, \ldots, (a_i^* a_{i-1}):x_{i-1}, (a_i^* a_{i+1}):x_{i+1}, (a_i^* a_n):x_n]$

(E) *Expansion rule for y*:

For any n, m, $1 \le i \le m$ *and any regular expressions* $(a_1, \ldots, a_n, b_1, \ldots, b_m)$,

(T_1) $t_0 = x[b_1:y_1, \ldots, b_{i-1}:y_{i-1}, b_i:y[a_1:x_1, \ldots, a_n:x_n], b_{i+1}:y_{i+1}, \ldots, b_m:y_m]$

$t_1 = x[b_1:y_1, \ldots, b_{i-1}:y_{i-1}, (b_i a_1):x_1, \ldots, (b_i a_n):x_n, b_{i+1}:y_{i+1}, \ldots, b_m:y_m]$

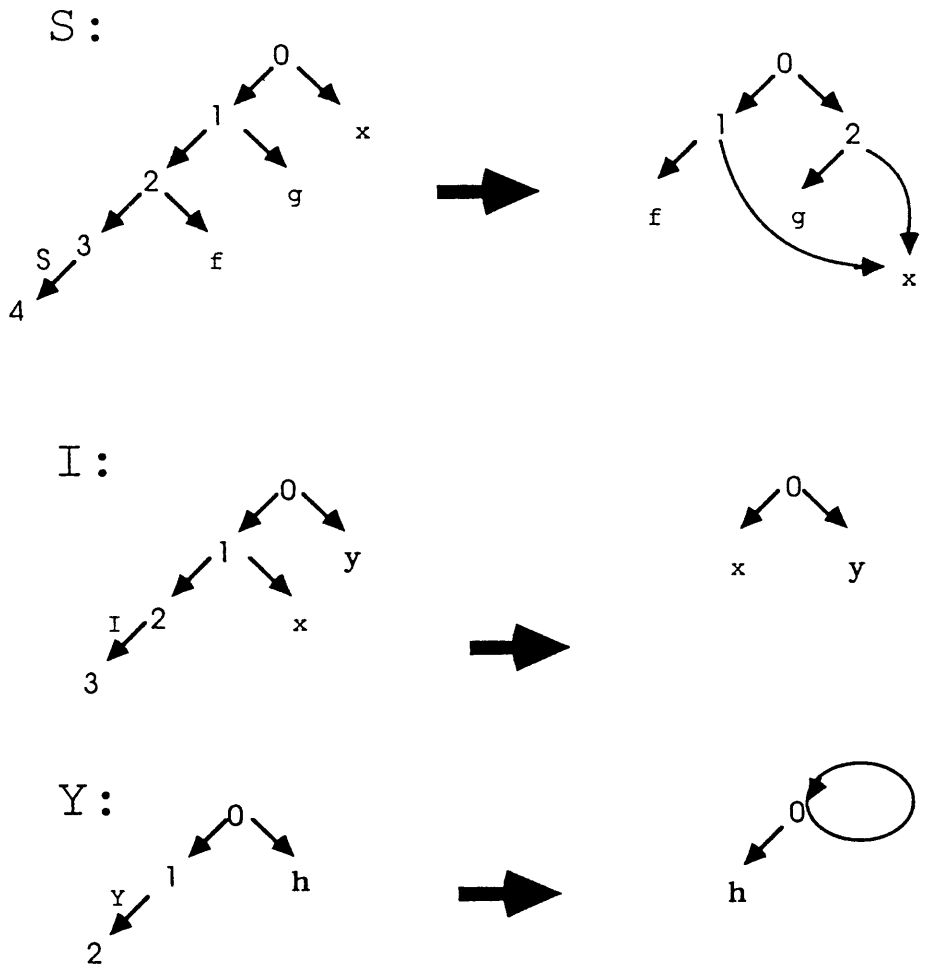

Figure 3. *SK-reduction rules*

(C) Combination rule for y:
For any n, $1 < i < j < n$ *and any regular expressions* $(a_1, \ldots, a_m)$,

(T_1) $t_0 = x[a_1 : y_1, \ldots, a_i : y, \ldots, a_j : y, \ldots, a_m : y_m]$
$t_1 = x[a_1 : y_1, \ldots, (a_i + a_j) : y_i, \ldots, a_{j-1} : y_{j-1}, a_{j+1} : y_{j+1}, a_m : y_m]$

PROPOSITION 4.4. *For any automaton graph G, there exists an automaton graph H which is reduced from G applying a finite number of reduction rules such that any reduction rules can not apply to the graph H. That is, for a suitable application of reduction rules, there exists the reduction strategy to the normal form of the automaton graph.*

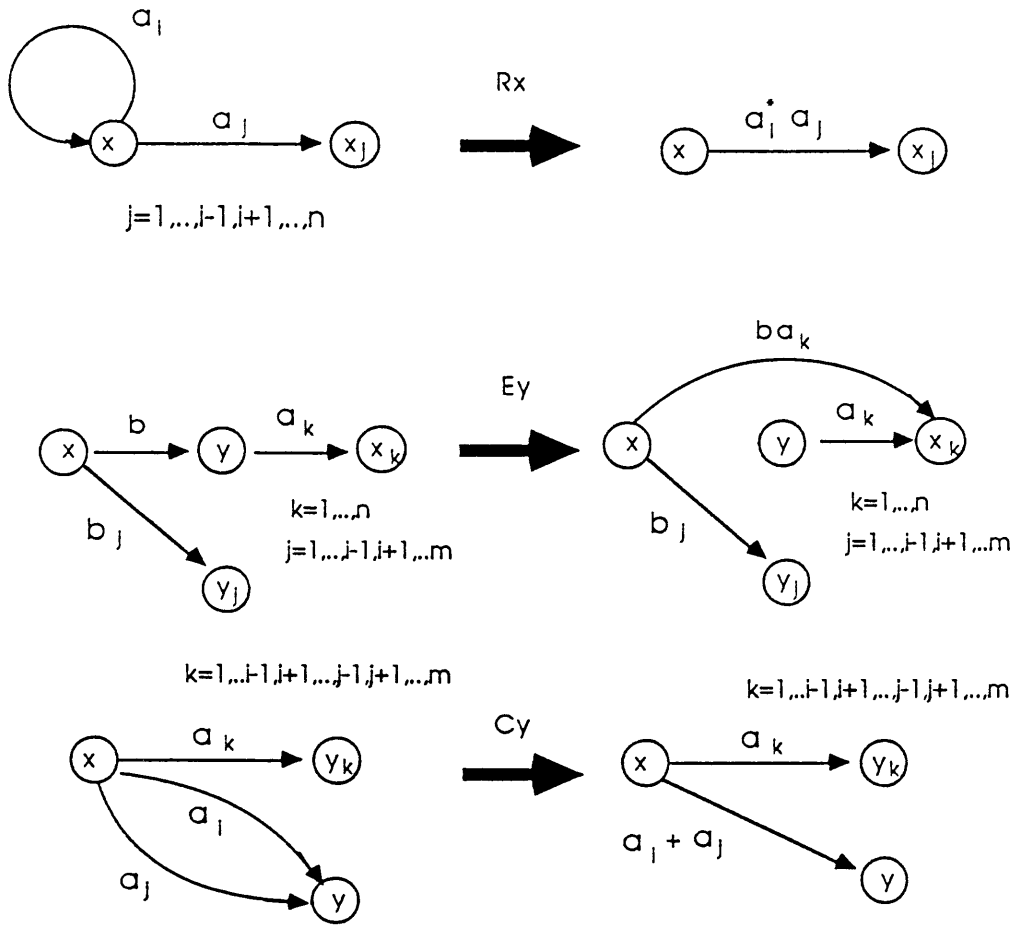

Figure 4. *Automaton graph reductions*

PROOF. At first, we note that we can not apply any reduction rules (R), (E) and (C) for a graph expression F. For another graph expression $x[a_1 : s_1, a_2 : s_2, \ldots, a_n : s_n]$, if x has an argument x itself, then first we apply the reduction rule (C) for x and next we apply the reduction rule (R) for x. For any graph expression which has an argument x, we apply the reduction rule (C) for x and next apply the reduction rule (E) for x. Since x itself does not have an argument x, the result of reduction does not have an argument x. Then we can eliminate the all incoming edges to x. So we can not apply any rules for x. Since G has finite element, by applying these processes to all graph expressions in G, we can get the normal form of the automaton graph. That is, we can not apply any reduction rules for any elements of G. ■

PROPOSITION 4.5. *For any reduction rules T, if $G \to_T H$, then $L_G(x) = L_H(x)$ (for any $x \in Rt(G)$).*

PROOF. We consider the set of equations of regular expressions, discussed after Proposition 4.2. $L_G(x)$ and $L_H(x)$ are solutions of the equation constructed by G and H, respectively. It is clear that each reduction rule does not change the solution of the equations. So we can conclude that $L_G(x) = L_H(x)$ for any $x \in Rt(G)$. ■

For an automaton graph G, we choose the suitable reduction procedure and get the normal form H of the automaton graph. Each element of H has the form x or $x[w:F]$. So we know that the set $L_H(x)$ easily $\emptyset$ or w respectively. That is, the procedure of reducing an automaton graph to the normal form is corresponds to the process of solving the equation of regular expressions constructed by the automaton graph.

For any reduction $G \rightarrow_R H$ and $G \rightarrow_C H$, there hold $Wt(G \rightarrow_R H) = Rd(G \rightarrow_R H)$ and $Wt(G \rightarrow_C H) = Rd(G \rightarrow_C H)$. Using the Proposition 3.5, we have a parallel reduction of (R) and (C) is a sequentially simulatable. But it is not always true that $Wt(G \rightarrow_E H) = Rd(G \rightarrow_E H)$.

THEOREM 4.6. *For any parallel reduction* $G \rightarrow_{\{t_1, t_2, \ldots, t_n\}} H$, *it holds* $L_G(x) = L_H(x)$ (*for any* $x \in Rt(G)$).

PROOF. Reduction rules correspond to the transformations of equations. For any reduction rules, it is obvious that a solution of the equation before a transformation becomes a solution of the equation after the transformation. So $L_G(x)$ is a solution of the equation constructed by the graph H. We know that a solution of equations of regular expressions is unique. Then we have $L_G(x)$ is the unique solution of the equation constructed by H. That is, $L_G(x) = L_H(x)$ for any $x \in Id(G)$. ■

5. Concluding Remark

Theorem 4.6 gives an interesting insight into parallel reductions. Parallel reductions in the system is not always sequentially simulatable. But if we restrict our aim to finding out a solution of the equation constructed by a graph, we can use parallel reductions. The importance of Theorem 4.6 is that it guarantees the correctness of solutions calculated by parallel reductions.

The result gives us a new insight into parallel executions. We are inclined to consider only sequentially simulatable parallel reductions. Because it is difficult to consider the behavior of a non sequential simulatable parallel reduction. But according to the semantics of a graph reduction system, the general parallel reduction may be meaningful. Furthermore we can formally prove the correctness of parallel reductions.

Acknowledgments

The author would like to express his sincere thanks to Professor Yasuo Kawahara for his valuable suggestions, constructive comments and encouragement during the course of this study.

References

[1] BOLTON, D., HANKIN, C., and KELLY, P.: *Parallel object-oriented descriptions of graph reduction machines*, Future Generations Computer Systems, **6** (1990), 225–239.

[2] CHENADEC, P.: *Canonical forms in finitely presented algebras*, Pitman, London, 1986.

[3] COURCELLE, B.: *A representation of graphs by algebraic expressions and its use for graph rewriting systems*, LNCS **291** (1986), 112–131.

[4] COURCELLE, B.: *The Monadic Second-Order Logic of Graphs, II: Infinite Graphs of Bounded Width*, Math. Systems Theory **21** (1989), 187–221.

[5] COUSINEAU, G., CURIEN, P-L., and MAUNY, M.: *The categorical abstract machine*, Func. Prog. Lang. and Comp. Arch, LNCS **201** (1985), 50–64.

[6] DERSHOWITZ, N.: *Orderings for term-rewriting systems*, Theoret. Comput. Sci., **17** (1982), 279–301.

[7] EHRIG, H., and ROSEN, B.K.: *Parallelism and concurrency of graph manipulations*, Theoret. Comput. Sci. **11** (1980), 247–275.

[8] EHRIG, H., NAGL, M., ROZENBERG, G. and ROSENFELD, A. (Eds.): *Graph-Grammars and Their Application to Computer Science*, 3rd International Workshop, LNCS **291** (1986).

[9] GLAUERT J.R.W., KENNAWAY, J.R. and SLEEP, M.R.: *DACTL: A computational model and compiler target language based on graph reduction*, Report SYS-C87-03, University of East Anglia, 1987.

[10] GLAUERT, J.R.W., HAMMOND, K., KENNAWAY, J.R., SLEEP, M.R., SOMMER, G.W., HOLT, N., REEVE, M. and WATSON, I.: *Extensions to Core Dactl 1*, Report SYS-C88-01, University of East Anglia, 1988.

[11] KAWAHARA, Y.: *Pushout-complements and basic concepts of grammars in toposes*, Theoret. Comput. Sci. **77** (1990), 267–289.

[12] KENNAWAY, R.: *On "On graph rewritings"*, Theoret. Comput. Sci. **52** (1987), 37–58.

[13] MIYANO, S.: *On an automaton which recognizes a family of automata*, Mem. of Fac. of Sci. Kyushu University, Series A, Math., **32** (1978), 37–51.

[14] MIZOGUCHI, Y., OHTSUKA H., and KAWAHARA, Y.: *A symbolic calculus of regular expressions*, Bull. of Informatics and Cybernetics, **22** (1987), 165–170.

[15] MIZOGUCHI, Y. and KAWAHARA, Y.: *Graph rewritings without gluing conditions*, RIFIS Technical Report, CS–42, Kyushu University (1991).

[16] MIZOGUCHI, Y.: *A graph structure over the category of sets and partial functions*, RIFIS Technical Report, CS–53, Kyushu University (1992).

[17] RAOULT, J.C.: *On graph rewritings*, Theoret. Comput. Sci. **32** (1984), 1–24.

[18] TURNER, D.A.: *A new implementation technique applicative languages*, Software-practice and experience, **9** (1979), 31–49.

Received June 5, 1991
Revised October 2, 1991.
Communicated by Y. Kawahara