Polynomial Time Algorithm Solving the Refutation Tree Problem for Formal Graph Systems

Uchida, Tomoyuki
Department of Information Systems, Kyushu University

Shoudai, Takayoshi
College of General Education, Kyushu University

Miyano, Satoru
Research Institute of Fundamental Information Science, Kyushu University

https://doi.org/10.5109/3069

出版情報：Bulletin of informatics and cybernetics. 26 (1/2), pp.55-74, 1993-09-30. Research Association of Statistical Sciences
バージョン：
権利関係：

Polynomial Time Algorithm Solving the Refutation Tree Problem for 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

September 30, 1993

† : Department of Information Systems, Kyushu University 39, Kasuga 816, Japan.
‡ : College of General Education, Kyushu University, Fukuoka 810, Japan
∗: Research Institute of Fundamental Information Science, Kyushu University 33, Fukuoka 812, Japan.

Abstract

The refutation tree problem is to compute a refutation tree which is associated with the structure of a graph generated by a formal graph system (FGS). We present subclasses of FGSs, called simple FGSs, size-bounded simple FGSs and bounded simple FGSs. In order to show that the refutation tree problem for simple FGSs can be solved in polynomial time, we prove that the refutation tree problem for size-bounded simple FGSs is NC^2-reducible to the graph ismorphism problem. For bounded simple FGSs, we show that the refutation tree problem is in NC^2. We present an FGS Γ_{GI} and show that the graph isomorphism problem is log-space reducible to the graph isomorphism problem for Γ_{GI}.

1 Introduction

A formal graph system (FGS) [21] is a kind of logic programs which deals with graphs just like terms [10]. By regarding terms as trees, conventional logic programs can be directly simulated by FGSs. Moreover, an elementary formal system [2, 3, 18], that is a logic program on strings, is also a special case of FGSs. We have shown in [21] that a family of graphs is generated by a hyperedge replacement grammar (HRG) [7, 8] if and only if it is defined by a regular FGS that is an FGS of a very simple form. This result implies that some interesting families of graphs are definable by regular FGSs. For example, regular FGSs generate trees, two-terminal series parallel graphs, the homeomorphisms of a given graph, outerplanar graphs, graphs of cyclic bandwidth ≤ 2 and k-decomposable graphs [7, 8, 13]. Thus FGSs have a diversity of expressibility.

A refutation tree [12, 16, 19] describes how a term is generated by applying the rules of a logic program. A refutation tree of a graph for an FGS is defined in a similar way. It represents the logical structure of the graph in the FGS and simultaneously describes a decomposition of the graph explicitly. Therefore, it is useful to employ FGSs in designing efficient algorithms

for graph problems by using refutation trees. The refutation tree problem is to compute a refutation tree of a graph generated by an FGS. Thus the refutation tree problem is an important issue in designing efficient algorithms for graphs defined by FGSs.

In [21], we have considered the family of two-terminal series parallel graphs and the family of outerplanar graphs both of which are definable by regular FGSs. We have shown efficient parallel algorithms solving the refutation tree problem for these regular FGSs.

A graph is generated from an FGS by replacing hyperedges labeled with the same variable by the same graph at a time. Hence, in general, it requires to solve the graph isomorphism problem in order to compute refutation trees. It is well known that the graph isomorphism problem is in NP, and it does not seem to be solvable in polynomial time. However, for graphs of constantly bounded valence, the graph isomorphism problem allows a polynomial-time algorithm [1, 11]. The purpose of this paper is to find subclasses of FGSs for which the refutation tree problem is solvable in polynomial time.

In Section 3, we present three subclasses of FGSs: simple FGSs, size-bounded simple FGSs and bounded simple FGSs. We first prove that the refutation tree problem for simple FGSs is solvable in polynomial time. In this proof, we give a polynomial-time algorithm which constructs and solves a path system. In constructing a path system, it is necessary to solve the graph isomorphism problem. However, since graphs generated by a simple FGS are of constantly bounded valence, the graph isomorphism problem for these graphs can be solved in polynomial time [11]. This polynomial-time algorithm does not seem to be efficiently parallelizable, since the problem of solving a path system is P-complete [4].

We say that a problem R is NC^k-reducible to a problem S if R can be solved with uniform $O((\log n)^k)$-depth circuits using oracles for S [5]. A generation tree of a path system depicts how an instance is derived from the path system. Rytter [14] has shown that a path system Q can be solved in NC^2, when a generation tree of Q is of polynomial size. For a size-bounded simple FGS, we can show that generation trees of the path systems constructed by our polynomial-time algorithm are of polynomial size. Therefore, the refutation tree problem for a size-bounded simple FGS is NC^2-reducible to the graph isomorphism problem for graphs of constantly bounded valence. On the other hand, we construct in Section 5 a size-bounded FGS Γ_{GI} such that the graph isomorphism problem is log-space reducible to the refutation tree problem for Γ_{GI}. These results imply that the complexity of the refutation tree problem for these FGSs centers around the complexity of the graph isomorphism problem. Hence, for size-bounded simple FGSs generating undirected trees of constantly bounded valence, we show that the refutation tree problem for these FGSs is in NC^2 by employing the NC^2 algorithm in [1] for the graph isomorphism problem. Lingas [9] has shown that the graph isomorphism problem is in NC^3, when an instance is restricted to connected graphs of constantly bounded valence and having a constantly bounded separator. We can show that the refutation tree problem for size-bounded simple FGSs is in NC^3, when an input graph is restricted to the family of connected graphs of constantly bounded valence and having a constantly bounded separator. A bounded simple FGS is a size-bounded simple FGS Γ such that it does not require to solve the graph isomorphism problem in order to compute a refutation tree of a graph defined by Γ. We show that the refutation tree problem for bounded simple FGSs is in NC^2.

The parsing problem for a context-free graph grammar (CFGG) is to construct a parse tree of an input graph. As CFGGs, we deal with those defined in [17]. Rytter and Szymacha [15] have proved that the parsing problem for a CFGG is in NC^2. In Section 4, we prove that the parsing problem for a CFGG is NC^1-reducible to the refutation tree problem for a bounded simple FGS. This means that the result on bounded simple FGSs in Section 3 includes their result in [15]. This result also asserts that efficient parallel algorithms may exist for a large number of NP-complete problems when these problems are restricted to the families of graphs

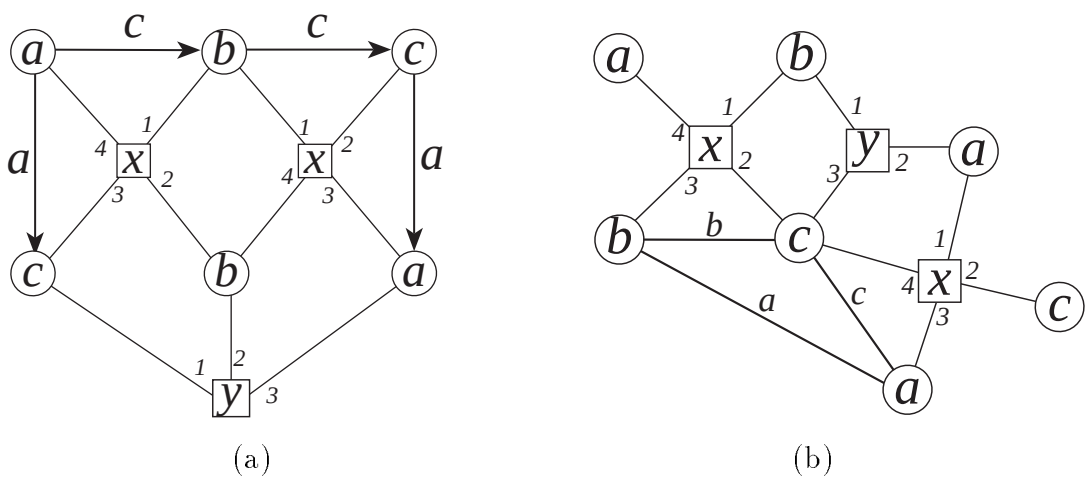

Figure 1: Term graphs. A hyperedge is represented by a box with lines to its ports. The order of the ports is indicated by numbers at these lines.

definable by bounded simple FGS.

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*.

(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 identity ground term graphs with colored-graphs. 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.

Example 1. We draw directed and undirected term graphs as in Fig. 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 $p, q, \cdots$ 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. Especially, a graph rewriting rule of the form $A \leftarrow$ is called a *fact*. 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 by $\pi^*((v_1, \ldots, v_m)) = (\pi(v_1), \ldots, \pi(v_m))$.

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 $e \in H_g$ 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 Fig. 2, and $\theta = \{x := [g, (u_1, u_2)], y :=$

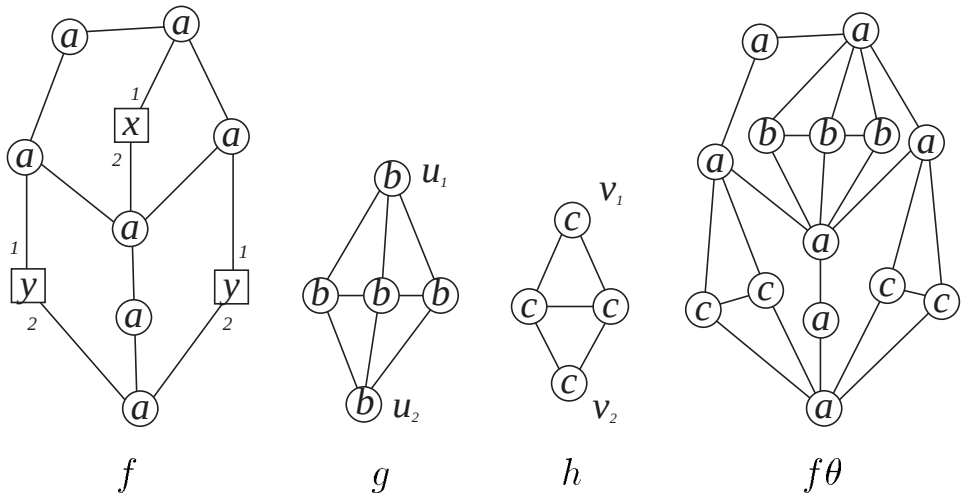

Figure 2: An example of an instance $f\theta$ with $\theta = \{x := [g, (u_1, u_2)], y := [h, (v_1, v_2)]\}$

$[h, (v_1, v_2)]\}$ 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 Fig. 2.

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' then g_1 is called a *variant* of g_2. That is, let F be a one-to-one and onto mapping from X to X. g_1 is a term graph (atom) obtained from g_2 by replacing a label $x \in X$ of each hyperedge in g_2 with $F(x)$. 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. 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 2. 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 $\Gamma \vdash A \leftarrow B_1, \ldots, B_{i-1}, C_1, \ldots, C_m, B_{i+1}, \ldots, B_n$.

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.

Definition 3. 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 Simple FGSs, Size-Bounded Simple FGSs and Bounded Simple FGSs

This section defines simple FGSs, size-bounded simple FGSs and bounded simple FGSs. We investigate the complexities of the refutation tree problems for these FGSs.

Definition 4. Let Δ be a subset of Λ called an *enclosure set*. A term graph $g = (V, E, \varphi, \psi, H, \lambda, ports)$ is *simple* with Δ if g satisfies the following conditions:

(1) If a vertex v is a port of two distinct hyperedges in H, then any edge in E having v as an endpoint is not labeled with any symbol in Δ.
(2) For any hyperedge $e \in H$, if a port v of e is an endpoint of an edge labeled with a symbol in Δ, then every edge adjoining to v is labeled with a symbol in Δ.

Definition 5. Let Δ be an enclosure set. An FGS Γ is said to be *simple* with Δ if, for each predicate symbol p of Γ with the arity n, there is a sequence $S(p)$ of n ground term graphs in $\mathcal{T}(\Sigma, \Delta)$ such that each graph rewriting rule

$$q_0(g^1_0, \ldots, g^{l_0}_0) \leftarrow q_1(g^1_1, \ldots, g^{l_1}_1), \ldots, q_k(g^1_k, \ldots, g^{l_k}_k)$$

in Γ satisfies the conditions (1)–(4): For each $0 \le i \le k$, let $S(q_i) = (f^1_i, \ldots, f^{l_i}_i)$.

(1) For each $0 \le i \le k$ and $1 \le j \le l_i$, the subgraph of g^j_i induced by the all edges labeled with a symbol in Δ is isomorphic to the ground term graph f^j_i.
(2) For $1 \le j \le l_0$, the term graph $g^j_0 = (V^j_0, E^j_0, \varphi^j_0, \psi^j_0, H^j_0, \lambda^j_0, ports^j_0)$ is simple with Δ.
(3) For $1 \le i \le k$ and $1 \le j \le l_i$, the term graph $g^j_i = (V^j_i, E^j_i, \varphi^j_i, \psi^j_i, H^j_i, \lambda^j_i, ports^j_i)$ is a simple with Δ satisfying the following conditions:
   (i) Every edge in E^j_i is labeled with a symbol in Δ.

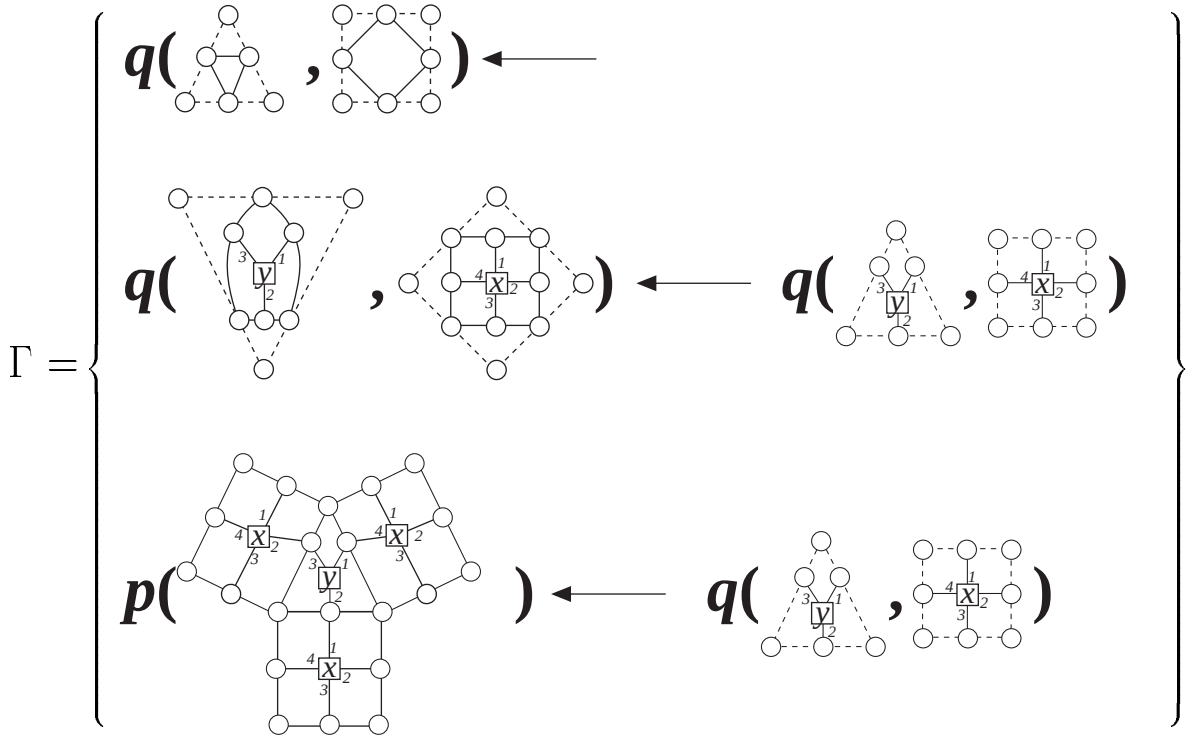

Figure 3: Simple FGS Γ. The broken lines represent the edges labeled with elements in the enclosure set.

(ii) There exists a term graph g_0^l $(1 \leq l \leq l_0)$ which has a subgraph isomorphic to g_i^j where the vertex and edge labelings are ignored.

(4) $\bigcup_{j=1}^{l_0} \lambda_0^j(H_0^j) = \bigcup_{i=1}^{k} \bigcup_{j=1}^{l_i} \lambda_i^j(H_i^j)$.

Example 3. We draw a simple FGS Γ as Fig. 3. An edge labeled with a symbol in the enclosure set is shown by a broken line.

Let $g = (V, E, \varphi, \psi)$ be a ground term graph. For a vertex $v \in V$, the *degree* of v is the number of edges having v as an endpoint and is denoted by $\deg(v)$. If g is directed, the degree of v is the number of edges beginning from v or ending to v. Then the *valence* of g is $\max\{\deg(v) \mid v \in V\}$. The following lemma is obvious from the definition.

Lemma 1. Let Γ be a simple FGS and p be a predicate symbol in Γ. Then the term graphs in $GL(\Gamma, p)$ are of constantly bounded valence.

Theorem 1. Let Γ be a simple FGS and p be a unary predicate symbol in Γ. When an input graph is connected, $\mathrm{RT}(\Gamma, p)$ is solvable in polynomial time.

Proof. We consider a connected ground term graph $g = (V, E, \varphi, \psi)$ with $V = \{1, \ldots, n\}$ as an input graph. Let $V_{E'}$ be the set of endpoints of the edges in E' for a subset $E' \subseteq E$. We denote the subgraph of g formed from E' by $\ll E' \gg = (V_{E'}, E', \varphi', \psi')$ and call it the *edge-induced subgraph* of E'. For a sequence α, $set(\alpha)$ denotes the set of all elements in α.

In this proof, an algebraic approach of path systems[14] is taken. A path system is given by a 4-tuple $Q = (N, T, GEN, S)$ [6, 14], where N is a finite set of vertices, T is a subset of N whose vertices are called terminal vertices, GEN is a function whose arguments are vertices in N and whose values are subsets of N, and S is a vertex of N called the goal. A *derivation* of S is a sequence of pairs (D_i, C_i) $(i = 0, 1 \ldots)$ satisfying the following conditions:

(1) D_i and C_i are sequences of vertices in N, and $D_0 = S$.

(2) Let $D_i = (a_1, \ldots, a_n)$ and a_m be the left-most vertex in D_i not in T. Let $C_i = (b_1, \ldots, b_k)$. Then $D_{i+1} = (a_1, \ldots, a_{m-1}, b_1, \ldots, b_k, a_{m+1}, \ldots, a_n)$.

Let $F = \{(D_i, C_i)\}_{0 \le i \le l}$ be a finite derivation of S such that $set(D_l) \subseteq T$. The *generation tree* of F is a tree T_l defined as follows:

(3) Every vertex is in N.

(4) The root is the goal S.

(5) T_0 is a tree consisting of the only vertex S. For $0 \le i < l$, T_{i+1} is a tree obtained by applying (D_i, C_i) to the tree T_i as follows: Let a_m be the left-most vertex in D_i not in T and $C_i = (b_1, \ldots, b_k)$. Then T_{i+1} is obtained by adding new k vertices $b_1, \ldots, b_k$ as the children of a_m.

The system Q is *solvable* if S is generated from T using the function GEN, that is, there is a finite derivation $\{(D_i, C_i)\}_{0 \le i \le l}$ of S on Q such that $set(D_l) \subseteq T$.

First, we number the edges of g. We give a procedure *MakeAtom* in Fig. 4 which computes the set of sequences $[q, a_1, \ldots, a_l]$ satisfying the following conditions for a graph rewriting rule $r = q_0(g_0^1, \ldots, g_0^{l_0}) \leftarrow q_1(g_1^1, \ldots, g_1^{l_1}), \ldots, q_k(g_k^1, \ldots, g_k^{l_k})$ in Γ:

(6) If r is a fact, then q is the predicate symbol q_0 in r. Otherwise q is a predicate symbol in the body of r, and l is the arity of q.

(7) Let $q = q_i$ for some $0 \le i \le k$. For $1 \le j \le l (= l_i)$, let $e_j^1, \ldots, e_j^{h_j}$ be all hyperedges in g_i^j. For each $1 \le j \le l$, a_j is a pair $(F_j, (\alpha_j^1, \ldots, \alpha_j^{h_j}))$ defined as follows:

   (a) F_j is the set of m_j distinct edges of g, where m_j is the number of edges of g_i^j.

   (b) For each $1 \le s \le h_j$, α_j^s is a sequence of t_s distinct endpoints of edges in F_j, where t_s is the number of ports of the hyperedge e_j^s in g_i^j.

   Hence, if g_i^j is ground then $a_j = (F_j, ())$.

Since Γ is fixed, *MakeAtom* runs in constant time.

Then we give a path system $Q = (N, T, GEN, S)$ as follows: Let $\sharp$ be the symbol corresponding to the input graph g. Let $S = [p, \sharp]$ and let $N = \{MakeAtom(r) \mid r \in \Gamma\} \cup \{[p, \sharp]\}$.

For an element a_i $(1 \le i \le l)$ of $[q, a_1, \ldots, a_l] \ne [p, \sharp]$ in N, $G[a_i] = (V_{a_i}, E_{a_i}, \varphi_{a_i}, \psi_{a_i})$ denotes a subgraph of g induced by a_i defined as follows: Let $a_i = (F_i, (\alpha_i^1, \ldots, \alpha_i^{t_i}))$. Let $f_1, \ldots, f_m$ be the connected components of the edge-induced subgraph $\ll E - F_i \gg$ of g such that each f_i $(1 \le i \le m)$ has a vertex in $\bigcup_{1 \le j \le t_i} set(\alpha_i^j)$. Then the vertex set V_{a_i} and the edge set E_{a_i} are the sets of all vertices and edges in $f_1, \ldots, f_m$, and $\ll F_i \gg$, respectively. For the symbol $\sharp$ of $[p, \sharp]$ in N, we define $G[\sharp] = g$. Let T be the set of every element $[q, a_1, \ldots, a_l]$ of N such that, for a fact $q(g_1, \ldots, g_l) \leftarrow$ in Γ and each $1 \le i \le l$, $G[a_i]$ is isomorphic to g_i except the edge and vertex labelings for edges in g_i labeled with symbols in Δ and those endpoints, respectively. We can compute the sets N, T and S in constant time, since *MakeAtom* runs in constant time.

We define a function $GEN : N \cup N^2 \cup \cdots \cup N^{k-1} \to 2^N$ as follows where k is the maximum number of atoms of graph rewriting rules in Γ: $GEN([q_1, a_1^1, \ldots, a_1^{l_1}], \ldots, [q_m, a_m^1, \ldots, a_m^{l_m}])$ contains an element $[q, a_0^1, \ldots, a_0^l]$ of N if, for a graph rewriting rule r in Γ, a substitution θ obtained by the procedure $MakeSub([q, a_0^1, \ldots, a_0^l], ([q_1, a_1^1, \ldots, a_1^{l_1}], \ldots, [q_m, a_m^1, \ldots, a_m^{l_m}]), r)$ in Fig. 5 satisfies the following condition: Let $r = q(g_0^1, \ldots, g_0^l) \leftarrow q_1(g_1^1, \ldots, g_1^{l_1}), \ldots, q_m(g_m^1, \ldots, g_m^{l_m})$. For $0 \le i \le m$ and $1 \le j \le l_i$, the instance $g_i^j\theta$ of g_i^j by θ is isomorphic to $G[a_i^j]$, except the edge and vertex labelings for edges in g_i^j labeled with symbols in Δ and those endpoints, respectively. *MakeSub* can be executed in NC^2 by using the NC^2 algorithm [6, 20] computing connected components. It requires to solve the graph isomorphism problem in order to decide

Procedure $MakeAtom(r)$;

/* $r = q_0(g_0^1, \ldots, g_0^{l_0}) \leftarrow q_1(g_1^1, \ldots, g_1^{l_1}), \ldots, q_k(g_k^1, \ldots, g_k^{l_k})$. */

$N := \emptyset$; $K_1 := 1$; $K_2 := k$;
if r is a fact **then** $K_1 := 0$; $K_2 := 0$;
for each $K_1 \leq i \leq K_2$ **pardo**
    for each $1 \leq j \leq l_i$ **pardo**
        let s be the number of edges in g_i^j.
        let F be the family of sets of s distinct edges of g;
        if g_i^j has no hyperedges **then** $MakeTerm(g_i^j) := \{(f, ()) \mid f \in F\}$
        else
            for each $\{h_1, \ldots, h_s\} \in F$ **pardo**
                let $e_1, \ldots, e_m$ be the all hyperedges of g_i^j;
                for each $1 \leq t \leq m$ **pardo**
                let n_t be the number of ports of e_t;
                let $D_{e_t} := \{(v_1, \ldots, v_{n_t}) \mid v_1, \ldots, v_{n_t} \in h_1 \cup \cdots \cup h_s\}$;
                end pardo
                let $MakePorts(\{h_1, \ldots, h_s\}) := \{(d_1, \ldots, d_m) \mid d_1 \in D_{e_1}, \ldots, d_m \in D_{e_m}\}$;
            end pardo
            $MakeTerm(g_i^j) := \{(f, \alpha) \mid f \in F, \alpha \in MakePorts(f)\}$;
    end pardo
    $N := N \cup \{[q_i, a_1, \ldots, a_{l_i}] \mid a_j \in MakeTerm(g_i^j), 1 \leq j \leq l_i\}$;
end pardo

Figure 4: *MakeAtom*

Procedure $MakeSub(L, M, r)$;

/* $L = [q, a_0^1, \ldots, a_0^l]$ */

/* $M = ([q_1, a_1^1, \ldots, a_1^{l_1}], \ldots, [q_m, a_m^1, \ldots, a_m^{l_m}])$ */

/* $a_j^i = (F_{ij}, (\alpha_{ij}^1, \ldots, \alpha_{ij}^{t_{ij}}))$ for $1 \le i \le m, 1 \le j \le l_i$ */

/* $r = q(g_0^1, \ldots, g_0^l) \leftarrow q_1(g_1^1, \ldots, g_1^{l_1}), \ldots, q_k(g_m^1, \ldots, g_m^{l_m})$. */

/* $g_i^j = (V_i^j, E_i^j, \varphi_i^j, \psi_i^j, H_i^j, \lambda_i^j, ports_i^j)$ for $0 \le i \le m, 1 \le j \le l_i$ */

1. $\theta := \emptyset$; $U := \emptyset$;
2. **for** $i := 1$ **to** m **do**
3. **for** $j := 1$ **to** l_i **do**
4. **let** $e_1, \ldots, e_t$ be the all hyperedges in H_i^j;
5. **if** $t = t_{ij}$ **then**
6. **for** $s := 1$ **to** t **do**
7. **if** $\lambda_i^j(e_s) \notin U$ **then**
8. $U := U \cup \{\lambda_i^j(e_s)\}$;
9. **let** $f_1, \ldots, f_{k_s}$ be the connected components of $\ll E - F_{ij} \gg$ such that each f_k $(1 \le k \le k_s)$ has a vertex in $set(\alpha_{ij}^s)$;
10. **let** V_s and E_s be the sets of vertices and edges in $f_1, \ldots,$ and f_{k_s};
11. **let** $g_s = (V_s, E_s, \varphi_s, \psi_s)$ be the subgraph of g_i^j induced by V_s and E_s;
12. $\theta := \theta \cup \{\lambda_i^j(e_s) := [g_s, \alpha_{ij}^s]\}$
13. **end do**
14. **end do**
15. **end do**

Figure 5: *MakeSub*

whether the substitution θ constructed by *MakeSub* satisfies the above condition or not. Thus the problem of computing GEN is NC^2-reducible to the graph isomorphism problem. By Lemma 1, the graphs defined by Γ are of constantly bounded valence. Therefore, we can compute GEN in polynomial time by using the polynomial-time algorithm [11] solving the graph isomorphism problem whose input graphs are restricted to a family of graphs with constantly bounded valence. Thus, we can construct the path system Q in polynomial time.

It can be easily seen that the path system Q is solvable if and only if the ground term graph g is defined by the FGS Γ. Since we can solve Q in polynomial time, we can construct a generation tree GT of $[p, \sharp]$ on the path system Q in polynomial time. Then we can compute a refutation tree for g in Γ from GT by replacing a label $[q, a_1, \ldots a_l]$ of each vertex in GT with the unit goal $\leftarrow q(G[a_1], \ldots, G[a_l])$ and by attaching a new vertex labeled with the empty goal to each leaf. Thus, the refutation tree problem for (Γ, p) is solvable in polynomial time. $\square$

Let $g = (V, E, \varphi, \psi, H, \lambda, ports)$ be a term graph and $\tilde{V}$ be the set of vertices which are not ports of any hyperedges in H. Then we define the *size* of g, denoted by $|g|$, as a pair $(\#\tilde{V}, \#E)$ of the numbers $\#\tilde{V}$ and $\#E$ of vertices in $\tilde{V}$ and edges in E, respectively. Let g_i be a term graph and $(\#\tilde{V}_i, \#E_i)$ be the size of g_i for $1 \le i \le n$. Then, for an atom $p(g_1, \ldots, g_n)$, we define $\|p(g_1, \ldots, g_n)\| = (\#\tilde{V}_1 + \cdots + \#\tilde{V}_n, \#E_1 + \cdots + \#E_n)$. Let $A_1, \ldots, A_n$ be atoms and let $\|A_i\| = (s_i, t_i)$ for $1 \le i \le n$. Then, we denote $\|A_1\| \ge \|A_2\| + \cdots + \|A_n\|$ if $s_1 \ge s_2 + \cdots + s_n$ and $t_1 \ge t_2 + \cdots + t_n$.

Definition 6. A graph rewriting rule $A \leftarrow B_1, \ldots, B_m$ is said to be *size-bounded* if $\|A\theta\| \ge \|B_1\theta\| + \cdots + \|B_m\theta\|$ for any substitution θ. An FGS Γ is *size-bounded* if every graph rewriting rule in Γ is size-bounded.

For example, the simple FGS Γ in Fig. 3 is size-bounded.

Corollary 1. Let Γ be a size-bounded simple FGS and p be a predicate symbol in Γ. When an input graph is connected, $\mathrm{RT}(\Gamma, p)$ is NC^2-reducible to the graph isomorphism problem for graphs of constantly bounded valence.

Proof. From Lemma 1, the graphs defined by Γ are of constantly bounded valence. By using the algorithm in Theorem 1, the problem of constructing a path system $Q = (N, T, GEN, S)$ simulating how Γ generates graphs is NC^2-reducible to the graph isomorphism problem for graphs of constantly bounded valence.

An element $[q, a_1, \ldots, a_l]$ of N is *realizable* if there is a refutation from $\leftarrow q(G[a_1], \ldots, G[a_l])$ on Γ. By the definition of the simple FGS Γ, the set N has a polynomial number of elements. Since Γ is size-bounded, the generation tree of each realizable element in the path system Q is of polynomial size. Therefore, the set of realizable elements is computable in NC^2 [14]. Thus, we can solve the path system Q in NC^2. We can compute a refutation tree for an input graph from a generation tree of Q in the same way as Theorem 1. $\square$

Let Γ be a size-bounded simple FGS and p be a unary predicate symbol in Γ such that each graph rewriting rule $q_0(g_0^1, \ldots, g_0^{l_0}) \leftarrow q_1(g_1^1, \ldots, g_1^{l_1}), \ldots, q_k(g_k^1, \ldots, g_k^{l_k})$ in Γ satisfies the following conditions: Let $g_i^j = (V_i^j, E_i^j, \varphi_i^j, \psi_i^j, H_i^j, ports_i^j)$ for $0 \leq i \leq k$ and $1 \leq j \leq l_i$.

(1) For each $0 \leq i \leq k$ and $1 \leq j \leq l_i$, $\hat{g}_i^j = (V_i^j, E_i^j, \varphi_i^j, \psi_i^j)$ is an undirected colored-tree.

(2) For each $0 \leq i \leq k$ and $1 \leq j \leq l_i$, the number of ports of every hyperedge in H_i^j is only one.

Then it is easy to see that the graphs generated by Γ are undirected colored-trees of constantly bounded valence. The graph isomorphism problem is in NC^2 [1] when an input graph is restricted to a tree of constantly bounded valence. By Corollary 1, the refutation tree problem $\mathrm{RT}(\Gamma, p)$ is in NC^2.

Let m be a positive integer. Let $g = (V, E, \varphi, \psi)$ be a ground term graph and let $W_1, \ldots, W_k$ be subsets of V. We assume that W is a subset of V such that the number of vertices in W is not greater than m. The sequence $(W_1, \ldots, W_k, W)$ is an *m-separation* of g if the removal of W from g disconnects g into connected components f_i $1 \leq i \leq k$, such that the vertex set of f_i is W_i for each $1 \leq i \leq k$. A ground term graph $g = (V, E, \varphi, \psi)$ is said to have an *m-separator* if, for any subset U of V with l vertices, there is an m-separation $(W_1, \ldots, W_k, W)$ of g such that for each $1 \leq i \leq k$, the intersection $W_i \cap U$ has less than $(2/3)l$ vertices. We remark that, if g has an m-separator then any subgraph of g has also an m-separator. Let h and g be connected ground term graphs of constantly bounded valence. We assume that g has an m-separator. Then Lingas [9] has presented the NC^3 algorithm deciding whether h and g are isomorphic. Let Γ be a size-bounded simple FGS and p be a unary predicate symbol in Γ. When an input graph is restricted to the family of connected graphs of constantly bounded valence and having constantly bounded separator, the refutation tree problem $\mathrm{RT}(\Gamma, p)$ is in NC^3.

Let $g = (V, E, \varphi, \psi, H, \lambda, ports)$ be a term graph. For a variable $x \in X$, the number of hyperedges in H labeled with x is denoted by $o_x(g)$. For an atom $p(g_1, \ldots, g_n)$ and a variable $x \in X$, we define $o_x(p(g_1, \ldots, g_n)) = o_x(g_1) + \cdots + o_x(g_n)$. A graph rewriting rule $A \leftarrow B_1, \ldots, B_m$ is said to be *occurrence-bounded* if $o_x(A) \leq 1$ and $o_x(B_1) + \cdots + o_x(B_m) \leq 1$ for each variable $x \in X$. An FGS Γ is *bounded* if every graph rewriting rule in Γ is size-bounded and occurrence-bounded.

Corollary 2. Let Γ be a bounded simple FGS and p be a unary predicate symbol in Γ. Then $\mathrm{RT}(\Gamma, p)$ is in NC^2.

Proof. By the definition of bounded simple FGS Γ, it does not require to solve the graph isomorphism problem for graphs of constantly bounded valence in order to compute a refutation tree of an input graph for (Γ, p). Hence, from Corollary 1, $\mathrm{RT}(\Gamma, p)$ is in NC^2. $\square$

4 Bounded Simple FGS and Context-Free Graph Grammar

The *parsing problem* for a context-free graph grammar (CFGG) is to construct a parse tree of an input graph. As CFGGs, we deal with those defined in [17]. Rytter and Szymacha [15] have shown that the parsing problem for a CFGG is in NC^2. This section shows that the parsing problem for CFGGs is NC^1-reducible to the refutation tree problem for bounded simple FGSs, and investigates that Theorem 1 includes their result.

A *vertex-colored* graph $G = (V, E, \varphi)$ over Σ consists of a vertex set V, an edge set E, and a vertex labeling $\varphi : V \to \Sigma$. A *star graph* is a pair $S = (G, R)$, where G is a vertex-colored graph and R is the set of edges disjoint with the set of edges of G. We require that each edge in R has exactly one common vertex with G. The graph G is called a *kernel* of S. The vertices of G are labeled by terminal or nonterminal symbols. We denote terminal symbols by $a, b, \cdots$ and use capital letters for representing nonterminal symbols. The edge in R is called the *leg* of G. A *simple-star* is a star graph with its kernel being a single vertex without any edges. We assume that each nonterminal symbol A has the *rank*, denoted by $rank(A)$, that is a nonnegative integer.

Definition 7. [17] A *context-free graph grammar* (CFGG) $\mathcal{G} = (\sigma, R)$ is defined as follows:

(1) σ is a simple-star without any legs, called the *axiom*.

(2) R is a finite set of *productions* of the form $Y \to Z$, where Y is a simple-star with r legs such that its kernel is labeled by a nonterminal symbol A with $rank(A) = r$, and Z is a star graph with same r legs as Y.

We draw a CFGG $\mathcal{G}$ in Fig. 6 as an example.

Let $\mathcal{G}$ be a CFGG. For a simple-star Y, a star graph Z, and $i \geq 1$, we define the relation $Y \to^i Z$ inductively as follows:

(1) We denote $Y \to^1 Z$ if there is a production $Y \to Z$ is in $\mathcal{G}$.

(2) For $i \geq 2$, we denote $Y \to^i Z$ if there are $j, l \geq 1$, a star graph Z_1 having a simple-star Y_0 as a subgraph, and a star graph Z_2 such that $j + l = i$, $Y \to^j Z_1$, $Y_0 \to^l Z_2$, and Z is a star graph by replacing the kernel of the subgraph Y_0 of Z_1 by the kernel of Z_2.

We write $Y \to^+ Z$ if $Y \to^i Z$ for some $i \geq 1$. A CFGG $\mathcal{G}$ generates a vertex-colored graph g if $\sigma \to^+ g$.

Theorem 2. The parsing problem for CFGGs is NC^1-reducible to the refutation tree problem for bounded simple FGSs.

Proof. Let $\mathcal{G}$ be a CFGG. Without loss of generality, we assume that each star graph on the right-hand side of a production in $\mathcal{G}$ has at most two vertices labeled by nonterminal symbols, and that p is a nonterminal symbol not appearing in $\mathcal{G}$.

A *linear term graph* over $\langle \Sigma, \Lambda \rangle$ with m edges is a ground term graph $f = (V_f, E_f, \varphi_f, \psi_f)$ defined as follows: The vertex set V_f consists of $m+1$ vertices $v_1, \ldots, v_{m+1}$ labeled with symbols in Σ. The edge set E_f contains an edge labeled with a symbol in Λ which has vertices v_i and v_{i+1} as endpoints for each $1 \leq i \leq m$. The vertices v_1 and v_{m+1} are called the *end-vertices*.

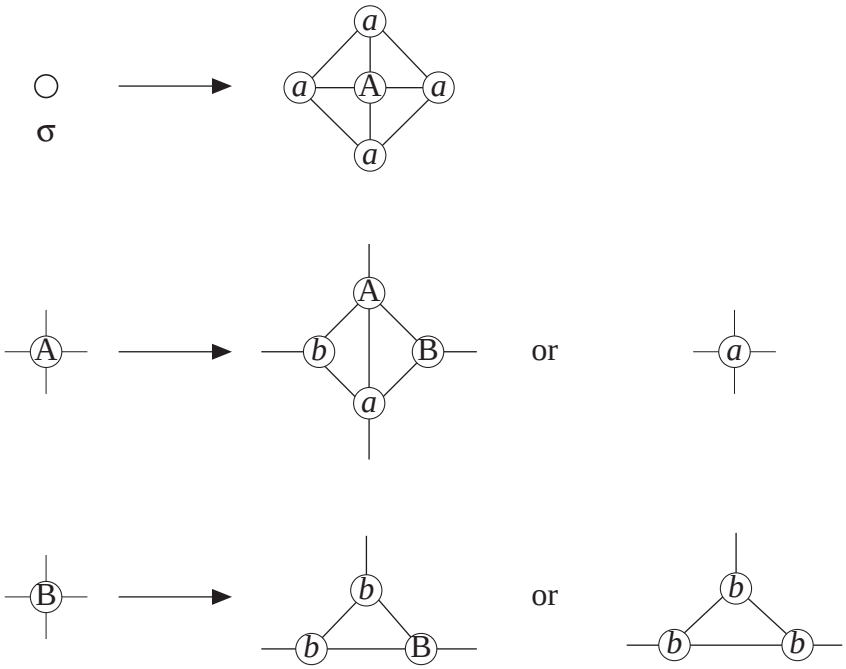

Figure 6: Productions of the CFGG $\mathcal{G}$.

We say that $f' = (V', E', \varphi', \psi', H', \lambda', ports')$ is a *star term 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'.

We define a bounded simple FGS $\Gamma_{\mathcal{G}}$ in the following way: Let N be the set of nonterminal symbols in $\mathcal{G}$ and the nonterminal symbol p, and Σ be the set of terminal symbols in $\mathcal{G}$. Let Σ_1 and Σ_2 be non-empty disjoint sets of symbols such that $(N \cup \Sigma) \cap (\Sigma_1 \cup \Sigma_2) = \emptyset$, and let Λ_1 and Λ_2 be non-empty disjoint sets. We regard each nonterminal symbol 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 in Σ_2. That is, $\Sigma_2 = \bigcup_{A \in N} set(reg(A))$. For a simple-star Y in $\mathcal{G}$, let $labels(Y)$ be a sequence of m_Y symbols in Λ_2 where m_Y is the number of legs of Y. For each production $Y \to Z$ in $\mathcal{G}$, Let A and B_1, B_2 be nonterminal symbols in Y and Z, respectively. If $Y = \sigma$ then let A be the nonterminal symbol p. Then, $\Gamma_{\mathcal{G}}$ contains the graph rewriting rule $A(h) \leftarrow B_1(g_1), B_2(g_2)$, where simple term graphs h, g_1, g_2 with the enclosure set Λ_2 in $\mathcal{T}(\Sigma \cup \Sigma_1 \cup \Sigma_2, \Lambda_1 \cup \Lambda_2, X)$ are defined as follows (see Example 4):

(1) We assume that the star graph $Z = (G, R)$ has two simple-stars $Y_1 = (G_1, R_1)$ and $Y_2 = (G_2, R_2)$ as its subgraphs. Let $h = (V_h, E_h, \varphi_h, \psi_h, H_h, \lambda_h, ports_h)$ be a term graph obtained from Z in the following way: Let f be a ground term graph over $\langle N \cup \Sigma, \Lambda_1 \rangle$ defined as the vertex-colored graph G having an edge labeling from the edge set of G to Λ_1. For each edge e in f with endpoints u_e, and v_e, we attach a new linear term graph f_e over $\langle \Sigma_1, \Lambda_1 \rangle$ with 3 edges to f by removing e from f and by identifying u_e and v_e with the end-vertices u_f and v_f of f_e, respectively. Then, the labels of u_e and v_e are $\varphi(u_e)$ and $\varphi(v_e)$, respectively. If u_e (resp., v_e) is the kernel of the simple-star Y_i $(i = 1, 2)$ then we remove e from R_i and add an edge having u_f (resp., v_f) of f_e to R_i as a leg of Y_i. Let f_1 be the resulting graph over $\langle N \cup \Sigma \cup \Sigma_1, \Lambda_1 \rangle$.

For each leg r of Z, let $com(r)$ be the vertex that r and the vertex-colored graph G share in common. Let $r_1, \ldots, r_l$ be the legs of Z. For each $1 \leq i \leq l$, we join f_1 and a new linear term graph h^i with 2 edges by identifying $com(r_i)$ of f_1 with an end-vertex u of h^i, where h^i satisfies the following conditions:

(a) Let v be another end-vertex of h^i. Then the label of v is the ith symbol in $reg(A)$, and the other vertices are labeled with symbols in Σ_1.

(b) The label of the edge having v is the ith symbol of $labels(Y)$ in Λ_2, and the other edge is labeled with a symbol in Λ_1.

Here, the label of $com(r_i)$ in the resulting term graph is that of $com(r_i)$ in f_1. If $com(r_i)$ is the kernel of Y_j ($j = 1,2$) then we remove r_i from R_j and add the edge having u of h^i to R_j. Let f_2 be the resulting ground term graph over $\langle N \cup \Sigma_1 \cup \Sigma_2, \Lambda_1 \cup \Lambda_2 \rangle$. We remark that for each symbol a in $reg(A)$, the number of vertices in f_2 labeled with a is at most one.

For each $i = 1,2$, we construct a star term graph g_{Y_i} for a variable x_i from f_2 such that the rank of x_i is equal to the number of legs of Y_i. For each leg r of Y_i, let $end(r)$ be an endpoint of r not the kernel of Y_i. The term graph h is obtained from f_2 in the following way: For each $i = 1,2$, let $r_i^1, \ldots, r_i^{l_i}$ be the legs of Y_i. For each $i = 1,2$, we attach a star term graph g_{Y_i} to f_2 by removing the kernel of Y_i and its all adjoining edges from f_2 and by identifying the jth port of the unique hyperedge of g_{Y_i} with $end(r_i^j)$ for each $1 \leq j \leq l_i$.

(2) Let e_1 and e_2 be the corresponding hyperedges in h to the simple-star Y_1 and Y_2 in Z. For each $i = 1,2$, the term graph g_i is obtained by applying the same operation as (1) to Y_i such that the label of its unique hyperedge is of e_i in h.

It is easy to see that $\Gamma_{\mathcal{G}}$ consisting of these graph rewriting rules is simple and bounded. Since $\mathcal{G}$ is fixed, we can compute $\Gamma_{\mathcal{G}}$ in constant time.

Let $F = (V, E, \varphi)$ be a vertex-colored graph over Σ for the parsing problem on $\mathcal{G}$, and let $g = (V, E, \varphi, \psi)$ be a ground term graph over $\langle \Sigma, \Lambda_1 \rangle$ obtained from F by adding an edge labeling $\varphi : E \to \Lambda_1$. We construct a ground term graph g' from g in the following way: For each edge e of g which has endpoints u and v, we attach a new linear term graph with 3 edges to g by removing e from E and by identifying its two end-vertices w_1 and w_2 with u and v, respectively. This transformation from g to g' is computable in constant time. Then, the parsing problem for an input vertex-colored graph F on a CFGG $\mathcal{G}$ is NC^1-reducible to the refutation tree problem for the input colored-graph g' on a corresponding bounded simple FGS $\Gamma_{\mathcal{G}}$. $\square$

Example 4. Let $\mathcal{G}$ be the CFGG in Fig. 6. Then we draw the bounded simple FGS $\Gamma_{\mathcal{G}}$ in Fig. 7 such that the parsing problem for $\mathcal{G}$ is NC^1-reducible to the refutation tree problem for $(\Gamma_{\mathcal{G}}, p)$.

Rytter and Szymacha [15] have shown that the following problems are in NC^1 if the parse tree is provided: edge coloring, vertex coloring Hamiltonian cycle, traveling salesman problem, maximal simple path, maximal independent set, minimal dominating set and minimal vertex cover. All these problems are NP-complete. Hence Theorem 2 and Corollary 2 assert that efficient parallel algorithms may exist for a large number of NP-complete problems when these problems are restricted to the families of graphs definable by bounded simple FGSs.

5 Reduction of Graph Isomorphism Problem to Refutation Tree Problem

For two graphs g_1 and g_2, the graph isomorphism problem is to decide whether g_1 and g_2 are isomorphic or not. We present a size-bounded FGS Γ_{GI} in Fig. 8 such that the graph isomorphism problem is reduced to the refutation tree problem for Γ_{GI}.

Let Γ be an FGS and p be its unary predicate symbol. When a ground term graph g is given as an input, the decision version of the refutation tree problem for (Γ, p), denoted by $\mathrm{DRT}(\Gamma, p)$, is to decide whether there is a refutation from the goal $\leftarrow p(g)$ in Γ.

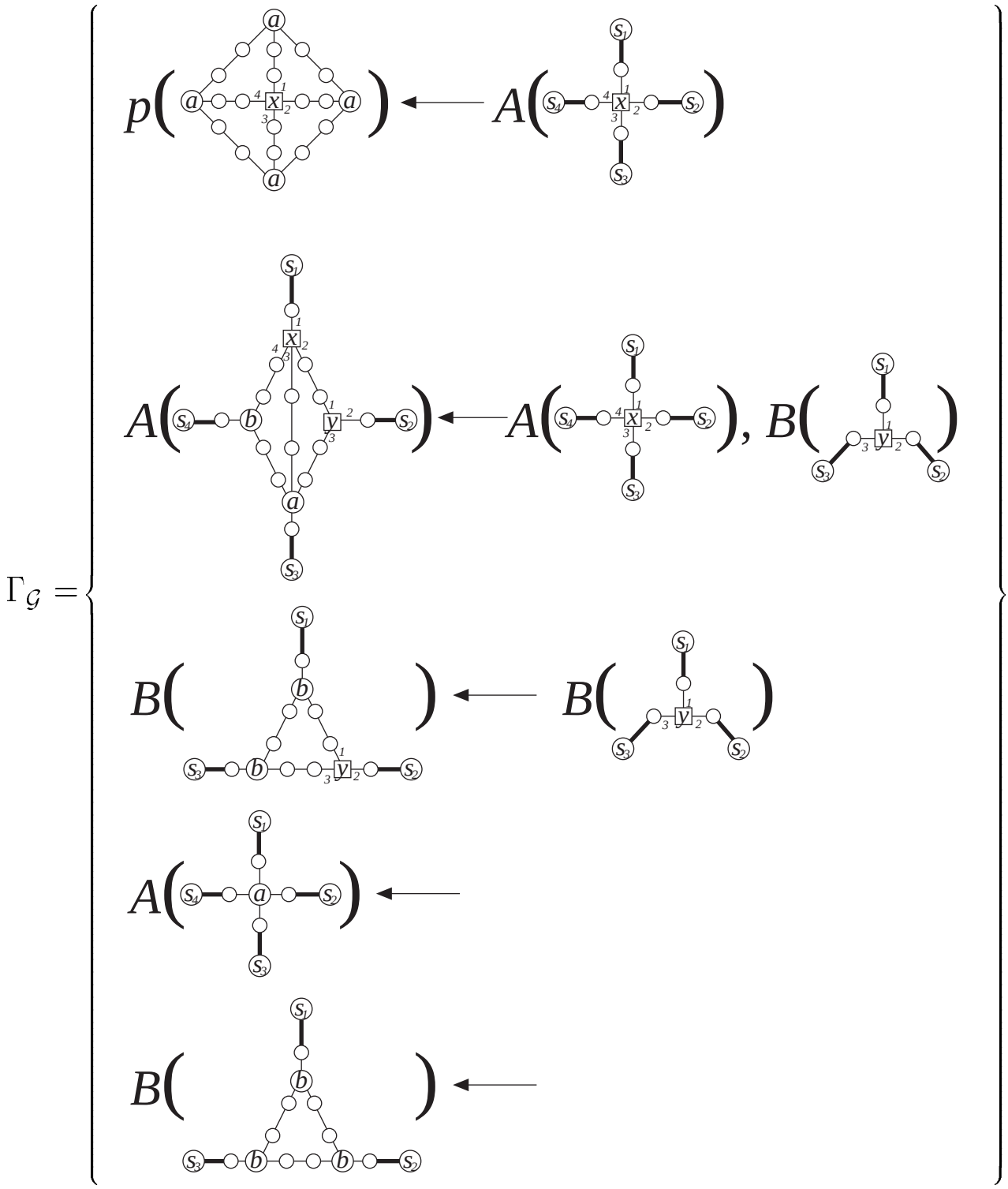

Figure 7: Bounded simple FGS $\Gamma_{\mathcal{G}}$. The labels of thick lines are in the enclosure set.

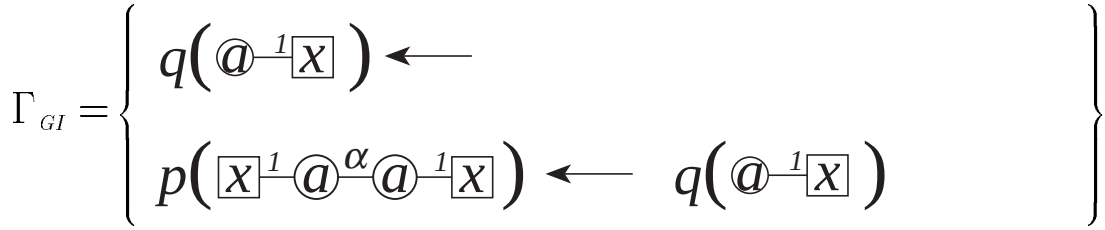

Figure 8: FGS Γ_{GI}

Theorem 3. Let Γ_{GI} be a size-bounded FGS in Fig. 8. Then the graph isomorphism problem is log-space reducible to $\mathrm{DRT}(\Gamma_{GI}, p)$.

Proof. Let $g_1 = (V_1, E_1, \varphi_1, \psi_1)$ and $g_2 = (V_2, E_2, \varphi_2, \psi_2)$ be ground term graphs over $\langle \Sigma, \Lambda \rangle$ for the graph isomorphism problem. Let α is a new symbol not in Λ. Then, we construct a ground term graph $g = (V, E, \varphi, \psi)$ over $\langle \Sigma, \Lambda \cup \{\alpha\} \rangle$ as follows:

(1) $V = V_1 \cup V_2$.

(2) Let c be a new edge having a vertex in V_1 and a vertex in V_2 as endpoints. Then $E = E_1 \cup E_2 \cup \{c\}$

(3) For each vertex v in V, if $v \in V_1$ then $\varphi(v) = \varphi_1(v)$. Otherwise $\varphi(v) = \varphi_2(v)$.

(4) For each edge e in $E_1 \cup E_2$, if $e \in E_1$ then $\psi(e) = \psi_1(e)$. Otherwise $\psi(e) = \psi_2(e)$. For the new edge c, $\psi(c) = \alpha$.

The ground term g is computable from g_1 and g_2 in log-space. We can decide whether g_1 and g_2 are isomorphic by solving $\mathrm{DRT}(\Gamma_{GI}, p)$ for g. $\square$

6 Conclusion

We presented simple FGSs as a subclass of FGSs for which refutation trees can be computed in polynomial time. It is not known whether the refutation tree problem for simple FGSs is P-complete or not. We defined size-bounded simple FGSs and bounded simple FGSs as subclasses of simple FGSs. We showed that the refutation tree problem for size-bounded simple FGSs is NC^2-reducible to the graph isomorphism problem for graphs of constantly bounded valence. For size-bounded simple FGSs generating undirected trees, the refutation tree problem is in NC^2. Moreover, when an input graph is restricted to the family of connected graphs of constantly bounded valence and having a constantly bounded separator, we showed that the refutation tree problem for size-bounded simple FGSs is in NC^3. For bounded simple FGSs, we showed that we can compute a refutation tree in NC^2, since it is not necessary to solve the graph isomorphism problem.

We gave a size-bounded FGS Γ_{GI} and showed that the graph isomorphism problem is log-space reducible to the refutation tree problem for Γ_{GI}. For size-bounded FGSs, the complexity of the refutation tree problem centers around the complexity of the graph isomorphism problem. We are interested in the relation between other graph problems and the refutation tree problem.

By proving that the parsing problem for CFGGs is NC^1-reducible to the refutation tree problem for bounded simple FGSs, we showed that our result for bounded simple FGSs includes the result given by Rytter and Szymacha [15]. However, this result does not show that the family of graphs generated by CFGGs can be also defined by bounded simple FGSs. It is necessary to clarify the relation between FGSs and CFGGs.

References

[1] T. Akutsu. An RNC algorithm for finding a largest common subtree of two trees. *IEICE Transactions on Information and Systems*, E75-D:95–101, 1992.

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

[3] S. Arikawa, T. Shinohara, and A. Yamamoto. Learning elementary formal systems. *Theoretical Computer Science*, 95:97–113, 1992.

[4] S. A. Cook. An observation on time-storage trade off. *Journal of Computer and System Sciences*, 9:308–316, 1974.

[5] S. A. Cook. A taxonomy of problems with fast parallel algorithms. *Information and Control*, 64:2–22, 1985.

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

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

[8] C. Lautemann. The complexity of graph languages generated by hyperedge replacement. *Acta Informatica*, 27:399–421, 1990.

[9] A. Lingas. Subgaph isomorphism for connected graphs of bounded valence and bounded separator is in NC. In *Proceedings of the International Conference on Parallel Processing*, pages 304–307, 1988.

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

[11] E. M. Luks. Isomorphism of graphs of bounded valence can be tested in polynomial time. *Journal of Computer and System Sciences*, 25:42–65, 1982.

[12] Y. Okabe and S. Yajima. Parallel complexity of logic programs and highly parallel computation by logic circuits. *Transactions of Information Processing Society of Japan*, 31:840–848, 1990.

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

[14] W. Rytter. The complexity of two-way pushdown automata and recursive programs. In *Combinatorial Algorithms on Words, NATO ACI Series*, volume F12, pages 341–356. Springer-Verlag, 1985.

[15] W. Rytter and T. Szymacha. Parallel algorithms for a class of graphs generated recursively. *Information Processing Letters*, 30:225–231, 1989.

[16] E. Y. Shapiro. Alternation and computational complexity of logic programs. *Journal of Logic Programming*, 1:19–33, 1984.

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

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

[19] L. Sterling and E. Shapiro. *The Art of Prolog: Advanced Programming Techniques.* The MIT Press, 1986.

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

[21] T. Uchida, T. Shoudai, and S. Miyano. Parallel algorithms for refutation tree problem on formal graph systems. RIFIS-TR-CS-59, Research Institute of Fundamental Information Science, Kyushu University, July 1992.