Parallel Algorithms for Refutation Tree Problem on Formal Graph Systems

Uchida, Tomoyuki
Department of Information Systems Kyushu University

Shoudai, Takayoshi
Faculty of Engineering, Yamaguchi University

Miyano, Satoru
Research Institute of Fundamental Information Science Kyushu University

出版情報：RIFIS Technical Report. 59, 1992-07-20. Research Institute of Fundamental Information Science, Kyushu University

RIFIS-TR-CS-59

RIFIS Technical Report

Parallel Algorithms for Refutation Tree Problem on Formal Graph Systems

Tomoyuki Uchida
Takayoshi Shoudai
Satoru Miyano

July 20, 1992
Revised: April 19, 1993

Research Institute of Fundamental Information Science
Kyushu University 33
Fukuoka 812, Japan

E-mail: uchida@rifis.sci.kyushu-u.ac.jp Phone: 092(641)1101 Ex. 4479

Parallel Algorithms for Refutation Tree Problem on Formal Graph Systems

Tomoyuki Uchida[†]
uchida@rifis.sci.kyushu-u.ac.jp

Takayoshi Shoudai[*]
shoudai@rifis.sci.kyushu-u.ac.jp

Satoru Miyano[‡]
miyano@rifis.sci.kyushu-u.ac.jp

† Department of Information Systems, Kyushu University 39, Kasuga 816, Japan.
* Faculty of Engineering, Yamaguchi University, Ube 755, Japan.
‡ Research Institute of Fundamental Information Science, Kyushu University 33, Fukuoka 812, Japan.

Abstract

We define a new framework for rewriting graphs, called a formal graph system (FGS), which is a logic program having hypergraphs instead of terms in first-order logic. We first prove that a class of graphs is generated by a hyperedge replacement grammar if and only if it is defined by an FGS of a special form called a regular FGS. In the same way as logic programs, we can define a refutation tree for an FGS. The classes of TTSP graphs and outerplanar graphs are definable by regular FGSs. Then, we consider the problem of constructing a refutation tree of a graph for these FGSs. For the FGS defining TTSP graphs, we present a refutation tree algorithm of $O(\log^2 n + \log m)$ time with $O(n+m)$ processors on an EREW PRAM. For the FGS defining outerplanar graphs, we show that the refutation tree problem can be solved in $O(\log^2 n)$ time with $O(n+m)$ processors on an EREW PRAM. Here, n and m are the numbers of vertices and edges of an input graph, respectively.

1 Introduction

The refutation tree problem is to compute refutation trees which is associated with the structure of a graph generated by a new formal system, called a formal graph system (FGS).

We define an FGS by extending an elementary formal system [1, 2, 19], which is a kind of logic program dealing with strings, so that it can directly manipulate graphs. FGSs are logic programs which use graphs in place of terms in first-order logic. In the same way as logic programs, we define a refutation tree for an FGS. A refutation tree of a graph G is a tree representing explicitly the decomposition of G. Since an FGS consists of definite clauses which explicitly describes how it generates graphs and since the refutation tree problem may allow the divide-and-conquer technique, it may be useful for graph problems to employ FGSs in designing efficient parallel algorithms. The refutation tree problem for an FGS is to construct refutation trees of graphs defined by the FGS and can be regarded as the problem to decide whether an input graph is generated by the FGS.

We consider two classes of graphs. One is the class of two-terminal series parallel (TTSP) graphs and the other is the class of outerplanar graphs. We present FGSs defining these classes and we devise efficient parallel algorithms solving the refutation tree problem with $O(n+m)$ processors on an EREW PRAM, where n (m) is the number of vertices (edges) of an input graph. For the class of TTSP graphs, we present an $O(\log^2 n + \log m)$ time algorithm. For the class of undirected outerplanar graphs, Diks *et. al.* [4] has given an optimal parallel algorithm on a CREW PRAM which solves the problem of testing outerplanarity in $T(n) = O(\log n \log^* n)$ time using $n/T(n)$ processors. In this paper, with $O(n+m)$ processors on an EREW PRAM, we give an $O(\log^2 n)$ time algorithm by employing the algorithm for TTSP graphs. These results assert that efficient parallel algorithms may exist for a large number of **NP**-complete problems when these problems are restricted to the classes of TTSP graphs and undirected outerplanar graphs. For example, Xin He [9] has presented efficient parallel algorithms for solving the following three problems on TTSP graphs: 3-coloring, depth-first spanning tree, and breadth-first spanning tree.

It has been shown that polynomial time algorithms [?, 18] and **NC**-algorithms [17] exist for a number of **NP**-complete problems when these problems are restricted to the class of graphs generated by context-free graph grammars [18]. Context-free graph grammars (CFGG) [14, 18, ?] and hyperedge replacement grammars (HRG) [3, 7, 11] have been known as formal systems derived by expanding the concept of context-free grammars to graphs.

We prove that a class of graphs is generated by a HRG due to Lautemann [11] if and only if it is defined by an FGS of a special form called a regular FGS. For a HRG, the refutation tree problem can be regarded as the parsing problem of constructing a "parse tree" of a graph generated by the HRG. Rytter and Szymacha [17] and Xin He [10] have presented parallel algorithms solving the parsing problem for any class of graphs generated by a CFGG. A parse tree of a graph G is, in a sense, a tree which shows a method of decomposing G into a set of terminal elements by assigning a right-hand side of a production in the HRG generating G to each node in the tree. Therefore, it does not express in a straightforward way how G is decomposed into the set of terminal elements. However, since an atom with a subgraph of G is assigned to each node of the tree, refutation trees express the structures of graphs directly and can also cope with a class of graphs for which no HRG exists. These show that an FGS has a richer structure for generating graphs than a HRG, and that refutation tree for an FGS is more suitable for expressing the structure of a graph than a parse tree for an HRG.

2 Formal Graph Systems and Refutation Tree Problem

Let Σ and Λ be finite alphabets. A *colored-graph* $g = (V, E, \varphi, \psi)$ over (Σ, Λ) consists of a vertex set V, an edge set E, a vertex labeling $\varphi : V \to \Sigma$ and an edge labeling $\psi : E \to \Lambda$. We allow multiple edges in g and g is directed or undirected. We use lower case letters for representing colored-graphs.

Let X be a finite alphabet whose elements are called *variables*. We denote variables by $x, y, \cdots$. We assume that each variable x in X has the *rank*, denoted by $rank(x)$, that is a nonnegative integer. Assume that $\Sigma \cap X = \emptyset$ and $\Lambda \cap X = \emptyset$.

Definition 1. A *directed term graph* $g = (V, E, \varphi, \psi, H, \lambda, ports)$ over $\langle \Sigma, \Lambda, X \rangle$ consists of the following:

(1) (V, E, φ, ψ) is a directed colored-graph over (Σ, Λ).

(2) H is a finite set whose elements are called *hyperedges*.

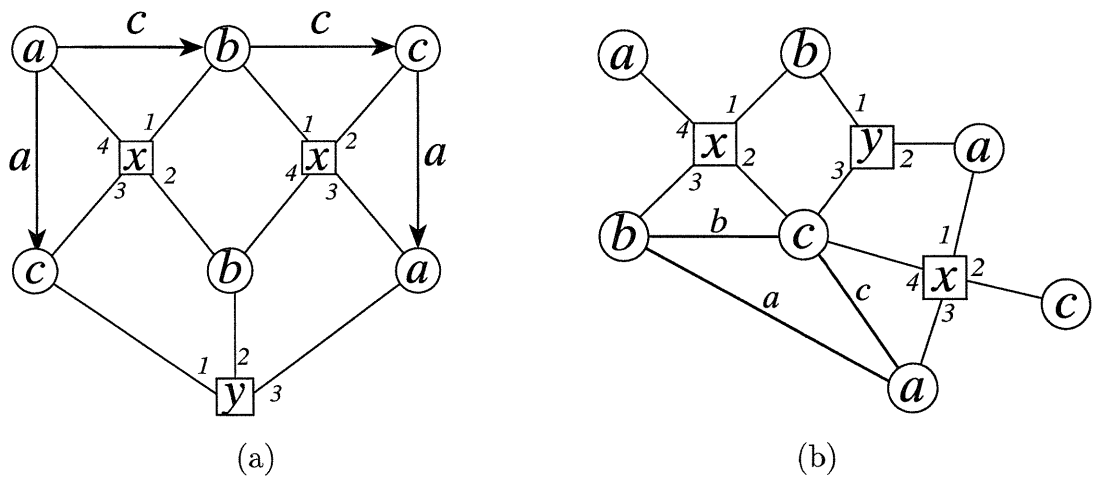


Figure 1: Term graphs. A hyperedge is represented by a box with lines to its ports. The order of the ports is indicated by numbers at these lines.

(3) $\lambda : H \to X$ is a labeling function. For a hyperedge $e \in H$, $rank(\lambda(e))$ is called the *rank* of e.

(4) $ports : H \to V^*$ is a mapping such that for every $e \in H$, $ports(e)$ is a list of $rank(\lambda(e))$ distinct vertices. These vertices are called the *ports* of e.

We denote term graphs by $f, g, \cdots$. A term graph is called *ground* if $H = \emptyset$. We denote by $\mathcal{T}(\Sigma, \Lambda, X)$ the family of term graphs over $\langle \Sigma, \Lambda, X \rangle$ and by $\mathcal{T}(\Sigma, \Lambda)$ the family of all ground term graphs. We can define an *undirected term graph* in the same way.

We identity directed (or undirected) term graphs without hyperedge with colored-graphs.

Example 1. We draw directed and undirected term graphs as in Figure 1 (a) and (b), respectively.

Let $g_1, \ldots, g_n$ be term graphs in $\mathcal{T}(\Sigma, \Lambda, X)$. An *atom* is an expression of the form $p(g_1, \ldots, g_n)$, where p is a predicate symbol with arity n. Hereafter we use $p, q, \ldots$ to denote predicate symbols. Let $A, B_1, \ldots, B_n$ be atoms, where $n \geq 0$. Then, a *graph rewriting rule* is a clause of the form $A \leftarrow B_1, \ldots, B_n$. We call the atom A the *head* and the part $B_1, \ldots, B_n$ the *body* of the graph rewriting rule.

Definition 2. A *formal graph system* (*FGS*) is a finite set of graph rewriting rules.

We give some notions for term graphs and atoms.

A term graph $f = (V_f, E_f, \varphi_f, \psi_f, H_f, \lambda_f, ports_f)$ is a *subgraph* of $g = (V_g, E_g, \varphi_g, \psi_g, H_g, \lambda_g, ports_g)$ if $f' = (V_f, E_f, \varphi_f, \psi_f)$ is a subgraph of $g' = (V_g, E_g, \varphi_g, \psi_g)$, $H_f \subseteq H_g$, and $\lambda_f(e) = \lambda_g(e)$ and $ports_f(e) = ports_g(e)$ for all $e \in H_f$.

Let $g_1 = (V_1, E_1, \varphi_1, \psi_1, H_1, \lambda_1, ports_1)$ and $g_2 = (V_2, E_2, \varphi_2, \psi_2, H_2, \lambda_2, ports_2)$ be term graphs. We say that g_1 and g_2 are *isomorphic*, denoted by $g_1 \simeq g_2$, if the following conditions are satisfied:

(1) The colored-graphs $g_1' = (V_1, E_1, \varphi_1, \psi_1)$ and $g_2' = (V_2, E_2, \varphi_2, \psi_2)$ are isomorphic including vertex and edge labelings.

(2) Let $\pi : V_1 \to V_2$ be the bijection giving the isomorphism in (1). There is a bijection $\varpi : H_1 \to H_2$ such that for every $e \in H_1$, $ports_2(\varpi(e)) = \pi^*(ports_1(e))$ and $\lambda_2(\varpi(e)) = \lambda_1(e)$, where π^* is defined as $\pi^*((v_1, \ldots, v_m)) = (\pi(v_1), \ldots, \pi(v_m))$.

Let $(v_1, \ldots, v_r)$ and $(u_1, \ldots, u_r)$ be two lists of r distinct vertices of g_1 and g_2, respectively. We also say that $[g_1, (v_1, \ldots, v_r)]$ and $[g_2, (u_1, \ldots, u_r)]$ are *isomorphic* if (1) and (2) are satisfied and $\pi(v_i) = u_i$ for each $1 \leq i \leq r$. We call a pair $[g, \sigma]$ of a term graph g over $\langle \Sigma, \Lambda, X \rangle$ and a list σ of r distinct vertices in g an *r-hypergraph* over $\langle \Sigma, \Lambda, X \rangle$.

For any two atoms $p(f_1, \ldots, f_n)$ and $p(g_1, \ldots, g_n)$, we denote $p(f_1, \ldots, f_n) \simeq p(g_1, \ldots, g_n)$ if $f_i \simeq g_i$ for each $1 \leq i \leq n$.

Let $g = (V_g, E_g, \varphi_g, \psi_g, H_g, \lambda_g, ports_g)$ and $f = (V_f, E_f, \varphi_f, \psi_f, H_f, \lambda_f, ports_f)$ be term graphs, e be a hyperedge in H_f of rank r with ports $(u_1, \ldots, u_r)$, and $[g, \sigma]$ be an r-hypergraph with $\sigma = (v_1, \ldots, v_r)$. The *hyperedge replacement* $e \leftarrow [g, \sigma]$ is the following operation on f: The term graph obtained by the hyperedge replacement $e \leftarrow [g, \sigma]$ on f, denoted by $f(e \leftarrow [g, \sigma]) = (V, E, \varphi, \psi, H, \lambda, ports)$ is defined in the following way: Let $g' = (V'_g, E'_g, \varphi'_g, \psi'_g, H'_g, \lambda'_g, ports'_g)$ be a copy of g. For a vertex $v \in V_g$, we denote the corresponding copy vertex by v'. We attach g' to f by removing the hyperedge e from H_f and by identifying the ports $u_1, \ldots, u_r$ of e in f with $v'_1, \ldots, v'_r$ in g', respectively. We set $\varphi(u_i) = \varphi_f(u_i)$ for each $1 \leq i \leq r$, i.e., the label of u_i in f is used for the new term graph $f(e \leftarrow [g, \sigma])$. Let $\Upsilon = \{e_1 \leftarrow [g_1, \sigma_1], \ldots, e_m \leftarrow [g_m, \sigma_m]\}$ be a set of hyperedge replacements on f. We denote by $f(\Upsilon)$ the term graph obtained by applying all hyperedge replacements in Υ in parallel.

Let x be a variable in X with $rank(x) = r$. Let $g = (V_g, E_g, \varphi_g, \psi_g, H_g, \lambda_g, ports_g)$ be a term graph in $\mathcal{T}(\Sigma, \Lambda, X)$ and $[g, \sigma]$ be an r-hypergraph. We call the form $x := [g, \sigma]$ a *binding* for x. We say that a binding $x := [g, \sigma]$ is *trivial* if g is a term graph consisting of r vertices without any edges such that it has a unique hyperedge $e \in H$ with $\lambda(e) = x$. A *substitution* θ is a finite collection of bindings $\{x_1 := [g_1, \sigma_1], \ldots, x_n := [g_n, \sigma_n]\}$, where x_i's are mutually distinct variables in X and each g_i $(1 \leq i \leq n)$ has no hyperedge labeled with a variable in $\{x_1, \ldots, x_n\}$.

Let $f = (V_f, E_f, \varphi_f, \psi_f, H_f, \lambda_f, ports_f)$ be a term graph and $\theta = \{x_1 := [g_1, \sigma_1], \ldots, x_n := [g_n, \sigma_n]\}$ be a substitution. For x_i, let $H_{\lambda_f}(x_i) = \{e \in H_f \mid \lambda_f(e) = x_i\}$ and $\Upsilon_i = \{e \leftarrow [g_i, \sigma_i] \mid e \in H_{\lambda_f}(x_i)\}$. Then let $\Upsilon_\theta = \bigcup_{i=1}^n \Upsilon_i$. The term graph $f\theta$ called the *instance* of f by θ is defined by $f(\Upsilon_\theta)$. We remark that the set of the hyperedges in $f\theta$ consists of the hyperedges in H_f which are not in $H_{\lambda_f}(x_1) \cup \cdots \cup H_{\lambda_f}(x_n)$ and the newly added hyperedges of the graphs attached to f by the substitution θ.

Example 2. Let f, g and h be term graphs given in Figure 2, and $\theta = \{x := [g, (u_1, u_2)], y := [h, (v_1, v_2)]\}$ be a substitution, where u_1 and u_2 (resp., v_1 and v_2) are vertices in g (resp., h). Then the instance $f\theta$ is a term graph shown in Figure 2.

We introduce some notions similar to those in logic programming [12]. Let $\theta = \{x_1 := [g_1, \sigma_1], \ldots, x_n := [g_n, \sigma_n]\}$ and $\tau = \{y_1 := [h_1, \sigma'_1], \ldots, y_m := [h_m, \sigma'_m]\}$ be substitutions. Then the *composition* $\theta\tau$ of θ and τ is the substitution obtained from $\{x_1 := [g_1\tau, \sigma_1], \ldots, x_n := [g_n\tau, \sigma_n], y_1 := [h_1, \sigma'_1], \ldots, y_m := [h_m, \sigma'_m]\}$ by deleting the following bindings:

(i) $x_i := [g_i\tau, \sigma_i]$ which is trivial.

(ii) $y_j := [h_j, \sigma'_j]$ with $y_j = x_k$ for some x_k.

Let $\theta_1, \ldots, \theta_n$ be substitutions. We denote a composition $\theta_1 \cdots \theta_n$ by $\prod_{i=1}^n \theta_i$. The substitution given by the empty set is called the *identity substitution* and is denoted by ε. The elementary properties of substitutions are given in the following proposition without proofs.

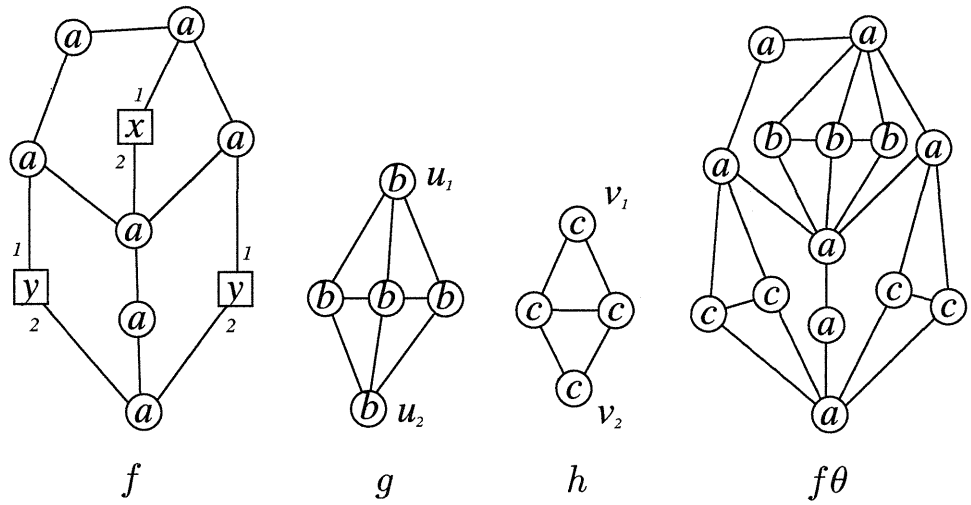


Figure 2: An example of an instance $f\theta$ with $\theta = \{x := [g, (u_1, u_2)], y := [h, (v_1, v_2)]\}$

Proposition 1. Let θ, τ and γ be substitutions.

(a) $\theta\varepsilon = \varepsilon\theta = \theta$.

(b) $(g\tau)\gamma = g(\tau\gamma)$ for any term graph g.

(c) $(\theta\tau)\gamma = \theta(\tau\gamma)$.

For a substitution θ, we define $p(f_1, \ldots, f_n)\theta = p(f_1\theta, \ldots, f_n\theta)$ and $(A \leftarrow B_1, \ldots, B_m)\theta = A\theta \leftarrow B_1\theta, \ldots, B_m\theta$.

Let g_1 and g_2 are two term graphs or atoms. Then a substitution θ is a *unifier* of g_1 and g_2 if $g_1\theta \simeq g_2\theta$. If $g_1 \simeq g_2\theta$ and $g_1\theta' \simeq g_2$ for some substitutions θ and θ', g_1 is called a *variant* of g_2. A *goal* is a graph rewriting rule of the form $\leftarrow B_1, \ldots, B_m$ $(m \geq 0)$. If $m = 1$, "$\leftarrow B_1$" is called a *unit goal.* If $m = 0$, we denote it "$\Box$" and call it an *empty goal.* We assume a *computation rule* Q to select an atom from a goal. For a graph rewriting rule C, let $var(C)$ be the set of all variables labeling the hyperedges of the term graphs in C.

Let Γ be an FGS, and D a goal. A *derivation from* D is a (finite or infinite) sequence of triples (D_i, θ_i, C_i) $(i = 0, 1, \ldots)$ that satisfies the following conditions:

(3) D_i is a goal, θ_i is a substitution, C_i is a variant of a graph rewriting rule in Γ, and $D_0 = D$.

(4) $var(C_i) \cap var(C_j) = \emptyset$ $(i \neq j)$, and $var(C_i) \cap var(D) = \emptyset$ for every i.

(5) Let D_i be $\leftarrow A_1, \ldots, A_k$ and A_m be the atom selected by Q. Let C_i be $A \leftarrow B_1, \ldots, B_q$. Then θ_i is a unifier of A and A_m and D_{i+1} is the goal $\leftarrow A_1\theta_i, \ldots, A_{m-1}\theta_i, B_1\theta_i, \ldots, B_q\theta_i, A_{m+1}\theta_i, \ldots, A_k\theta_i$.

A *refutation* is a finite derivation ending with the empty goal.

Let $F = \{(D_i, \theta_i, C_i)\}_{0 \leq i \leq k}$ be a refutation from a unit goal D in Γ. The *refutation tree* of F is a tree defined as follows:

(6) Every vertex is labeled with a unit goal or the empty goal.

(7) The label of the root is the unit goal $D_0 = D$.

(8) Every leaf is labeled with the empty goal.

(9) T_0 is a tree consisting of only one vertex labeled with D. For $0 \leq i \leq k$, T_{i+1} is a tree obtained by applying (D_i, θ_i, C_i) to the tree T_i as follows: let D_i be $\leftarrow A^i_1, \ldots, A^i_{n_i}$. Assume that T_i has a leaf labeled with $\leftarrow A^i_{m_i}$. Let C_i be $A^i \leftarrow B^i_1, \ldots, B^i_{q_i}$ and θ_i be a unifier of A^i and the atom $A^i_{m_i}$. Then T_{i+1} is obtained by adding new q_i vertices labeled with $\leftarrow B^i_1\theta_i, \ldots, \leftarrow B^i_{q_i}\theta_i$ as the children of the vertex labeled with $\leftarrow A^i_{m_i}$.

Definition 3. Let Γ be an FGS. We define the relation $\Gamma \vdash C$ for a rule C inductively as follows:

(1) If $\Gamma \ni C$, then $\Gamma \vdash C$.
(2) If $\Gamma \vdash C$, then $\Gamma \vdash C\theta$ for any substitution θ.
(3) If $\Gamma \vdash A \leftarrow B_1, \ldots, B_n$ and $\Gamma \vdash B_i \leftarrow C_1, \ldots, C_m$, then $\Gamma \vdash A \leftarrow B_1, \ldots, B_{i-1}, C_1, \ldots, C_m, B_{i+1}, \ldots, B_n$.

A rule C is *provable from* Γ if $\Gamma \vdash C$. For an FGS Γ and its predicate symbol p with arity n, we define $GL(\Gamma, p) = \{(h_1, \ldots, h_n) \in (\mathcal{T}(\Sigma, \Lambda))^n \mid \Gamma \vdash p(h_1, \ldots, h_n) \leftarrow\}$. In case $n = 1$, $GL(\Gamma, p)$ defines a subset of $\mathcal{T}(\Sigma, \Lambda)$ called a *graph language*. We say that a graph language $L \subseteq \mathcal{T}(\Sigma, \Lambda)$ is *definable by* FGS or an *FGS language* if such a pair (Γ, p) exists.

Let $SS(\Gamma)$ is the set of all ground atoms such that there exists a refutation from $\leftarrow A$ in Γ. Let $PS(\Gamma)$ be the set of all ground atoms which are provable from Γ. Then we can prove the following proposition in the same way as logic programs (see Ref. [12]).

Proposition 2. For any FGS Γ, $SS(\Gamma) = PS(\Gamma)$.

Definition 4. Let Γ be an FGS and p be its unary predicate symbol. The *refutation tree problem* for (Γ, p), denoted by $\mathrm{RT}(\Gamma, p)$, is defined as follows:

INSTANCE: A ground term graph g.
PROBLEM: If there is a refutation from the goal $\leftarrow p(g)$ in Γ, construct its refutation tree.

3 Regular FGS and Hyperedge Replacement Grammar

This section introduces a subclass of FGSs called regular FGSs, and characterizes hyperedge replacement grammars [7, 11, 21] by regular FGSs. Hyperedge replacement is one of the most elementary and frequently used concepts of graph transformation with the characteristics of context-free rewriting. Hyperedge replacement grammars are certain context-free graph grammars.

Definition 5. Let Δ be a subset of Σ called a *pointer set*. A *regular term graph* with a pointer set Δ is a term graph $g = (V, E, \varphi, \psi, H, \lambda, ports)$ such that there is at most one hyperedge labeled with x for each $x \in X$ and there is at most one vertex labeled with s for each $s \in \Delta$.

A term graph $g = (V, E, \varphi, \psi, H, \lambda, ports)$ is a *star graph* for a variable x if $E = \emptyset$ and H consists of a unique hyperedge e labeled with x such that the set of ports of e is V.

Definition 6. Let Γ be an FGS and Π be the set of predicate symbols in Γ. Then Γ is said to be *regular* if every predicate symbol in Γ is unary and there is a sequence $reg(p)$ of distinct symbols in Σ for every $p \in \Pi$ such that each graph rewriting rule $p_0(g_0) \leftarrow p_1(g_1), \ldots, p_n(g_n)$ in Γ satisfies the following conditions:

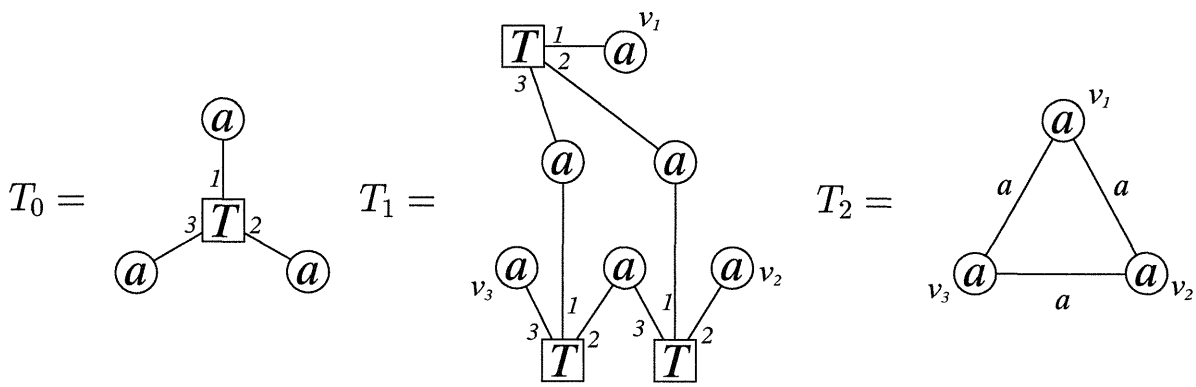


Figure 3: HRG $\mathcal{G} = (S, R)$ with $R = \{S \to T_0, T \to [T_1, (v_1, v_2, v_3)], T \to [T_2, (v_1, v_2, v_3)]\}$

(1) Each term graph $g_i = (V_i, E_i, \varphi_i, \psi_i, H_i, \lambda_i, ports_i)$ $(0 \leq i \leq n)$ is a regular term graph with respect to $\Delta = \bigcup_{p \in \Pi}\{a \mid a \text{ is a symbol of } reg(p)\}$.
(2) For every label a of vertices in g_0, if a is in Δ, then a is a symbol of $reg(p_0)$.
(3) For $1 \leq i \leq n$, g_i is a star graph for a variable of rank m_i such that jth port $(1 \leq j \leq m_i)$ of the hyperedge in H_i is labeled with the jth symbol of $reg(p_i)$, where m_i is the length of $reg(p_i)$.
(4) $\lambda_0(H_0) = \bigcup_{1 \leq i \leq n} \lambda_i(H_i)$ and $\lambda_i(H_i) \cap \lambda_j(H_j) = \emptyset$ for $1 \leq i < j \leq n$.

We extend hyperedge replacement grammars in Lautemann [11] by introducing vertex and edge labelings in such a way that these labels do not interfere with hyperedge replacement mechanism.

Let N be a finite alphabet. We call an element of N a *nonterminal.* Nonterminals are denoted by capital letters $A, B, C, \cdots$. We assume that each nonterminal A in N has the *rank*, denoted by $rank(A)$, that is a nonnegative integer.

Definition 7. A *hyperedge replacement grammar* (HRG) $\mathcal{G} = (S, R)$ is defined as follows:

(1) S is a nonterminal in N with $rank(S) = 0$, called the *start symbol.*
(2) R is a finite set of *productions* of the form $A \to [g, \sigma]$, where A is a nonterminal in N with $rank(A) = r$ and $[g, \sigma]$ is an r-hypergraph over $\langle \Sigma, \Lambda, N \rangle$.

Example 3. We draw an HRG $\mathcal{G} = (S, R)$ as Figure 3, where $R = \{S \to T_0, T \to [T_1, (v_1, v_2, v_3)], T \to [T_2, (v_1, v_2, v_3)]\}$.

Let $\mathcal{G} = (S, R)$ be an HRG. For $A \in N$, an r-hypergraph $[g, \sigma]$ over $\langle \Sigma, \Lambda, N \rangle$, and $i \geq 1$, we define the relation $A \to^i [g, \sigma]$ inductively as follows:

(1) We denote $A \to^1 [g, \sigma]$ if there is a production $A \to [g, \sigma]$ is in R.
(2) For $i \geq 2$, we denote $A \to^i [g, \sigma]$ if there are $j, l \geq 1$, an r-hypergraph $[f, \sigma]$, a hyperedge e in f of rank s with label B, and an s-hypergraph $[h, \sigma']$ such that $j + l = i$, $A \to^j [f, \sigma]$, $B \to^l [h, \sigma']$, and $[g, \sigma] = [f(e \leftarrow [h, \sigma']), \sigma]$.

We write $A \to^+ [g, \sigma]$ if $A \to^i [g, \sigma]$ for some $i \geq 1$. The *graph language generated by an HRG* $\mathcal{G} = (S, R)$ is the set $L(\mathcal{G}) = \{g \mid g \text{ is a term graph in } \mathcal{T}(\Sigma, \Lambda) \text{ and } S \to^+ g\}$. A set L of term graphs in $\mathcal{T}(\Sigma, \Lambda)$ is called an HR language if $L = L(\mathcal{G})$, for some HRG $\mathcal{G}$.

Theorem 1. A graph language L is definable by a regular FGS if and only if L is an HR language.

Proof. Let $\mathcal{G} = (S, R)$ be an HRG. Let N be the set of nonterminals of $\mathcal{G}$. Let Σ and Λ be the sets of labels of vertices and edges of the term graphs in R, respectively.

We define a regular FGS $\Gamma_{\mathcal{G}}$ in the following way: We regard each nonterminal A in N as a unary predicate symbol. For a unary predicate symbol A, let $reg(A)$ be a sequence of $rank(A)$ distinct symbols not in Σ. Then let $\Delta = \bigcup_{A \in N}\{a \mid a \text{ is a symbol in } reg(A)\}$.

For each production $A \to [g, (v_1, \ldots, v_r)]$ of $\mathcal{G}$, let $g = (V, E, \varphi, \psi, H, \lambda, ports)$ and let $H = \{e_1, \ldots, e_n\}$. Then $\Gamma_{\mathcal{G}}$ contains the graph rewriting rule $A(h) \leftarrow \lambda(e_1)(f_1), \ldots, \lambda(e_n)(f_n)$, where regular term graphs h and $f_1, \ldots, f_n$ with respect to Δ are defined as follows (see Example 4):

(a) $h = (V, E, \varphi', \psi, H, \lambda', ports)$ is defined by modifying the vertex and hyperedge labelings of g as follows:

  (i) For hyperedges $e_1, \ldots, e_n$ in H, let $\lambda'(e_1), \ldots, \lambda'(e_n)$ be distinct variables of rank $rank(\lambda(e_1)), \ldots, rank(\lambda(e_n))$, respectively.

  (ii) For each vertex v_j $(1 \leq j \leq r)$, v_j is labeled with the jth symbol of $reg(A)$. For other vertices in V, the same labeling as φ is used for φ'.

  Then h is a regular term graph over $\langle \Sigma \cup \Delta, \Lambda, \{\lambda'(e_1), \ldots, \lambda'(e_n)\}\rangle$ with respect to Δ.

(b) For $1 \leq i \leq n$, f_i is a star graph for a variable $\lambda'(e_i)$ such that the jth port of the hyperedge in f_i is labeled with the jth symbol in $reg(\lambda(e_i))$ for $1 \leq j \leq rank(\lambda'(e_i))$.

It is easy to see that $\Gamma_{\mathcal{G}}$ consisting of these graph rewriting rules is regular. Then we can show that $L(\mathcal{G})$ is definable by the FGS $\Gamma_{\mathcal{G}}$.

Conversely, let (Γ, p) be a pair of a regular FGS and its predicate symbol p. Without loss of generality, we may assume that p is the only predicate symbol in Γ that does not appear in the body of any graph rewriting rule in Γ. Let Σ and Λ be the sets of labels of vertices and edges of the term graphs in Γ, respectively.

We define an HRG $\mathcal{G}_\Gamma = (p, R_\Gamma)$ in the following way: Let Π be the set of predicate symbols of Γ. Since Γ is regular, every predicate symbol q in Π is unary and there is a sequence $reg(q)$ of distinct symbols in Σ such that each graph rewriting rule in Γ satisfies the conditions of Definition 6. Here we note that $reg(p)$ is the empty sequence. We regard a predicate symbol q in Π as a nonterminal. For a nonterminal q, let the *rank* of q be the length of $reg(q)$. Especially, p is the start symbol of $\mathcal{G}_\Gamma$.

For each graph rewriting rule $q_0(g_0) \leftarrow q_1(g_1), \ldots, q_n(g_n)$ of Γ, let $g_i = (V_i, E_i, \varphi_i, \psi_i, H_i, \lambda_i, ports_i)$ and r_i be the rank of q_i for $0 \leq i \leq n$. Then R_Γ contains the production $q_0 \to [h, (v_1, \ldots, v_{r_0})]$, where a term graph h over $\langle \Sigma, \Lambda, \Pi\rangle$ and the sequence $(v_1, \ldots, v_{r_0})$ of vertices in h are defined by modifying the hyperedge labeling of g_0 as follows:

(i) Let g_i $(1 \leq i \leq n)$ be the term graph with the hyperedge labeled with $\lambda_0(e)$. Then let the new label of e in h be q_i.

(ii) Let $(v_1, \ldots, v_{r_0})$ be the sequence of vertices in h with $(\varphi_0(v_1), \ldots, \varphi_0(v_{r_0})) = reg(q_0)$.

Then h is a term graph over $\langle \Sigma, \Lambda, \Pi\rangle$ and $[h, (v_1, \ldots, v_{r_0})]$ is an r_0-hypergraph.

Since Γ is regular, it is easy to see that $\mathcal{G}_\Gamma = (p, R_\Gamma)$ consisting of the start symbol p and these productions is an HRG. Then we can show that $GL(\Gamma, p)$ is definable by the HRG $\mathcal{G}_\Gamma = (p, R_\Gamma)$. $\square$

Example 4. Let $\mathcal{G} = (S, R)$ be an HRG given in Figure 3. Then the regular FGS $\Gamma_{\mathcal{G}}$ obtained from $\mathcal{G}$ is given in Figure 4.

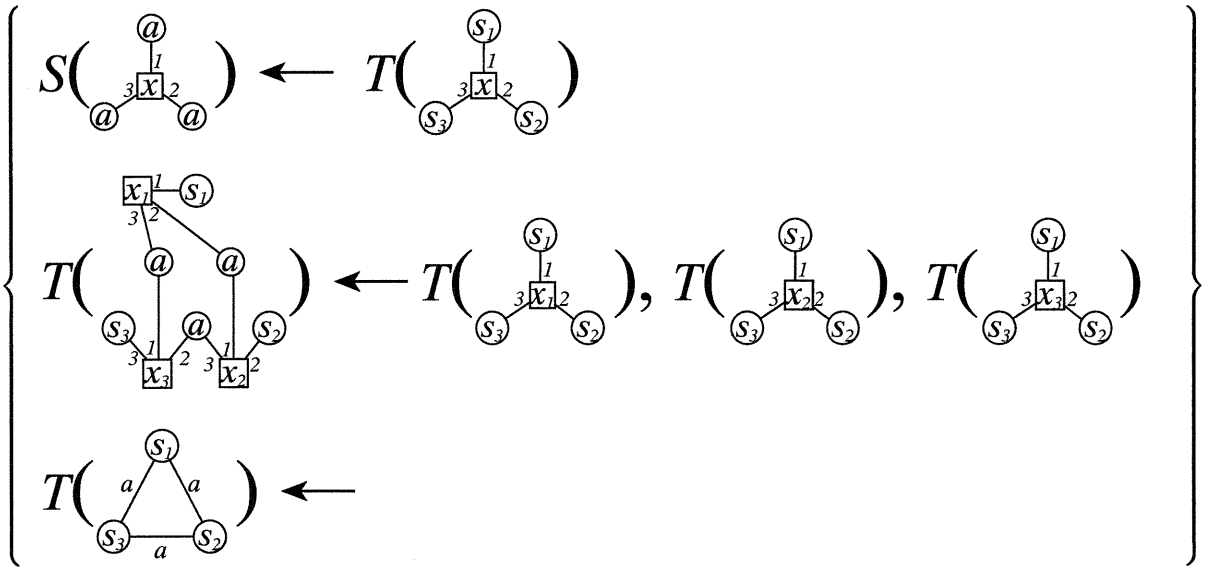


Figure 4: FGS $\Gamma_{\mathcal{G}}$.

It is known in Refs. [8], [11], [14] that the classes of trees, two-terminal series parallel graphs, graphs homeomorphic to a given graph, outerplanar graphs, graphs of cyclic bandwidth ≤ 2 and k-decomposable graphs are generated by HRGs. By Theorem 1, these classes are also definable by regular FGSs.

4 Parallel Algorithms for Refutation Tree Problem

Consider the regular FGSs Γ_{SP} in Figure 5 and Γ_{OP} in Figure 8. These FGSs define the classes of TTSP graphs and outerplanar graphs, respectively. This section gives parallel algorithms for constructing refutation trees for Γ_{SP} and Γ_{OP}.

4.1 Parallel Algorithm for TTSP Graphs

A *multidag* is a directed colored-graph $g = (V, E, \varphi, \psi)$ that allows multiple edges and does not contain any cycles. For a vertex $v \in V$, $indeg(v)$ denotes the number of edges entering v and $outdeg(v)$ denotes the number of edges leaving v. A vertex v with $indeg(v) = 0$ (resp., $outdeg(v) = 0$) is called a *source* (resp., a *sink*). A *two-terminal multidag* is a multidag with exactly one source and one sink.

Definition 8. *Two-terminal series parallel* (TTSP) graphs are two-terminal multidags over $\langle\{a, s, t\}, \{b\}\rangle$ defined as follows:

(1) A directed colored-graph consisting of two vertices u labeled with s and v labeled with t, and a single edge (u, v) labeled with b is a TTSP graph. The vertices u and v are the source and sink, respectively.

(2) For $i = 1, 2$, let g_i be a TTSP graph with the source u_i labeled with s and the sink v_i labeled with t. Then the graph obtained by either of the following two operations is a TTSP graph:

   (a) *Parallel composition*: Identify u_1 with u_2, and identify v_1 with v_2. The resulting graph has u_1 $(= u_2)$ as the source and v_1 $(= v_2)$ as the sink.

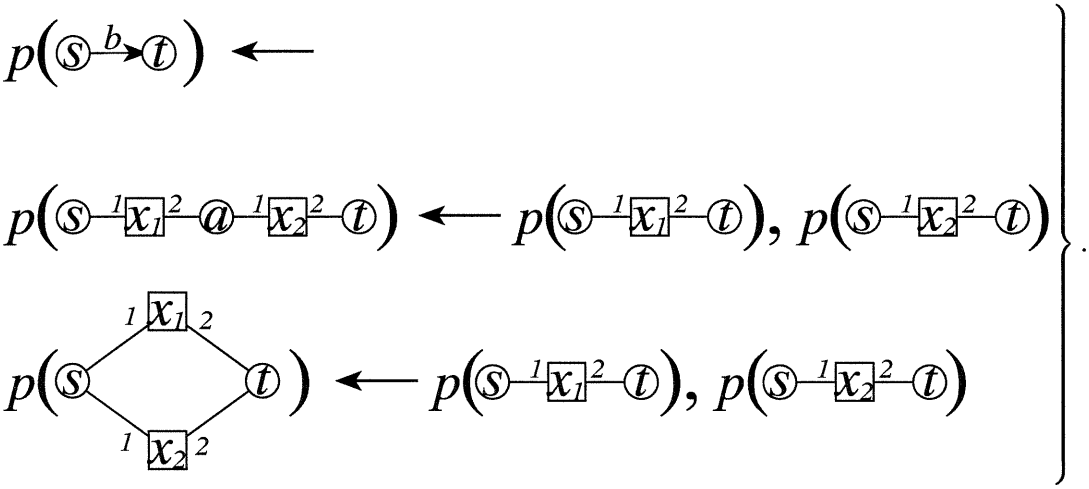


Figure 5: FGS Γ_{SP} defining the class of TTSP graphs.

(b) *Series composition*: Identify v_1 with u_2. The source and sink of the resulting graph are u_1 and v_2, respectively. The identified vertex v_1 $(= u_2)$ is labeled with a.

The source and sink of a TTSP graph are labeled with s and t, respectively. Other vertices are labeled with a and all edges are labeled with b.

In this section, we consider directed term graphs over $\langle\{a, s, t\}, \{b\}, X\rangle$ and colored-graphs over $\langle\{a, s, t\}, \{b\}\rangle$, where X consists of variable symbols of rank at most 2.

The following lemma is obvious.

Lemma 1. Let Γ_{SP} be the regular FGS in Figure 5. Then $GL(\Gamma_{SP}, p)$ is the class of TTSP graphs.

Let $g = (V, E, \varphi, \psi, H, \lambda, ports)$ be a directed term graph over $\langle\{a, s, t\}, \{b\}, X\rangle$ such that the rank of every hyperedge in H is at most 2. Then we define a directed colored-graph $\hat{g} = (\hat{V}, \hat{E}, \hat{\varphi}, \hat{\psi})$ over $\langle\{a, s, t\}, \{b\}\rangle$ as follows:

(1) $\hat{V} = V$ and $\hat{\varphi} = \varphi$.

(2) Let E_H be the set of new edges defined from H in the following way: For a hyperedge $e \in H$ with $rank(e) = 2$, let $ports(e) = (u_e, v_e)$. Then E_H contains a new edge beginning at u_e and ending at v_e for every hyperedge e in H. We define $\hat{E} = E \cup E_H$. For an edge $e \in \hat{E}$, we define $\hat{\psi}(e) = b$.

The colored-graph $\hat{g}$ is called the *transformed graph* of g. We can define the transformed graphs for undirected term graphs and undirected colored-graphs in the same way.

We say that a term graph g over $\langle\{a, s, t\}, \{b\}, X\rangle$ is a *two-terminal term multidag* with the source u and the sink v if its transformed graph $\hat{g}$ is a two-terminal multidag with the source u and the sink v. Let g and f be two-terminal term multidags. We say that f is a *reducing subgraph* of g if f is a subgraph of g except the vertex labeling, i.e., the source and sink of f are respectively labeled with s and t, but these vertices do not necessarily have the same labels in g. Let g_i be a reducing subgraph of g with the source u_i and the sink v_i for $1 \leq i \leq k$. We say that g_i and g_j $(i \neq j)$ are *pairwise disjoint* if g_i and g_j share only vertices in $\{u_i, u_j, v_i, v_j\}$ and have no edges and no hyperedges in common.

Let $g = (V, E, \varphi, \psi, H, \lambda, ports)$ be a two-terminal multidag with the source u and the sink v, and let $g_i = (V_i, E_i, \varphi_i, \psi_i, H_i, \lambda_i, ports_i)$ be a reducing subgraph of g with the sink u_i and

the sink v_i for $1 \leq i \leq k$. Assume that $g_1, \ldots, g_k$ are pairwise disjoint. Then we define a two-terminal term multidag $f = (V_f, E_f, \varphi_f, \psi_f, H_f, \lambda_f, ports_f)$ called the *reduction* of $g_1, \ldots, g_k$ in g as follows:

(7) $V_f = V_g - \bigcup_{i=1}^{k}(V_i - \{u_i, v_i\})$.
(8) $E_f = E_g - \bigcup_{i=1}^{k} E_i$.
(9) For each $1 \leq i \leq k$, let e_i be a new hyperedge with $ports_f(e_i) = (u_i, v_i)$. Let $S_f = \{e_1, \ldots, e_k\}$. Then $H_f = H_g \cup S_f - \bigcup_{i=1}^{k} H_i$. The hyperedges in S_f are distinctively labeled with new variables of rank 2.

The substitution

$$\theta = \{\lambda_f(e_1) := [g_1, (u_1, v_1)], \ldots, \lambda_f(e_k) := [g_k, (u_k, v_k)]\}$$

is called the *reducing substitution* for $g_1, \ldots, g_k$.

Let Γ be a regular FGS and p be its predicate symbol. Let g be a term graph such that its transformed graph $\hat{g}$ is in $GL(\Gamma, p)$. Let T be a refutation tree from the goal $\leftarrow p(\hat{g})$ in Γ. We define a *decomposition tree* T' of g corresponding to T in the following way: Since Γ is regular, the label of an internal vertex v in T is of the form $\leftarrow q(\hat{f})$. By the definitions of a refutation tree and a regular FGS, the colored-graph $\hat{f}$ is a subgraph of $\hat{g}$ where the vertex labelings are ignored. Hence there exists a subgraph f of g except the vertex labelings such that $\hat{f}$ is the transformed graph of f. Let T_0 be a tree obtained from T by replacing the label $\leftarrow q(\hat{f})$ of v with the label $\leftarrow q(f)$ for each internal vertex v in T. Then T' is obtained from T_0 by removing every leaf u from T_0 together with its incident edge if the parent of u is labeled with a goal $\leftarrow q(h)$ where h is a star graph for a variable of rank 2.

Example 5. Let $g = (V, E, \varphi, \psi, H, \lambda, ports)$ be a TTSP graph in Figure 6. This term graph can be regarded as a colored-graph since it contains no hyperedge. Figure 6 shows a reducing subgraph f of g and its reduction h. Then $\hat{h}$ is the transformed graph of h. Figure 6 gives a refutation tree T of f and a decomposition tree T' of h.

Theorem 2. Let Γ_{SP} be a regular FGS in Figure 5 defining the class of TTSP graphs. The refutation tree problem for (Γ_{SP}, p) can be solved in $O(\log^2 n + \log m)$ time with $O(n + m)$ processors on an EREW PRAM, where n and m are the numbers of vertices and edges of an input graph, respectively.

Proof. We give a parallel algorithm *Ref_TTSP* in Figure 7 for solving $\mathrm{RT}(\Gamma_{SP}, p)$ that uses the decomposition technique in Ref. [10]. We assume that a graph is given by edge list form. For a graph g, $|g|$ denotes the number of vertices in g.

We first show that *Ref_TTSP* computes a refutation tree T of a given graph g if any exists. Let f be the two-terminal term multidag of line 9 and assume that the decomposition is successful in line 9.

Let $\hat{f}_1, \ldots, \hat{f}_q$ be the decomposition of $\hat{f}$ and let $f_1, \ldots, f_q$ be their reducing subgraphs of f. Let f_0 be the reduction of $f_1, \ldots, f_q$ in f and let $\hat{f}_0$ be the transformed graph of f_0 provided in line 9. He and Yesha [10] showed that $\hat{f}$ is a TTSP graph if and only if $\hat{f}_0, \hat{f}_1, \ldots, \hat{f}_q$ are TTSP graphs. Since the computation starts with $g = \hat{g}$ and since lines 6, 8 and 9 reject graphs other than TTSP graphs, we see that *Ref_TTSP* recognizes the class of TTSP graphs.

Now we shall explain how to compute line 15 from W and θ when a TTSP graph g is given as an input. For a decomposition tree S and a substitution τ, let $S\tau$ be the tree obtained from S by replacing each vertex label $\leftarrow B$ of S with $\leftarrow B\tau$. Then for the set of decomposition trees W and the substitution θ, let $W\theta = \{S\theta \mid S \in W\}$. We construct a refutation tree T of g

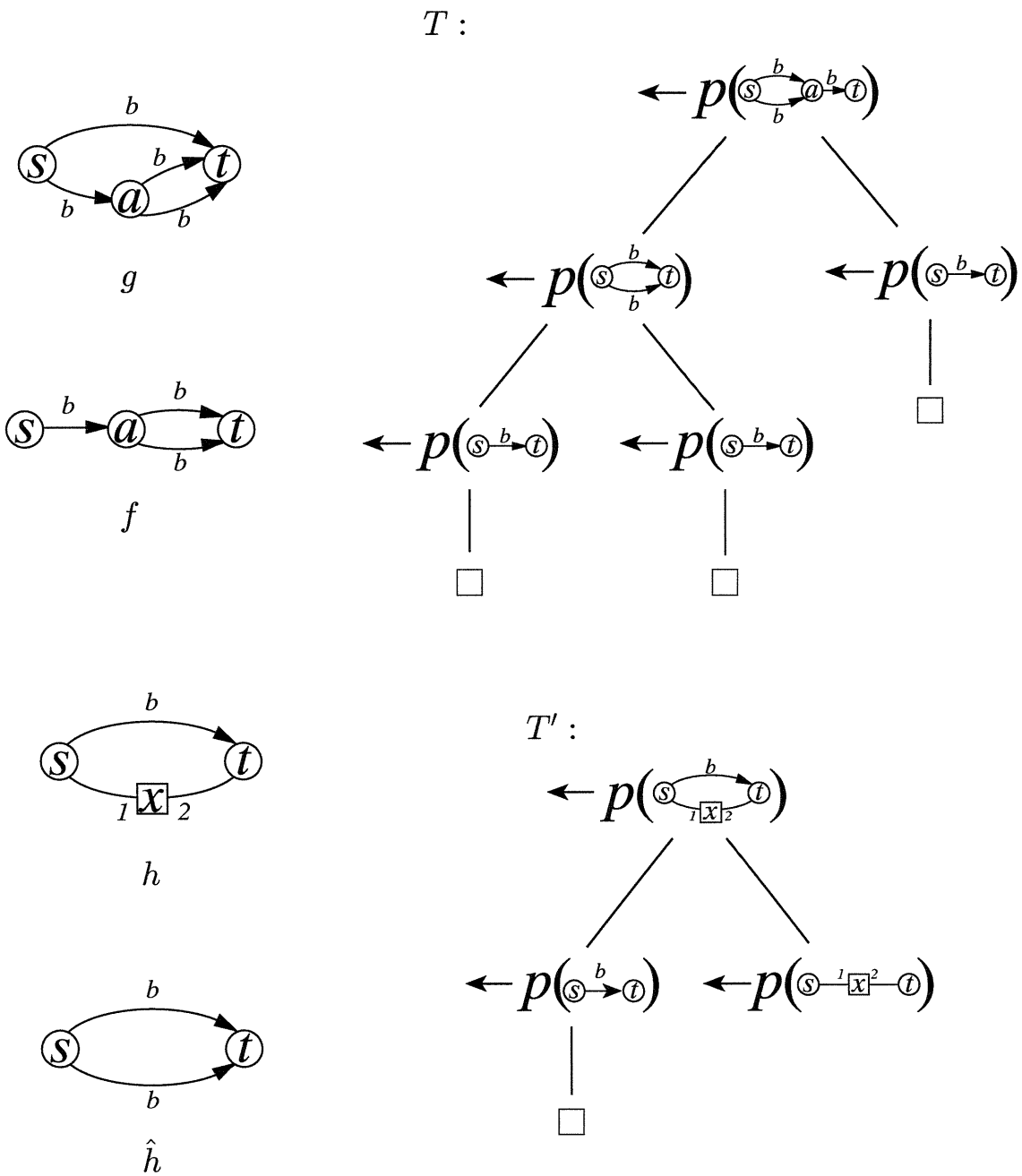


Figure 6: The graph f is a reducing subgraph of g and h is its reduction. T is a refutation tree of f and T' is a decomposition tree of h.

Ref_TTSP Algorithm

INPUT: A graph g.

OUTPUT: A refutation tree T.

```
U := {g}; i := 0; W := ∅;
while U ≠ ∅ do
    θ_i := ∅;
    for each f ∈ U pardo
        U := U − {f};
        if f is not a two-terminal term multidag
        then return "g is not a TTSP graph" and exit;
        case
            |f| < 12: if f̂ is a TTSP graph
                      then put a decomposition tree of f into W
                      else return "g is not a TTSP graph" and exit
            |f| ≥ 12: apply the decomposition algorithm by Ref. [10] to f̂;
                      if the decomposition is successful
                      then let f̂_1, ..., f̂_q be the decomposition of f̂;
                           let f_1, ..., f_q be the corresponding reducing subgraphs of f;
                           let f_0 be the reduction of f_1, ..., f_q;
                           U := U ∪ {f_0, f_1 ..., f_q};
                           put all bindings in the reducing substitution
                               for f_1, ..., f_q into θ_i
                      else return "g is not a TTSP graph" and exit
        end case
    end pardo
    i := i + 1;
end do
compute a substitution θ = ∏_{k=0}^{i} θ_k;
construct a refutation tree T of g from W and θ.
```

Figure 7: *Ref_TTSP* algorithm

from $W\theta$ in the following way: Initially, let T be a decomposition tree in $W\theta$ whose root label is $\leftarrow p(g)$. From the definitions of Γ_{SP} and $W\theta$, it is easy to see that T is unique in $W\theta$. For each leaf v of T not labeled with the empty goal, we attach the decomposition tree T_v in $W\theta$ whose root label is equal to that of v by identifying its root with v. Again from the definitions of Γ_{SP} and $W\theta$, it is clear that T_v is unique in $W\theta$. We repeat this procedure for T until all leaves of T are labeled with the empty goal. We can see that the resulting decomposition tree T is a refutation tree of g.

We illustrate the complexity of *Ref_TTSP*. In order to simplify the analysis of *Ref_TTSP*, we reduce an input multidag g to a simple directed colored-graph by performing the following operation as initialization: *Ref_TTSP* counts the number of edges between each pair of vertices and replaces them by a single edges. The resulting graph g_0 is a directed colored-graph without any multiple edges. We can see that g is a TTSP graph if and only if g_0 is a TTSP graph. Moreover, the refutation tree of g can be easily obtained from the refutation tree of g_0. The conversion from g to g_0 takes $O(\log m)$ time with $O(m)$ processors. Thus without loss of generality, we assume that g is a simple directed colored-graph.

Line 8 requires constant time. Line 9 takes in $O(\log n)$ time using $O(n+m)$ processors on an EREW PRAM by the decomposition technique in Ref. [10]. Furthermore, we can show in the same way as Ref. [10] that the while loop (lines 2–13) repeats $O(\log n)$ times. We omit the detailed argument.

Let U_j be the content of U just after the jth iteration (lines 2–13). Let d_j be the number of elements in U_j. Since $\sum_{j=0}^{O(\log n)} d_j = O(n)$, the number of bindings in $\theta = \prod_{k=0}^{O(\log n)} \theta_k$ is $O(n)$. Therefore, by using the balanced binary tree method [6], θ can be computed in $O(\log n)$ time with $O(n)$ processors. We can construct a refutation tree T of g in $O(\log n)$ time with $O(n)$ processors from W and θ by executing the above procedure. Therefore, lines 14 and 15 can be computed in $O(\log n)$ time with $O(n)$ processors. Thus *Ref_TTSP* constructs the refutation tree in $O(\log^2 n + \log m)$ time with $O(n+m)$ processors. □

4.2 Parallel Algorithm for Outerplanar Graphs

A *cutpoint* of a colored-graph is a vertex whose removal increases the number of components. A *nonseparable* colored-graph is a connected colored-graph which has at least two vertices and no cutpoints. A *block* (sometimes called *biconnected component*) of a colored-graph is a maximal nonseparable subgraph. A colored-graph is *outerplanar* if it can be embedded in the plane so that all its vertices lie on the same face.

Let g be a connected colored-graph with the cutpoints $u_1, \ldots, u_l$. A *block sequence* $(b_1, \ldots, b_k)$ of g is a sequence of all the blocks of g. A sequence $\langle f_1, v_1\rangle, \ldots, \langle f_k, v_k\rangle$ is called a *dividing sequence* of g with respect to $(b_1, \ldots, b_k)$, if each $\langle f_i, v_i\rangle$ satisfies the following conditions for $1 \le i \le k$:

(1) v_i is a vertex of b_i with $v_i \in \{u_1, \ldots, u_l\}$.

(2) $f_i = (V_i, E_i, \varphi_i, \psi_i, H_i, \lambda_i, ports_i)$ is a term graph such that $f' = (V_i, E_i, \varphi_i, \psi_i)$ is the block b_i, where the label of v_i is ignored and v_i is labeled with s. The term graph f_i is called a *dividing component* of g with a *head attachment* v_i.

(3) For each vertex w in $V_i \cap \{u_1, \ldots, u_l\}$, if there exists $j < i$ such that b_j and b_i share the vertex w, then H_i contains a hyperedge with the port w labeled with a new distinct variable of rank one.

For each $1 \le i \le k$ and each $e \in H_i$, let i_e be the maximal number less than i such that f_{i_e} and f_i share only the port of e. Then let $\theta_i = \{\lambda_i(e) := [f_{i_e}, ports(e)] \mid e \in H_i\}$. The substitution $\theta = \prod_{1 \le i \le k} \theta_i$ is called a *dividing substitution* for g.

We can see from Theorem 1 that the following lemma holds.

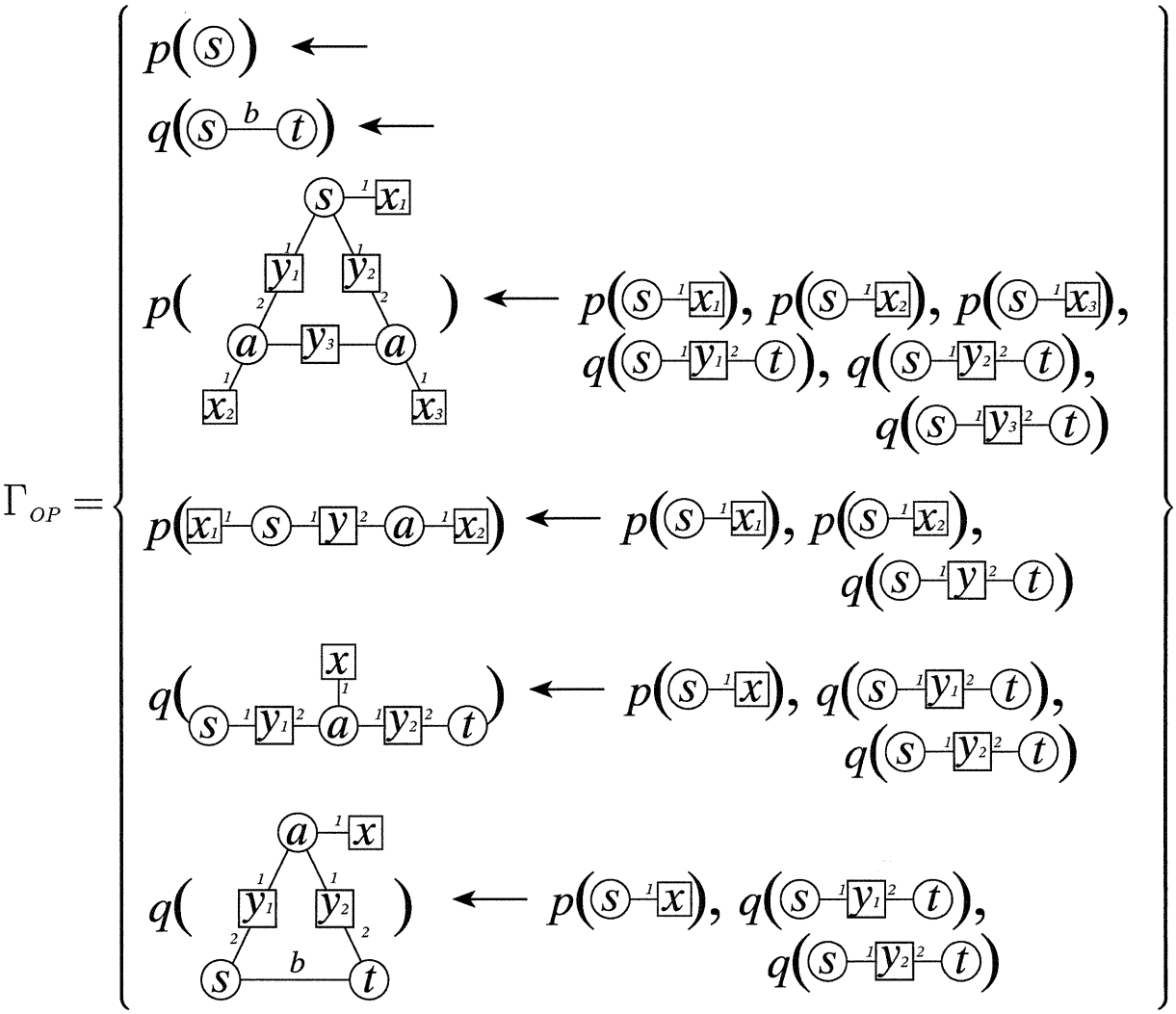


Figure 8: FGS Γ_{OP} defining the class of outerplanar graphs.

Lemma 2. Let Γ_{OP} be an FGS in Figure 8. Then $GL(\Gamma_{OP}, p)$ is the set of undirected outerplanar colored-graphs.

Theorem 3. Let Γ_{OP} be an FGS in Figure 8 defining the class of outerplanar graphs. The refutation tree problem for (Γ_{OP}, p) can be solved in $O(\log^2 n)$ time with $O(n+m)$ processors on an EREW PRAM, where n and m are the numbers of vertices and edges of an input connected colored-graph.

Proof. We give a parallel algorithm for solving $\mathrm{RT}(\Gamma_{OP}, p)$ in Figure 9 using the *Ref_TTSP* algorithm. We assume that a graph is given by edge list form.

By the definition of a dividing component of g, we can see that g is an outerplanar colored-graph if and only if the transformed graph of every dividing component of g is an outerplanar colored-graph.

Let f be a dividing component of g with a head attachment v and $\{w, v\}$ be an edge of f. An assignment of distinct integers to the vertices of the transformed graph $\hat{f}$ is called an *wv-numbering* of f for $\{w, v\}$ if two adjacent vertices w and v are the lowest- and highest-numbered, respectively, and every other vertex is adjacent to both a lower-numbered and a higher-numbered vertex. Since $\hat{f}$ is biconnected, it is easy to see that $\hat{f}$ has an wv-numbering of f for the edge $\{w, v\}$.

Ref_OUTER Algorithm
INPUT: A connected colored-graph g.
OUTPUT: A refutation tree T.

let $(b_1, \ldots, b_k)$ be a block sequence of g;
let $\langle f_1, v_1 \rangle, \ldots, \langle f_k, v_k \rangle$ be a dividing sequence of g with respect to $(b_1, \ldots, b_k)$;
let θ be the dividing substitution for g;
$W := \emptyset$; $\tau := \emptyset$;
for each $\langle f_i, v_i \rangle$, $1 \leq i \leq k$ **pardo**
    Compute $w_i v_i$-numbering F_{w_i, v_i} for an edge $\{w_i, v_i\}$ of $\hat{f}_i$;
    case
        $|f_i| > 2$: **apply** the decomposition algorithm *DECOM* to f_i;
            if the decomposition is successful
            then let f_i^1, f_i^2, f_i^3 be the resulting reducing subgraphs of f_i;
                let f_i^0 be the reduction of f_i^1, f_i^2, f_i^3;
                put all bindings in the reducing substitution for f_i^1, f_i^2, f_i^3 into τ;
                put a decomposition tree of f_i^0 into W;
                apply *Ref_TTSP* to each F_{w_i, v_i}-directed graph $\tilde{f}_i^j$ $(1 \leq j \leq 3)$ of f_i^j;
                let H_i^1, H_i^2, H_i^3 be the resulting decomposition trees of
                $\tilde{f}_i^1, \tilde{f}_i^2, \tilde{f}_i^3$, respectively;
                let T_i^1, T_i^2, T_i^3 be the corresponding decomposition tree of
                f_i^1, f_i^2, f_i^3 to H_i^1, H_i^2, H_i^3, respectively;
                $W := W \cup \{T_i^1, T_i^2, T_i^3\}$;
            else return "g is not an outerplanar graph" and **exit**;
        $|f_i| \leq 2$: **put** a decomposition tree of f_i to W
    end case
end pardo
Construct the refutation tree T of g from W and τ.

Figure 9: *Ref_OUTER* algorithm

Claim 1. Let f be a dividing component of g with a head attachment v such that $|f| > 2$ and $\{w, v\}$ be an edge of f. Let $F_{w,v}$ be an wv-numbering of f for $\{w, v\}$. Assume that w' is the smallest vertex with $w' \neq w$ with respect to $F_{w,v}$. If g is outerplanar, then we can decompose f into the reducing subgraphs f^1, f^2, f^3 of f with the sources w, w, w' and the sinks v, w', v, respectively.

Proof of Claim 1: From the definition of a dividing component of g, the transformed graph $\hat{f}$ of f is a block of g where the label of v is ignored. Hence if g is an outerplanar colored-graph, then $\hat{f}$ is biconnected. From the selection of w and w', there exist reducing subgraphs f^1, f^2, f^3 of f with the sources w, w, w' and the sinks v, w', v, respectively, such that any edge in f is in either f^1, f^2 or f^3. If g is outerplanar, then we can decompose f into the reducing subgraphs f^1, f^2, f^3 of f with the sources w, w, w' and the sinks v, w', v, respectively. □

Let $\hat{f} = (\hat{V}_f, \hat{E}_f, \hat{\varphi}_f, \hat{\psi}_f)$ and $F_{w,v}$ be a wv-numbering of f for $\{w, v\}$. Then $F_{w,v}$ induces a direction on the edges of $\hat{f}$. We define a directed colored-graph $\tilde{f} = (\tilde{V}_f, \tilde{E}_f, \tilde{\varphi}_f, \tilde{\psi}_f)$ as follows:

(1) $\tilde{V}_f = \hat{V}_f$ and $\tilde{\varphi}_f = \hat{\varphi}_f$.

(2) Let u, u' be any two vertices in $\hat{V}_f$ such that $F_{w,v}(u) < F_{w,v}(u')$. Then $\{u, u'\}$ is in $\hat{E}_f$ if and only if an ordered edge (u, u') is in $\tilde{E}_f$. For an edge (u, u') in $\tilde{E}_f$, $\tilde{\psi}_f((u, u')) = \hat{\psi}_f(\{u, u'\})$.

The directed colored-graph $\tilde{f}$ is called the $F_{w,v}$-*directed graph* of f.

Let h be a two-terminal term multidag. We say that h is of *chain-type* if there exists a path containing all vertices in the transformed graph $\hat{h}$.

Claim 2. Let f be a dividing component of g with a head attachment v such that $|f| > 2$ and $\{w, v\}$ be an edge of f. Let $F_{w,v}$ be an wv-numbering of f for $\{w, v\}$. Assume that f^1, f^2, f^3 are the reducing subgraphs of f obtained in Claim 1. Then f is an outerplanar colored-graph if and only if the $F_{w,v}$-directed graphs $\tilde{f}^1, \tilde{f}^2, \tilde{f}^3$ of f^1, f^2, f^3 are chain-type TTSP graphs, respectively.

Proof of Claim 2. We can see that if $\tilde{f}^1, \tilde{f}^2, \tilde{f}^3$ are chain-type TTSP graphs then $\hat{f}$ is outerplanar. We show that if $\hat{f}$ is outerplanar then $\tilde{f}^1, \tilde{f}^2, \tilde{f}^3$ are chain-type TTSP graphs. By Claim 1, if g is an outerplanar colored-graph, then we can decompose f into the reducing subgraphs f^1, f^2, f^3 of f with the sources w, w, w' and the sinks v, w', v, respectively. From the definitions of $F_{w,v}$ and $F_{w,v}$-directed graph $\tilde{f}$ of f, $\tilde{f}$ is a directed colored-graph with no cycle. Hence $\tilde{f}^1, \tilde{f}^2, \tilde{f}^3$ of f are two-terminal directed colored-graphs. Since $\hat{f}$ is an outerplanar colored-graph, there exists a path containing all vertices in f^j for each reducing subgraph f^j $(1 \leq j \leq 3)$. Hence, if f is outerplanar then $\tilde{f}^1, \tilde{f}^2, \tilde{f}^3$ is chain-type TTSP graphs. □

DECOM is a procedure of selecting the smallest vertex w' of f with $w' \neq w$ with respect to $F_{w,v}$ and decomposing f into reducing subgraphs f^1, f^2, f^3 with the sources w, w, w' and the sinks v, w', v, respectively, such that $\tilde{f}^1, \tilde{f}^2, \tilde{f}^3$ are chain-type TTSP graphs, if possible.

Next, we illustrate some implementation details. The input to the *Ref_OUTER* algorithm is an undirected connected colored-graph $g = (V, E, \varphi, \psi)$ given by edge list form with $|V| = n$ and $|E| = m$.

The all blocks $b_1, \ldots, b_k$ of g are computed in $O(\log^2 n)$ time with $O(n + m)$ processors by the block-finding algorithm in Ref. [20]. The block-finding algorithm in Ref. [20] uses any spanning tree of g in order to find the all blocks of g. Then line 16 can be computed in $O(\log^2 n)$ time with $O(n+m)$ processors. Line 17 can be implemented as follows: Let S be an undirected spanning tree constructed in order to find the blocks of g in line 16. By selecting a cutpoint of g in S, we make S a rooted tree. Let $u_1, \ldots, u_l$ be the cutpoints of g. Let each block b_i $(1 \leq i \leq k)$ contain cutpoints $v_1^i, \ldots, v_{p_i}^i$ of g. By applying the pointer jump technique

in Ref. [20] to S, we can identify the vertex v_1^i such that a subtree of S rooted at v_1^i contains $v_2^i, \ldots, v_{p_i}^i$ in $O(\log n)$ time with $O(n)$ processors. Hence for the block sequence $(b_1, \ldots, b_k)$ of g, we can construct a dividing sequence $\langle f_1, v_1^1 \rangle, \ldots, \langle f_k, v_1^k \rangle$ of g in $O(\log n)$ time with $O(n)$ processors. Then line 17 takes $O(\log n)$ time with $O(n+m)$ processors.

From the dividing sequence of g computed in line 17, we can compute line 18 in $O(\log n)$ time with $O(n+m)$ processors.

Next, we show that lines 20–26 can be computed in $O(\log^2 n)$ time with $O(n+m)$ processors. It takes clearly constant time in line 24. Hence we show that lines 21 and 23 can be computed in $O(\log^2 n)$ time with $O(n+m)$ processors. Let $\langle f_1, v_1 \rangle, \ldots, \langle f_k, v_k \rangle$ be the block sequence of g computed in line 17. We assume that, for each dividing component f_i of g with a head attachment v_i, we denote the numbers of vertices and edges of f_i by n_i and m_i, respectively. The $w_i v_i$-numbering F_{w_i,v_i} for an edge $\{w_i, v_i\}$ of f_i can be computed in $O(\log^2 n_i)$ time with $O(n_i + m_i)$ processors by the algorithm in Ref. [13]. *DECOM* can be implemented in constant time if g is an outerplanar graph. By using the *Ref_TTSP* algorithm, we can construct the decomposition trees of the resulting reducing components f_i^1, f_i^2, f_i^3 of f_i in $O(\log^2 n_i)$ time with $O(n_i + m_i)$ processors.

Since $\sum_{1 \le i \le k} n_i = O(n)$ and $\sum_{1 \le i \le k} m_i = O(m)$, lines 20–26 can be computed in $O(\log^2 n)$ time with $O(n+m)$ processors. In line 27, we can compute in $O(\log n)$ time with $O(n)$ processors by using the same procedure in line 15 of the *Ref_TTSP* algorithm. Thus we can solve the refutation tree problem for outerplanar graphs in $O(\log^2 n)$ time with $O(n+m)$ processors on an EREW PRAM. □

5 Conclusion

We defined the formal graph system as a new formal system for generating graphs. The refutation tree problem for an FGS is to construct a refutation tree of a graph defined by the FGS. We can decide whether an input graph is generated by the FGS by solving the refutation tree problem for the FGS. We presented two parallel algorithms solving the refutation tree problem for two classes of graphs, TTSP graphs and outerplanar graphs, with a linear number of processors on an EREW PRAM. The decomposition techniques presented in this paper are useful for solving the refutation tree problem for some regular FGSs in **NC**. By using a parallel algorithm for recognizing series-parallel graphs, Xin He [9] has presented efficient parallel algorithms for solving three basic problems on series-parallel graphs: 3-coloring, depth-first spanning tree, and breadth-first spanning tree. By using the refutation tree, we may be able to design efficient parallel algorithms for a large number of **NP**-complete problems when these problems are restricted to the class of outerplanar graphs.

We showed that the class of graph languages generated by HRGs can be also defined by the regular FGSs. Vogler [21] has shown that HRGs are characterized by vertex labeling boundary node label controlled (BNLC) grammars of bounded nonterminal degree. The class of graphs generated by BNLC grammars of bounded nonterminal degree can be also defined by the regular FGSs. But the relation between FGSs and BNLC grammars is not yet clear.

We are interested in the refutation tree problem for an FGS Γ satisfying the following condition (i.e., see Figures 10 and 11): For a variable $x \in X$, there exist atoms A in Γ such that the number of hyperedges labeled with x in A is greater than two. It is necessary to solve the graph isomorphism problem in order to compute the refutation tree problem for Γ. We need to clarify the relation between the refutation tree problem for Γ and the graph isomorphism problem.

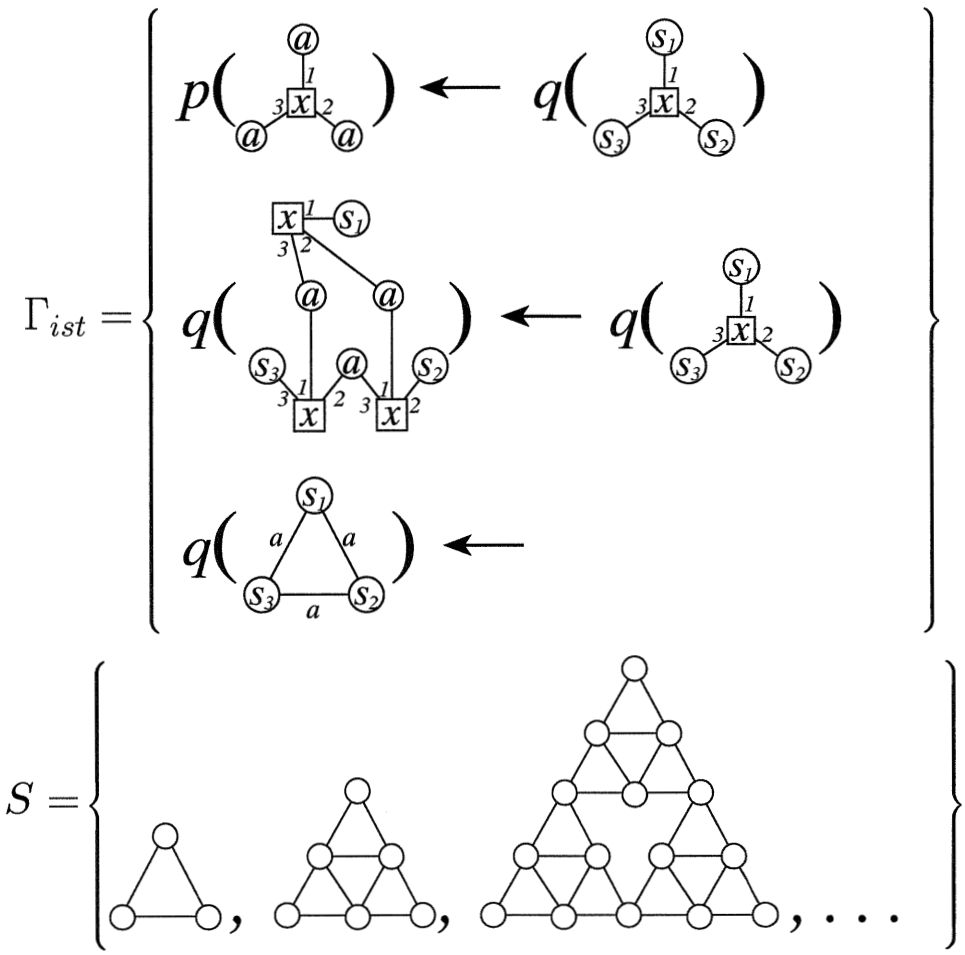


Figure 10: FGS Γ_{ist} generating the set of Sierpinski triangles.

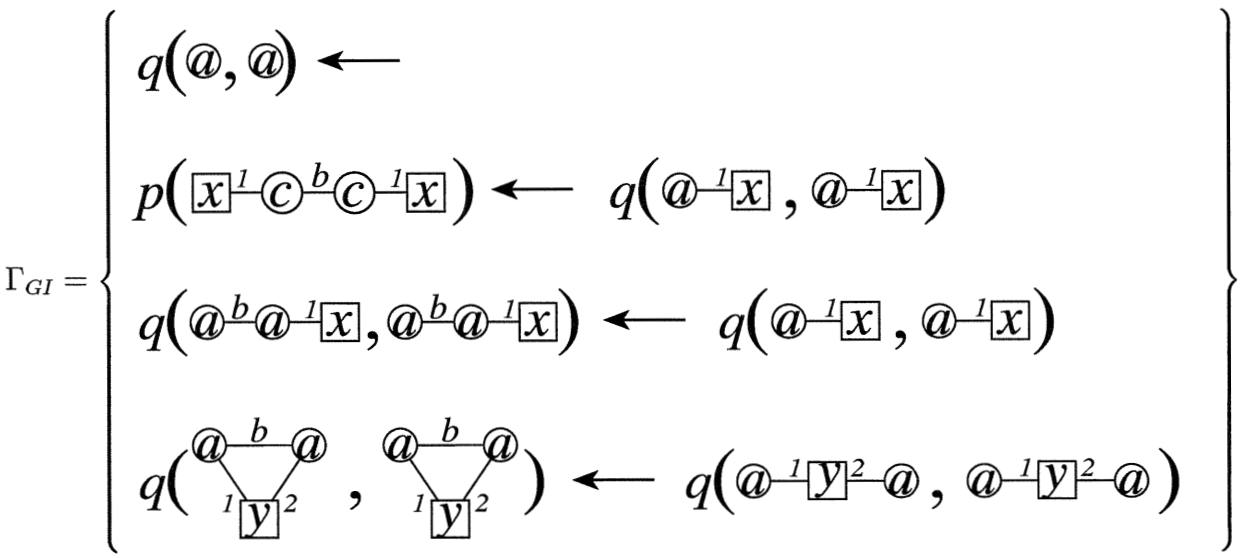


Figure 11: FGS Γ_{GI}

6 Acknowledgment

We would like to thank Hiroki Arimura and Eizyu Hirowatari for many helpful comments and discussions.

References

[1] Arikawa S., Miyano S., Shinohara A., Shinohara T. and Yamamoto A., "Algorithmic learning theory with elementary formal systems", *IEICE Transactions on Information and Systems*, **E75-D**, 4, pp. 405–414 (1992).

[2] Arikawa S., Shinohara T. and Yamamoto A., "Learning elementary formal systems", *Theoretical Computer Science*, **95**, pp. 97–113 (1992).

[3] Bauderon M. and Courcelle B., "Graph expressions and graph rewritings", *Mathematical Systems Theory*, **20**, pp. 83–127 (1987).

[4] Diks K., Hagerup T. and Rytter W., "Optimal parallel algorithms for the recognition and colouring outerplanar graphs", *Mathematical Foundations of Computer Science, Lecture Notes in Computer Science*, **379**, pp. 207–217 (1989).

[5] Drewes F. and Kreowski H. J., "A note on hyperedge replacement", *Graph grammars and their application to computer science*, Lecture Notes in Computer Science, **532**, pp. 1–11 (1991).

[6] Gibbons A. and Rytter W., *Efficient Parallel Algorithms*, Cambridge University Press (1988).

[7] Habel A. and Kreowski H. J., "Some structural aspects of hypergraph languages generated by hyperedge replacement", *Proceedings of the 4th Annual Symposium on Theoretical Aspects of Computer Science*, Lecture Notes in Computer Science, **247**, pp. 207–219 (1987).

[8] Habel A., Kreowski H. J. and Vogler W., "Decidable boundedness problems for sets of graphs generated by hyperedge-replacement", *Theoretical Computer Science*, **89**, pp. 33–62 (1991).

[9] He X., "Efficient parallel algorithms for series parallel graphs", *Journal of Algorithms*, **12**, pp. 409–430 (1991).

[10] He X. and Yesha Y., "Parallel recognition and decomposition of two terminal series parallel graphs", *Information and Computation*, **75**, pp. 15–38 (1987).

[11] Lautemann C., "The complexity of graph languages generated by hyperedge replacement", *Acta Informatica*, **27**, pp. 399–421 (1990).

[12] Lloyd J. W., *Foundations of Logic Programming, Second, Extended Edition*, Springer-Verlag (1987).

[13] Maon Y., Schieber B. and Vishkin U., "Parallel ear decomposition search (EDS) and *st*-numbering in graphs", *VLSI Algorithms and Architectures – AWOC 86*, Lecture Notes in Computer Science, **227**, pp. 34–45 (1986).

[14] Pavlidis T., "Linear and context-free graph grammars", *Journal of the Association for Computing Machinery*, **19**, 1, pp. 11–22 (1972).

[15] Rozenberg G. and Welzl E., "Boundary NLC graph grammars–Basic definitions, normal forms, and complexity", *Information and Control*, **69**, pp. 136–167 (1986).

[16] Rozenberg G. and Welzl E., "Graph theoretic closure properties of the family of boundary NCL graph languages", *Acta Informatica*, **23**, pp. 289–309 (1986).

[17] Rytter W. and Szymacha T., "Parallel algorithms for a class of graphs generated recursively", *Information Processing Letters*, **30**, pp. 225–231 (1989).

[18] Slisenko A. O., "Context-free grammars as a tool for describing polynomial-time subclasses of hard problems", *Information Processing Letters*, **14**, 2, pp. 52–56 (1982).

[19] Smullyan R. M., *Theory of Formal Systems*, Princeton Univ. Press (1961).

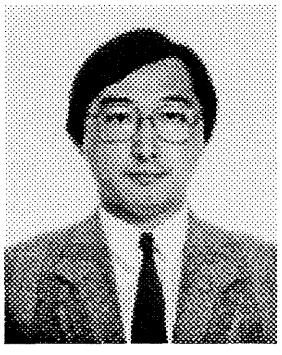
[20] Tarjan R. E. and Vishkin U., "An efficient parallel biconnectivity algorithm", *SIAM Journal on Computing*, **14**, 4, pp. 862–874 (1985).

[21] Vogler W., "On hyperedge replacement and BNLC graph grammars", *Graph-Theoretic Concepts in Computer Science – WG'89*, Lecture Notes in Computer Science, **411**, pp. 78–93 (1990).

About the Authors

Tomoyuki Uchida (内田智之) was born in Kumamoto on March 3, 1966. He received the B.S. degree in 1989 in Mathematics and the M.S. degree in 1991 in Information Systems from Kyushu University. He is now a graduate student of Doctor Cource at Department of Information Systems, Kyushu University. He is studying the parallel algorithms and computational complexity.

Takayoshi Shoudai (正代隆義)was born in Fukuoka on December 30, 1961. He received the B.S. degree in 1986 and the M.S. degree in 1988 in Mathematics from Kyushu University. Presently, he is a lecturer at Faculty of Engineering, Yamaguchi University, Ube. His research Interests are parallel algorithms, probabilistic algorithms and computational complexity.



Satoru Miyano (宮野　悟) was born in Oita on December 5, 1954. He received the B.S. in 1977, the M.S. degree in 1979 and the Dr. Sci. in 1984 all in Mathematics from Kyushu University. Presently, he is a Professor of Research Institute of Fundamental Information Science, Kyushu University. His present interests include parallel algorithms, computational complexity, computational learning theory and Genome Informatics.

Research Institute of Fundamental Information Scinece, Kyushu University 33, Fukuoka 812, Japan.