Distributional Learning of Regular Formal Graph System of Bounded Degree

Takayoshi Shoudai[1(✉)], Satoshi Matsumoto[2], and Yusuke Suzuki[3]

[1] Faculty of International Studies, Kyushu International University, Kitakyushu, Japan
shoudai@isb.kiu.ac.jp
[2] Faculty of Science, Tokai University, Hiratsuka, Japan
[3] Graduate School of Information Sciences, Hiroshima City University, Hiroshima, Japan

Abstract. In this paper, we describe how distributional learning techniques can be applied to formal graph system (FGS) languages. An FGS is a logic program that deals with term graphs instead of the terms of first-order predicate logic. We show that the regular FGS languages of bounded degree with the 1-finite context property (1-FCP) and bounded treewidth property can be learned from positive data and membership queries.

1 Introduction

In the field of algorithmic learning theory, many models and algorithmic techniques for learning from examples have been developed. Distributional learning was first proposed by Clark and Eyraud [3] to learn a subclass of context-free grammars efficiently. Recently, distributional learning techniques have been developed for learning of various subclasses of context-free grammars [11]. These techniques were extended to languages that have more complex structures [7]. Yoshinaka [12] introduced distributional properties on grammars and showed that grammars with distributional properties are learnable with standard distributional learning techniques if the grammars satisfy certain conditions, e.g. polynomial time decomposability of objects into contexts and substructures.

Graph grammar has been developed as an extension to graphs from strings of grammatical forms. Graph grammar has been applied to a wide range of fields including pattern recognition and image analysis. Uchida et al. [10] introduced a framework called formal graph system (FGS) as a graph grammar. An FGS is a logic program that deals with term graphs, which can be considered to be types of hypergraphs, instead of the terms of first-order predicate logic.

For the learning of graph grammar, Okada et al. [9] showed that some classes of graph pattern languages are learned from a minimally adequate teacher (MAT) in polynomial time. Hara and Shoudai [6] proposed an algorithm for

T. Shoudai—This work was partially supported by JSPS KAKENHI (26280087, 15K00313) and MEXT KAKENHI (24106010).

J. Cussens and A. Russo (Eds.): ILP 2016, LNAI 10326, pp. 68–80, 2017.
DOI: 10.1007/978-3-319-63342-8_6

learning the class of c-deterministic regular FGS languages in the framework of MAT learning. There have been other studies on graph grammars from the viewpoint of application, but discussions on computational learning of graph grammars are not yet sufficient. In this paper, we show that the regular FGS languages of bounded degree with the 1-finite context property (1-FCP) [2] and bounded treewidth property can be learned from positive data and membership queries with current distributional learning techniques [11].

2 Preliminaries

For a set or a list S, $|S|$ denotes the number of all elements that are contained in S. For a set S, S^* denotes the set of all finite lists consisting of elements in S. For a list S and an integer i $(1 \leq i \leq |S|)$, $S[i]$ denotes the i-th member of S. Let Σ and Λ be finite alphabets. Let X be an infinite alphabet, whose elements are called *variables*. We assume that each symbol $x \in X$ has a nonnegative integer $rank(x)$, $\Sigma \cap X = \emptyset$ and $\Lambda \cap X = \emptyset$.

Definition 1 (Term graph). *A term graph $g = (V, E, \varphi, \psi, H, \lambda, ports)$ is defined as follows:*

1. *(V, E) is a vertex- and edge-labeled (directed or undirected) graph,*
2. *$\varphi : V \to \Sigma$ and $\psi : E \to \Lambda$ are vertex- and edge-labeling functions,*
3. *H is a finite multiset of hyperedges that are elements of 2^V,*
4. *$\lambda : H \to X$ is a variable-labeling function, and*
5. *$ports : H \to V^*$ is a mapping s.t. for every $h \in H$, $ports(h)$ is a list of $rank(\lambda(h))$ distinct vertices in V. These vertices are called the ports of h.*

We give an example of term graphs in Fig. 1. A hyperedge is drawn as a box with lines to its ports. The order of the ports is indicated by digits at these lines.

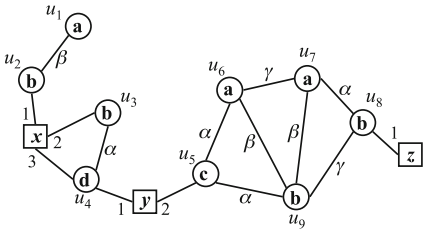

Fig. 1. A term graph $g = (V, E, \varphi, \psi, H, \lambda, ports)$ on $\Sigma = \{\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}\}$ and $\Lambda = \{\alpha, \beta, \gamma\}$: $H = \{\{u_2, u_3, u_4\}, \{u_4, u_5\}, \{u_8\}\}, \lambda(\{u_2, u_3, u_4\}) = x, \lambda(\{u_4, u_5\}) = y, \lambda(\{u_8\}) = z, ports(\{u_2, u_3, u_4\}) = (u_2, u_3, u_4), ports(\{u_4, u_5\}) = (u_4, u_5), ports(\{u_8\}) = (u_8)$.

For a term graph g, its 7-tuple is denoted by $(V_g, E_g, \varphi_g, \psi_g, H_g, \lambda_g, ports_g)$. A term graph g is called *ground* if $H_g = \emptyset$ and both λ_g and $ports_g$ are empty functions $\emptyset$. We define the size of a term graph g, denoted by $|g|$, as $|V_g| + |E_g| + |H_g|$. A term graph g is a *star term graph* if $E_g = \emptyset$ and $H_g = \{V_g\}$. For a star term graph g, h_g denotes the unique hyperedge of g.

Definition 2 (Treewidth [5]). *A tree decomposition of a term graph* $g = (V, E, \varphi, \psi, H, \lambda, ports)$ *is a rooted tree* $\mathcal{T} = (\mathcal{I}, \mathcal{F})$ *whose vertices* $i \in \mathcal{I}$ *are associated with* $V_i \subseteq V$, $E_i \subseteq E$ *and* $H_i \subseteq H$ *that satisfy the following conditions:*

1. *For each* $v \in V$, *there is a vertex* $i \in \mathcal{I}$ *such that* $v \in V_i$.
2. *For each* $e = (u, v) \in E$, *there is exactly one vertex* $i \in \mathcal{I}$ *such that* $u, v \in V_i$ *and* $e \in E_i$.
3. *For each* $h = \{v_1, \ldots, v_m\} \in H$, *there is exactly one vertex* $i \in \mathcal{I}$ *such that* $v_1, \ldots, v_m \in V_i$ *and* $h \in H_i$.
4. *For each* $v \in V$, *the subtree of* $\mathcal{T}$ *induced by* $\{i \in \mathcal{I} \mid v \in V_i\}$ *is connected.*

The *width* of $\mathcal{T}$ is defined as $\max_{i \in \mathcal{I}} |V_i| - 1$. The *treewidth* of g is the minimum width of any tree decomposition $\mathcal{T} = (\mathcal{I}, \mathcal{F})$ of g.

Two term graphs f and g are said to be *isomorphic*, if there is a bijection π from V_f to V_g, such that (1) $(u, v) \in E_f$ if and only if $(\pi(u), \pi(v)) \in E_g$, (2) $\varphi_f(u) = \varphi_g(\pi(u))$ for each vertex $u \in V_f$ and $\psi_f(u, v) = \psi_g(\pi(u), \pi(v))$ for each edge $(u, v) \in E_f$, (3) $\{v_1, \ldots, v_\ell\} \in H_f$ if and only if $\{\pi(v_1), \ldots, \pi(v_\ell)\} \in H_g$, (4) $\lambda_f(\{v_1, \ldots, v_\ell\}) = \lambda_f(\{u_1, \ldots, u_\ell\})$ if and only if $\lambda_g(\{\pi(v_1), \ldots, \pi(v_\ell)\}) = \lambda_g(\{\pi(u_1), \ldots, \pi(u_\ell)\})$ for each hyperedge $\{v_1, \ldots, v_\ell\}, \{u_1, \ldots, u_\ell\} \in H_f$, and (5) $ports_f(\{v_1, \ldots, v_\ell\}) = ports_g(\{\pi(v_1), \ldots, \pi(v_\ell)\})$ for each $\{v_1, \ldots, v_\ell\} \in H_f$. A bijection π satisfying (1)–(5) is called an *isomorphism* from f to g.

Let d and w be nonnegative integers. The degree of a vertex v is defined as $|\{e \in E_g \mid v \in e\}| + |\{h \in H_g \mid v \in h\}|$. $\mathcal{G}(\Sigma, \Lambda, X)$ (resp. $\mathcal{G}_{d,w}(\Sigma, \Lambda, X)$) denotes the set of all term graphs (resp. all term graphs of maximum degree d and treewidth w) over $\langle \Sigma, \Lambda, X \rangle$. Moreover, $\mathcal{G}(\Sigma, \Lambda)$ (resp. $\mathcal{G}_{d,w}(\Sigma, \Lambda)$) denotes the set of all ground term graphs (resp. all ground term graphs of maximum degree d and treewidth w).

Let f be a term graph in $\mathcal{G}(\Sigma, \Lambda, X)$ and σ an ordered list of ℓ distinct vertices in V_f $(0 \leq \ell \leq |V_f|)$. A pair $[f, \sigma]$ is called a *term graph fragment*. If f is a ground term graph, we call it a *ground term graph fragment*. Let $\mathcal{F}(\Sigma, \Lambda)$ be the set of all ground term graph fragments. For nonnegative integers d and w, $\mathcal{F}_{d,w}(\Sigma, \Lambda) = \{[f, \sigma] \in \mathcal{F}(\Sigma, \Lambda) \mid f \in \mathcal{G}_{d,w}(\Sigma, \Lambda)$ and $|\sigma| \leq d + 1\}$. For a term graph fragment $[f, \sigma]$ and a variable $x \in X$ with $rank(x) = |\sigma|$. Let $\sigma = (v_1, \ldots, v_\ell)$ $(\ell \geq 1)$. The *binding* $x := [f, \sigma]$ for a term graph g is defined to be an operation on g that works in the following way: For each $h \in H_g$ with $\lambda_g(h) = x$, let $f' = (V_{f'}, E_{f'}, \varphi_{f'}, \psi_{f'}, H_{f'}, \lambda_{f'}, ports_{f'})$ be a copy of f. For a vertex $v \in V_f$, we denote the corresponding copy vertex of f' by v'. We attach f' to g by removing the hyperedge h from H_g and by identifying the ports $u_1, \ldots, u_\ell$ of h in g with $v'_1, \ldots, v'_\ell$ in f', respectively. We set the new vertex-label of u_i to be the original vertex-label of u_i, i.e., $\varphi_g(u_i)$. A *substitution* θ is a finite set of bindings $\{x_1 := [f_1, \sigma_1], \ldots, x_n := [f_n, \sigma_n]\}$, where x_i's are mutually distinct variables in X and each f_i has no hyperedge labeled with a variable in $\{x_1, \ldots, x_n\}$. We give an example of term graphs and substitutions in Fig. 2.

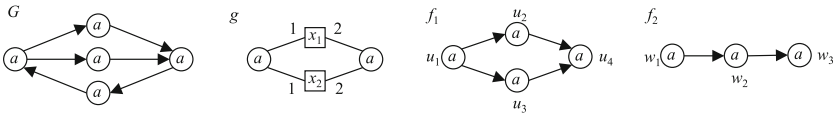

Fig. 2. A graph G can be obtained from g by applying a substitution $\theta = \{x_1 := [f_1, (u_1, u_4)], x_2 := [f_2, (w_3, w_1)]\}$, i.e., $g\theta$ is isomorphic to G.

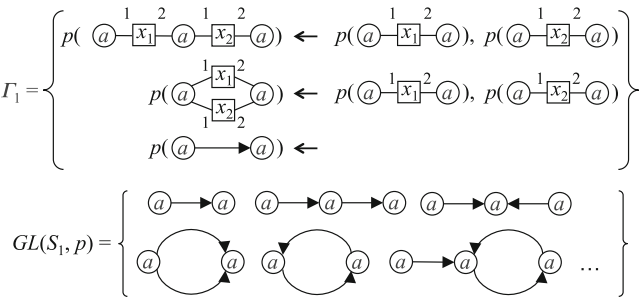

Fig. 3. A formal graph system $S_1 = (\Sigma_1, \Lambda_1, X_1, \Pi_1, \Gamma_1)$, where $\Sigma_1 = \{a\}, \Lambda_1 = \{\epsilon\}, X_1 = \{x_1, x_2, \ldots\}, \Pi_1 = \{p\}$, and its FGS language $GL(S_1, p)$.

Definition 3 (Formal graph system [10]). *Let $g_1, \ldots, g_n \in \mathcal{G}(\Sigma, \Lambda, X)$ $(n \geq 1)$. Let Π_n be a finite set of n-ary predicate symbols. Let $\Pi = \bigcup_{i \geq 0} \Pi_i$. For $p \in \Pi_n$, we say that $p(g_1, \ldots, g_n)$ is an atom. Let $A, B_1, \ldots, B_m$ be atoms $(m \geq 0)$ consisting of term graphs in $\mathcal{G}(\Sigma, \Lambda, X)$ and predicate symbols in Π. A graph rewriting rule over $\langle \Sigma, \Lambda, X, \Pi \rangle$ is a clause of the form $A \leftarrow B_1, \ldots, B_m$. For a clause $A \leftarrow B_1, \ldots, B_m$, the atom A is called the head and the right hand side of the arrow $B_1, \ldots, B_m$ is called the body of the rule. Let Γ be a finite set of graph rewriting rules over $\langle \Sigma, \Lambda, X, \Pi \rangle$. A formal graph system (abbreviated to FGS) is the 5-tuple $S = (\Sigma, \Lambda, X, \Pi, \Gamma)$.*

For a substitution θ and an atom $p(g_1, \ldots, g_n)$, we define $p(g_1, \ldots, g_n)\theta$ to be $p(g_1\theta, \ldots, g_n\theta)$. For a graph rewriting rule $A \leftarrow B_1, \ldots, B_m$, we also define $(A \leftarrow B_1, \ldots, B_m)\theta$ to be $A\theta \leftarrow B_1\theta, \ldots, B_m\theta$.

Definition 4. *Let $S = (\Sigma, \Lambda, X, \Pi, \Gamma)$ be an FGS. For a clause C, relation $\Gamma \vdash C$ is defined recursively in the following way:*

1. *If $C \in \Gamma$, then $\Gamma \vdash C$ holds.*
2. *If $\Gamma \vdash C$, then $\Gamma \vdash C\theta$ for an arbitrary substitution θ.*
3. *If $\Gamma \vdash A \leftarrow B_1, \ldots, B_n$ and for some i $(1 \leq i \leq n)$, $\Gamma \vdash B_i \leftarrow C_1, \ldots, C_m$, then $\Gamma \vdash A \leftarrow B_1, \ldots, B_{i-1}, C_1, \ldots, C_m, B_{i+1}, \ldots, B_n$ holds.*

For an FGS $S = (\Sigma, \Lambda, X, \Pi, \Gamma)$ and a unary predicate symbol p, we define the *graph language* of (S, p) as $GL(S, p) = \{g \in \mathcal{G}(\Sigma, \Lambda) \mid \Gamma \vdash p(g) \leftarrow\}$. We say that a graph language $L \subseteq \mathcal{G}(\Sigma, \Lambda)$ is *definable by an FGS* or an *FGS language* if such a pair (Γ, p) exists. In Fig. 3, we give an example of the FGSs and its FGS language.

Let Σ be a set of vertex labels and Π a set of predicate symbols. Let δ be a function from Π to Σ^*. We call the function δ a *pointer* of Π if for any predicate symbol p, $\delta(p)[i] \neq \delta(p)[j]$ for all i, j $(1 \leq i < j \leq |\delta(p)|)$. Let $\delta(\Pi)$ be the set of all vertex labels appearing in $\delta(p)$ for all predicate symbols $p \in \Pi$. For a term graph g and a list of vertices $\sigma = (v_1, \ldots, v_\ell) \in V_g^\ell$ $(\ell \geq 1)$, $\varphi_g(\sigma)$ denotes $(\varphi_g(v_1), \ldots, \varphi_g(v_\ell))$.

Definition 5 (Regular formal graph system [10]). *We say that an FGS $S = (\Sigma, \Lambda, X, \Pi, \Gamma)$ is regular with a pointer δ of Π if all graph rewriting rules in Γ are of the form $q_0(g_0) \leftarrow q_1(g_1), \ldots, q_m(g_m)$ that satisfies the following conditions:*

1. *All $q_i \in \Pi$ $(0 \leq i \leq m)$ are unary predicate symbols.*
2. *Each g_i $(1 \leq i \leq m)$ is a star term graph s.t. $\varphi_{g_i}(ports_{g_i}(h_{g_i})) = \delta(q_i)$.*
3. *There is a list $(v_1, \ldots, v_{|\delta(q_0)|}) \in V_{g_0}^{|\delta(q_0)|}$ s.t. $\varphi_{g_0}(v_1, \ldots, v_{|\delta(q_0)|}) = \delta(q_0)$ and for any $u \in V_{g_0} \backslash \{v_1, \ldots, v_{|\delta(q_0)|}\}$, $\varphi_{g_0}(u) \in \Sigma \backslash \delta(\Pi)$.*
4. *$\bigcup_{h \in H_{g_0}} \{\lambda_{g_0}(h)\} = \bigcup_{i=1}^{m} \{\lambda_{g_i}(h_{g_i})\}$ and $\lambda_{g_i}(h_{g_i}) \neq \lambda_{g_j}(h_{g_j})$ for $1 \leq i < j \leq m$.*
5. *For every $h_1, h_2 \in H_{g_0}$, $h_1 \neq h_2$ if and only if $\lambda_{g_0}(h_1) \neq \lambda_{g_0}(h_2)$.*

A regular FGS $S = (\Sigma, \Lambda, X, \Pi, \Gamma)$ with a pointer δ is denoted by (S, δ) or $((\Sigma, \Lambda, X, \Pi, \Gamma), \delta)$. Below we call a regular FGS with a pointer a regular FGS.

Let (S, δ) be a regular FGS and p a unary predicate symbol in Π. We define the *graph language* of (S, δ, p) as $GL(S, \delta, p) = \{g \in \mathcal{G}(\Sigma, \Lambda) \mid \Gamma \vdash p(g) \leftarrow\}$. We say that a graph language $L \subseteq \mathcal{G}(\Sigma, \Lambda)$ is *definable by a regular FGS* or a *regular FGS language* if a triplet (S, δ, p) exists such that $L = GL(S, \delta, p)$. In Fig. 4, we give an example of the regular FGSs and its regular FGS language.

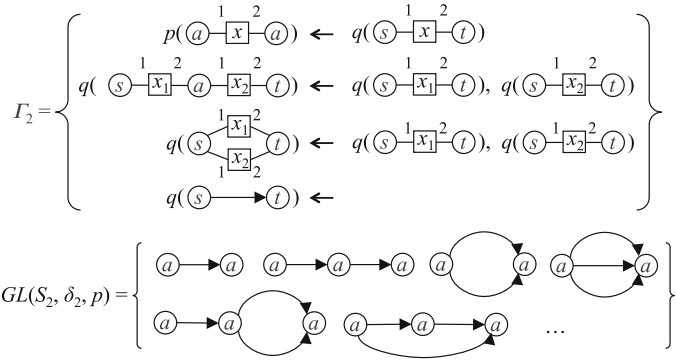

Fig. 4. A regular formal graph system $(S_2, \delta_2) = ((\Sigma_2, \Lambda_2, X_2, \Pi_2, \Gamma_2), \delta_2)$, where $\Sigma_2 = \{a, s, t\}$, $\Lambda_2 = \{\epsilon\}$, $X_2 = \{x_1, x_2, \ldots\}$, $\Pi_2 = \{p, q\}$, $\delta_2(p) = ()$, $\delta_2(q) = (s, t)$, and its FGS language $GL(S_2, \delta_2, p)$, which is equivalent to the set of all two terminal series parallel graphs (TTSP graphs). Every TTSP graph has treewidth at most 2.

Definition 6 (Chomsky normal form). *Let f_0 be a ground term graph of one vertex or two vertices with one edge, f_1 a term graph with two hyperedges and no edge, and f_2, f_3 star term graphs. Let p_0, p_1, p_2, p_3 be unary predicate symbols. A regular FGS (S, δ) is in Chomsky normal form if every graph rewriting rule of S is of the form.*

- *Terminal rule:* $p_0(f_0) \leftarrow$,
- *Unary rule:* $p_1(f_1) \leftarrow p_2(f_2)$.
- *Binary rule:* $p_1(f_1) \leftarrow p_2(f_2),\ p_3(f_3)$.

The regular FGS in Figs. 3 and 4 is written in Chomsky normal form.

We say that a term graph g is connected if for any two vertices u and v of g, there is a sequence of vertices $v_0(=u), v_1, \ldots, v_m(=v)$ for an integer m such that for all i $(0 \leq i \leq m-1)$, v_i and v_{i+1} are contained in the same edge or hyperedge. In this paper, we assume that all term graphs are connected.

Graph grammar has been defined in various ways. One of the famous context-free graph grammars is a hyperedge replacement grammar (HRG) [4]. Uchida et al. [10] showed that a class of graphs is generated by an HRG if and only if it is defined by a regular FGS. This result shows that regular FGSs can generate interesting graph classes including trees, two-terminal series parallel graphs (in Fig. 4), and so on.

In the research of HRGs, Lautemann [8] gave some conditions on either grammar or input graphs whose parsing can be done in polynomial time. A parsing algorithm due to Lautemann is known to be polynomial time for graphs that are connected and of bounded degree. As a more precise characterization of the algorithm's complexity, Chiang et al. [1] showed that the parsing algorithm runs in polynomial time if the maximum degree and treewidth of graphs in an HRG are bounded by some constants. Hence, we conclude the following lemma:

Lemma 1 ([1,10]). *Let (S, δ) be a regular FGS and p a unary predicate symbol. Given a ground term graph g, the problem of deciding whether or not $g \in GL(S, \delta, p)$ is computed in $O((3^d n)^{w+1})$ time, where n is the number of vertices of g, d is the maximum degree of g, and w is the maximum treewidth of the term graphs in the heads of graph rewriting rules in S.*

3 Learning Regular FGS with 1-Finite Context Property

We consider $\mathcal{G}_{d,w}(\Sigma, \Lambda)$ as a universal set $(d, w \geq 0)$. A positive presentation of a nonempty graph language $L \subseteq \mathcal{G}_{d,w}(\Sigma, \Lambda)$ is an infinite sequence $g_1, g_2, \ldots$ of elements in L such that $\{g_1, g_2, \ldots\} = L$.

An inductive inference machine (IIM, for short) is an effective procedure, or a certain type of Turing machine, which outputs a regular FGS and a predicate symbol each time a ground term graph is given. Let $L_* \subseteq \mathcal{G}_{d,w}(\Sigma, \Lambda)$ be a target graph language. We assume that an IIM has an access to an oracle Mem_{L_*} who answers membership queries. The query asks whether or not an arbitrary ground term graph g is included in L_*. Let $\tau = g_1, g_2, \ldots$ be a positive presentation of

L_*. An IIM outputs a regular FGS (S_i, δ_i) and a predicate symbol p_i by using membership queries each time a ground term graph g_i in τ is given. An IIM is said to *converge to a regular FGS* (S, δ) *and a predicate symbol* p *for* τ *with polynomial time update by using membership queries*, if M outputs a regular FGS (S_i, δ_i) and a predicate symbol p_i in polynomial time w.r.t. the sum of the size of the given ground term graphs so far, i.e., $|g_1| + |g_2| + \cdots + |g_i|$, and there exists a positive integer $n \geq 1$ with $(S_m, \delta_m) = (S, \delta)$ and $p_m = p$ for any $m \geq n$. Let $\mathcal{C} \subseteq 2^{\mathcal{G}_{d,w}(\Sigma,\Lambda)}$ be a class and $L_* \in \mathcal{C}$. A class $\mathcal{C}$ is said to be *identifiable in the limit with polynomial time update by using membership queries from positive data*, if there exists an IIM M such that for any $L_* \in \mathcal{C}$ and any presentation τ of L_*, M converges to a regular FGS (S, δ) and a predicate symbol p for τ with $GL(S, \delta, p) = L_*$ with polynomial time update by using membership queries.

Let $g = (V_g, E_g, \varphi_g, \psi_g, \emptyset, \emptyset, \emptyset)$ be a ground term graph and $\sigma = (v_1, \ldots, v_\ell)$ a list of distinct vertices in V_g $(1 \leq \ell \leq |V_g|)$. Let x be a new variable label in X that does not appear so far. For the ground term graph fragment $[g, \sigma]$, we denote by $g(\sigma)$ the term graph $(V_g, E_g, \varphi_g, \psi_g, \{h\}, \lambda_g, ports_g)$ where $h = \{v_1, \ldots, v_\ell\}$, $\lambda_g(h) = x$, and $ports_g(h) = \sigma$. In order to make the argument easier, we assume that g has no isolated vertex. Let $\{E_\alpha, E_\beta\}$ be a partition of E_g. Let V_α be the set of all endpoints of edges in E_α and V_β the set of all endpoints of edges in E_β. Let σ be one of the ordered lists of all vertices in $V_\alpha \cap V_\beta$. We obtain two ground term graph fragments $[\alpha, \sigma]$ and $[\beta, \sigma]$. We easily see that $\alpha(\sigma)\{x := [\beta, \sigma]\}$ and $\beta(\sigma)\{x := [\alpha, \sigma]\}$ are isomorphic to g.

For $[\alpha, \sigma_\alpha], [\beta, \sigma_\beta] \in \mathcal{F}(\Sigma, \Lambda)$, we define an operation $\odot$ as follows:

$$[\alpha, \sigma_\alpha] \odot [\beta, \sigma_\beta] = \begin{cases} \alpha(\sigma_\alpha)\{x := [\beta, \sigma_\beta]\} & \text{if } |\sigma_\alpha| = |\sigma_\beta|, \\ undefined & \text{otherwise.} \end{cases}$$

In Fig. 5, we give an example of α and β by a partition of E_g of a ground term graph g. Note that in general, $[\alpha, \sigma_\alpha] \odot [\beta, \sigma_\beta]$ is not always equivalent to $[\beta, \sigma_\beta] \odot [\alpha, \sigma_\alpha]$, because the vertex labels in the first operand always survive by any binding. If $\varphi_\alpha(\sigma_\alpha) = \varphi_\beta(\sigma_\beta)$, $[\alpha, \sigma_\alpha] \odot [\beta, \sigma_\beta] = [\beta, \sigma_\beta] \odot [\alpha, \sigma_\alpha]$ holds.

Let d and w be constant nonnegative integers. For a nonempty finite set of ground term graphs $D \subseteq \mathcal{G}_{d,w}(\Sigma, \Lambda)$, let

$$Sub(D) = \{[\beta, \sigma_\beta] \in \mathcal{F}_{d,w}(\Sigma, \Lambda) \mid \exists [\alpha, \sigma_\alpha] \in \mathcal{F}_{d,w}(\Sigma, \Lambda)[\, [\alpha, \sigma_\alpha] \odot [\beta, \sigma_\beta] \in D \,]\},$$
$$Con(D) = \{[\alpha, \sigma_\alpha] \in \mathcal{F}_{d,w}(\Sigma, \Lambda) \mid \exists [\beta, \sigma_\beta] \in \mathcal{F}_{d,w}(\Sigma, \Lambda)[\, [\alpha, \sigma_\alpha] \odot [\beta, \sigma_\beta] \in D \,]\}.$$

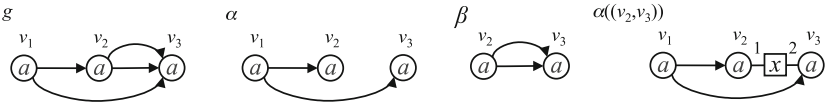

Fig. 5. Two ground term graphs α and β obtained from g by a partition of E_g: It is easy to see that $\alpha((v_2, v_3))\{x := [\beta, (v_2, v_3)]\}$ is isomorphic to g.

We give an example of $Con(D)$ in Fig. 6. It is easy to see that $Con(D) \subseteq Sub(D)$ holds. For any $[\beta, \sigma_\beta] \in Sub(D) \backslash Con(D)$, there is a ground term graph fragment $[\alpha, \sigma_\alpha] \in Con(D)$ such that α is isomorphic to β via an isomorphism ξ with $\xi(\sigma_\alpha) = \sigma_\beta$, ignoring the vertex labels in σ_α. Thus, unlike string grammars, we only use $Con(D)$ to learn a target regular FGS language from positive data. We have the following proposition:

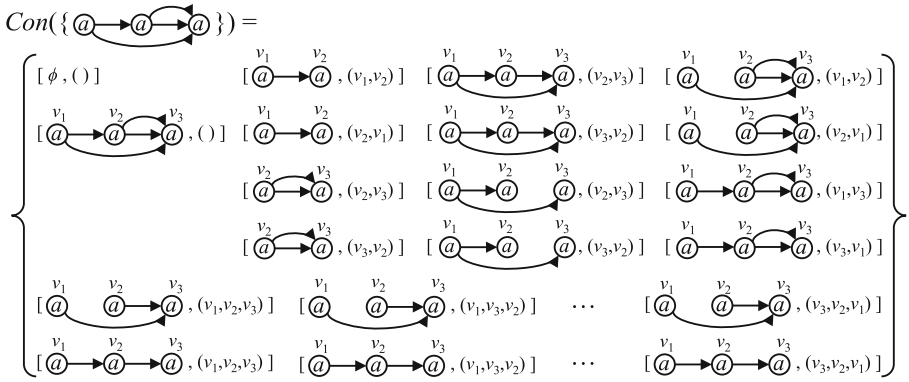

Fig. 6. An example of $Con(D)$.

Proposition 1. *Let D be a nonempty finite subset of $\mathcal{G}_{d,w}(\Sigma, \Lambda)$. Both $|Sub(D)|$ and $|Con(D)|$ are of polynomial size w.r.t. $\sum_{g \in D} |g|$.*

Let $(S, \delta) = ((\Sigma, \Lambda, X, \Pi, \Gamma), \delta)$ be a regular FGS. For a term graph f and $q \in \Pi$, if there are distinct $|\delta(q)|$ vertices $v_1, \ldots, v_{|\delta(q)|}$ in V_f such that $(\varphi_f(v_1), \ldots, \varphi_f(v_{|\delta(q)|})) = \delta(q)$ and for any $v \in V_f \backslash \{v_1, \ldots, v_{|\delta(q)|}\}$, $\varphi_f(v)$ is not a member of $\delta(q)$, we define $\varphi_f^{-1}(\delta(q)) = (v_1, \ldots, v_{|\delta(q)|})$, otherwise we define $\varphi_f^{-1}(\delta(q)) = ()$. Let p, q in Π and $[g, \sigma_g]$ in $\mathcal{F}_{d,w}(\Sigma, \Lambda)$. We define

$$C(S, \delta, p, q, [g, \sigma_g]) = \{f \in \mathcal{G}_{d,w}(\Sigma, \Lambda) \mid [g, \sigma_g] \odot [f, \varphi_f^{-1}(\delta(q))] \in GL(S, \delta, p)\}.$$

Definition 7 (1-FCP). *Let (S, δ) be a regular FGS and p, q unary predicate symbols of S. A term graph fragment $[g, \sigma_g]$ is said to be a context of q w.r.t. (S, δ, p) if $C(S, \delta, p, q, [g, \sigma_g]) = GL(S, \delta, q)$ holds. We say that (S, δ, p) has the 1-finite context property (1-FCP) if every predicate $q \in \Pi$ has a context of it.*

For the ground term graphs α, β in Fig. 5, $[\alpha, (v_2, v_3)]$ is a context of $q_{(2,2)}$ and $[\beta, (v_2, v_3)]$ is a context of $q_{(1,1)}$ w.r.t. $(S_3^{(3)}, \delta_3^{(3)}, p)$ in Fig. 7. We give an example $C(S_3^{(3)}, \delta_3^{(3)}, p, q_{(2,1)}, [g, \sigma_g])$ in Fig. 8 for some $[g, \sigma_g]$. We easily see that for any $d \geq 2$, the regular FGS in Fig. 7 has the 1-finite context property (1-FCP).

Definition 8 (1-FCP regular FGS language class). *1-FCP-$\mathcal{RFGSL}(d, w)$ denotes the set of all regular FGS languages $L \subseteq \mathcal{G}_{d,w}(\Sigma, \Lambda)$ that satisfies the following conditions:*

$$\Sigma_3 = \{a, s, t\}, \Lambda_3 = \{\epsilon\}, X_3 = \{x, x_1, \ldots\}, \Pi_3^{(d)} = \{p\} \cup \{q_{(i,j)} \mid 1 \leq i, j \leq d\},$$
$$\delta_3^{(d)}(p) = (), \delta_3^{(d)}(q_{(i,j)}) = (s, t)\ (1 \leq i, j \leq d).$$

$\Gamma_3^{(d)} =$ {
$p(a\,\overset{1}{-}\,x\,\overset{2}{-}\,a) \leftarrow q_{(d,d)}(s\,\overset{1}{-}\,x\,\overset{2}{-}\,t)$

$q_{(i_1+i_2, j_1+j_2)}(s, x_1, x_2, t) \leftarrow q_{(i_1,j_1)}(s\,\overset{1}{-}\,x_1\,\overset{2}{-}\,t),\ q_{(i_2,j_2)}(s\,\overset{1}{-}\,x_2\,\overset{2}{-}\,t)$
for all i_1, j_1, i_2, j_2 s.t. $1 \leq i_1 + i_2 \leq d$ and $1 \leq j_1 + j_2 \leq d$.

$q_{(i_1,j_2)}(s\,\overset{1}{-}\,x_1\,\overset{2}{-}\,a\,\overset{1}{-}\,x_2\,\overset{2}{-}\,t) \leftarrow q_{(i_1,j_1)}(s\,\overset{1}{-}\,x_1\,\overset{2}{-}\,t),\ q_{(i_2,j_2)}(s\,\overset{1}{-}\,x_2\,\overset{2}{-}\,t)$
for all i_1, j_1, i_2, j_2 s.t. $1 \leq i_1, j_2 \leq d$ and $1 \leq j_1 + i_2 \leq d$.

$q_{(1,1)}(s \longrightarrow t) \leftarrow$
}

Fig. 7. A regular FGS $(S_3^{(d)}, \delta_3^{(d)}) = ((\Sigma_3, \Lambda_3, X_3, \Pi_3^{(d)}, \Gamma_3^{(d)}), \delta_3^{(d)})$ that generates the TTSP graphs of maximum degree d ($d \geq 2$): Predicates $q_{(i,j)}$ generates all TTSP graphs whose vertices labeled with s and t are of degree at most i and j, respectively.

$C(S_3^{(3)}, \delta_3^{(3)}, p, q_{(2,1)}, [\,\overset{\gamma}{\underset{v_1}{a}}\underset{v_2}{a}\underset{v_3}{a}, (v_1, v_3)]) =$ {
$s \to t$; $s \to a \to t$; $s \to a \to a \to t$; $s \to a \to a \to a \to t$;
$s \to a \to t$; $s \to a \to a \to t$; $s \to a \to a \to t$; $s \to a \to a \to a \to t$;
$s \to a \to t$; $s \to a \to a \to t$; $s \to a \to a \to a \to t$; ...
}

Fig. 8. $C(S_3^{(3)}, \delta_3^{(3)}, p, q_{(2,1)}, [\gamma, (v_1, v_3)]) = GL(S_3^{(3)}, \delta_3^{(3)}, q_{(2,1)})$ holds, where $(S_3^{(3)}, \delta_3^{(3)})$ is a regular FGS in Fig. 7. Thus, $[\gamma, (v_1, v_3)]$ is a context of $q_{(2,1)}$ w.r.t. $(S_3^{(3)}, \delta_3^{(3)}, p)$.

1. *L is definable by $(S, \delta, p) = ((\Sigma, \Lambda, X, \Pi, \Gamma), \delta, p)$ that has the 1-FCP,*
2. *Γ is written in Chomsky normal form, and*
3. *The treewidth of each term graph in Γ is at most w. Therefore, the maximum length of ports of the hyperedges in Γ is also at most $w + 1$.*

Let $L_* \subseteq \mathcal{G}_{d,w}(\Sigma, \Lambda)$ be a target regular FGS language. We give a learning algorithm for 1-FCP-$\mathcal{RFGSL}(d, w)$ in Algorithm 1, which is a process of searching in $Con(D)$ for contexts of the predicate symbols in L_*. We construct a regular FGS $S(F, K) = (\Sigma, \Lambda, X, \Pi, \Gamma)$, pointer δ, and initial predicate p as follows:

- $\Sigma = \Sigma' \cup \{s_1, \ldots, s_{w+1}\}$, where $\Sigma' = \{a \mid \exists g_i \in D, \exists v \in V_{g_i}[\ \varphi_{g_i}(v) = a\ ]\}$ and $\Sigma' \cap \{s_1, \ldots, s_{w+1}\} = \emptyset$.
- $\Lambda = \{a \mid \exists g_i \in D, \exists e \in E_{g_i}[\ \psi_{g_i}(e) = a\ ]\}$.
- X: We use a new variable label only when needed.
- $\Pi = \{[\![\alpha, \sigma_\alpha]\!] \mid [\alpha, \sigma_\alpha] \in F \subseteq Con(D)\}$. Let $[\![\emptyset, ()]\!]$ be the initial predicate p.
- δ, Γ: In Table 1, we describe the pointer $\delta(q)$ for each predicate q in Π and the graph rewriting rules in Γ. In the table, we use the following notations.

Algorithm 1. Learn_1-FCP-$\mathcal{RFGSL}$

```
Let K := ∅, F := ∅;
for n = 1, 2, 3, ... do
    Let D = {g_1, g_2, ..., g_n};
    if D ⊈ GL(S(F, K), δ, p) then {By the parsing algorithm in [1].}
        Let F := Con(D);
    end if
    Let K := Con(D);
    output (S(F, K), δ, p);
end for
```

Let k, ℓ be two positive integers ($k \leq \ell$) and $\mathcal{P}_{k,\ell}$ the set of all list of k distinct positive integers that are less than or equal to ℓ. Let $\sigma = (\ell_1, \ldots, \ell_k) \in \mathcal{P}_{k,\ell}$. For a list of elements $\nu = (v_1, \ldots, v_\ell)$ ($k \leq \ell$), $\chi_\sigma(\nu)$ denotes $(v_{\ell_1}, \ldots, v_{\ell_k})$ and $\bar{\chi}_\sigma(\nu)$ denotes the list obtained from ν by deleting $v_{\ell_1}, \ldots, v_{\ell_k}$.

The graph rewriting rule R_1 in Fig. 9 is an example of the graph rewriting rules constructed by Table 1 for the target regular FGS language $GL(S_3^{(3)}, \delta_3^{(3)}, p)$.

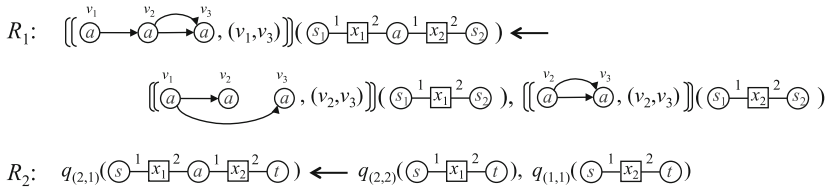

Fig. 9. A graph rewriting rule R_1 constructed by the second table in Table 1: This graph rewriting rule corresponds to the rule R_2 of $(S_3^{(3)}, \delta_3^{(3)})$ in Fig. 7.

Theorem 1. *Let d and w be constant integers greater than zero. The class* 1-*FCP-$\mathcal{RFGSL}(d, w)$ is identifiable in the limit with polynomial time update by using membership queries from positive data.*

Proof. Let $(S_1, \delta_1, p_1), (S_2, \delta_2, p_2), \ldots, (S_i, \delta_i, p_i), \ldots$ be hypotheses output by Algorithm 1, and $(S_i, \delta_i, p_i) = ((\Sigma, \Lambda, X, \Pi_i, \Gamma_i), \delta_i, p_i)$. We prove that there exists a positive integer k such that $GL(S_n, \delta_n, p_n) = L_*$ for any integer $n \geq k$. Let $(S_*, \delta_*) = ((\Sigma, \Lambda, X, \Pi_*, \Gamma_*), \delta_*)$ be a regular FGS and p_* a predicate symbol in Π_* with $L_* = GL(S_*, \delta_*, p_*)$. Let G_i be a ground term graph given to Algorithm 1 at the i-th time, and $D_i = \{G_1, G_2, \ldots, G_i\}$. From the property of positive presentations, there exists a positive integer $n \geq 1$ such that $Con(D_n)$ has a ground term graph fragment $[g, \sigma_g]$ with $C(S_*, \delta_*, p_*, q, g) = GL(S_*, \delta_*, q)$ for any $q \in \Pi_*$. From the n-th input and after, for any predicate symbol $q \in \Pi_*$, Algorithm 1 has a ground term graph fragment corresponding to q. Thus, any graph rewriting rule in Γ_* is included in Γ_m for any $m \geq n$. It follows that $L_* \subseteq GL(S_m, \delta_m, p_m)$ for any $m \geq n$.

Table 1. $(S(F,K),\delta,p)$: There are three types of terminal rules and one type of binary rule. Each graph rewriting rule is created if the corresponding condition is satisfied. All conditions can be determined by asking to the membership oracle Mem_{L_*}. The unary rules can be constructed in a similar way to the binary rules. We omit its detail.

Terminal rules $p_0(f_0) \leftarrow$ in $(S(F,K),\delta,p)$:

p_0	$\delta(p_0)$	f_0	Condition
$[\![g_0,\sigma_{g_0}]\!]$ $\lvert\sigma_g\rvert = 1$	(s_1)	$(\{v_1\},\emptyset,\varphi,\emptyset,\emptyset,\emptyset,())$ $\varphi(v_1) = s_1$	$[g_0,\sigma_{g_0}] \odot [f_0,(v_1)] \in L_*$
$[\![g_0,\sigma_{g_0}]\!]$ $\lvert\sigma_g\rvert = 1$	(s_1)	$(\{v_1,v_2\},\{(v_1,v_2)\},\varphi,\psi,\emptyset,\emptyset,())$ $\varphi(v_1) = s_1, \varphi(v_2) \in \Sigma'$	$[g_0,\sigma_{g_0}] \odot [f_0,(v_1)] \in L_*$
$[\![g_0,\sigma_{g_0}]\!]$ $\lvert\sigma_g\rvert = 2$	(s_1,s_2)	$(\{v_1,v_2\},\{(v_1,v_2)\},\varphi,\psi,\emptyset,\emptyset,())$ $\varphi(v_1) = s_1, \varphi(v_2) = s_2$	$[g_0,\sigma_{g_0}] \odot [f_0,(v_1,v_2)] \in L_*$

Binary rules $p_1(f_1) \leftarrow p_2(f_2),\ p_3(f_3)$ in $(S(F,K),\delta,p)$

$p_i\ (i=2,3)$	$\delta(p_i)\ (i=2,3)$	$f_i\ (i=2,3, j=1,\ldots,\ell_i)$
$[\![g_i,\sigma_{g_i}]\!]$ $\lvert\sigma_{g_i}\rvert = \ell_i$	$(s_1,\ldots,s_{\ell_i})$	$f_i = (\{v_{i,1},\ldots,v_{i,\ell_i}\},\emptyset,\varphi_i,\emptyset,\{h_i\},\lambda_i,port_i)$, where $\varphi_i(v_{i,j}) = s_j, \lambda_2(h_2) \neq \lambda_3(h_3), ports_i(h_i)[j] = v_{i,j}$.
p_1	$\delta(p_1)$	f_1
$[\![g_1,\sigma_{g_1}]\!]$ $\lvert\sigma_{g_1}\rvert = \ell_1$	$(s_1,\ldots,s_{\ell_1})$	$f_1 = [f_2,\chi_{\sigma_2}(ports_2(h_2))] \odot [f_3,\chi_{\sigma_3}(ports_3(h_3))]$, where $\sigma_i \in \mathcal{P}_{k,\lvert ports_i(h_i)\rvert}$ $(i=2,3)$ for some k. Let $\nu = ports_{f_2}(h_2) \cdot \bar{\chi}_{\sigma_3}(ports_{f_3}(h_3))$ and $\sigma_1 \in \mathcal{P}_{\ell_1,\lvert\nu\rvert}$. The vertices in ν are relabeled so that $\chi_{\sigma_1}(\nu) = (s_1,\ldots,s_{\ell_1})$ and $\bar{\chi}_\sigma(\nu) \in \Sigma'^{\lvert\nu\rvert-\ell_1}$.
Condition		
For $\forall[\tau_2,\sigma_{\tau_2}],[\tau_3,\sigma_{\tau_3}] \in K$, if $[g_2,\sigma_{g_2}] \odot [\tau_2,\sigma_{\tau_2}] \in L_*$ and $[g_3,\sigma_{g_3}] \odot [\tau_3,\sigma_{\tau_3}] \in L_*$, then $[g_1,\sigma_{g_1}] \odot [[\tau_2,\chi_{\sigma_2}(\sigma_{\tau_2})] \odot [\tau_3,\chi_{\sigma_3}(\sigma_{\tau_3})],\xi] \in L_*$, where $\xi = \chi_{\sigma_1}(\chi_{\sigma_2}(\sigma_{\tau_2}) \cdot \bar{\chi}_{\sigma_3}(\sigma_{\tau_3}))$.		

We assume that for any $n \geq 1$, there exists a positive integer $m \geq n$ such that $GL(S_m,\delta_m,p_m) \not\subseteq L_*$. Then there exists a ground term graph $G' \in GL(S_m,\delta_m,p_m) \backslash L_*$. Since $G' \in GL(S_m,\delta_m,p_m)$, there exist a graph rewriting rule $p_1(f_1) \leftarrow p_2(f_2),p_3(f_3)$ in Γ_m and ground term graph fragments $[\rho_2,\sigma_{\rho_2}]$ and $[\rho_3,\sigma_{\rho_3}]$ such that $[g_2,\sigma_{g_2}] \odot [\rho_2,\sigma_{\rho_2}] \in L_*$, $[g_3,\sigma_{g_3}] \odot [\rho_3,\sigma_{\rho_3}] \in L_*$ and G' is isomorphic to $[g_1,\sigma_{g_1}] \odot [[\rho_2,\chi_{\sigma_2}(\sigma_{\rho_2})] \odot [\rho_3,\chi_{\sigma_3}(\sigma_{\rho_3})],\xi]$, where p_1, p_2 and p_3 correspond to $[g_1,\sigma_{g_1}]$, $[g_2,\sigma_{g_2}]$ and $[g_3,\sigma_{g_3}]$, respectively. There exist ground term graph fragments $[\tau_2,\sigma_{\tau_2}],[\tau_3,\sigma_{\tau_3}] \in K = Con(D_\ell)$ with $[g_2,\sigma_{g_2}] \odot [\tau_2,\sigma_{\tau_2}] \in L_*$, $[g_3,\sigma_{g_3}] \odot [\tau_3,\sigma_{\tau_3}] \in L_*$ and $[g_1,\sigma_{g_1}] \odot [[\tau_2,\chi_{\sigma_2}(\sigma_{\tau_2})] \odot [\tau_3,\chi_{\sigma_3}(\sigma_{\tau_3})],\xi] \notin L_*$ for some positive integer ℓ. Thus, $p_1(f_1) \leftarrow p_2(f_2),p_3(f_3)$ is removed from Γ_ℓ. This contradicts that $p_1(f_1) \leftarrow p_2(f_2),p_3(f_3)$ in Γ_m. Therefore, we can show that there exists a positive integer k such that $GL(S_n,\delta_n,p_n) = L_*$ for any integer $n \geq k$. From Proposition 1, $\lvert Con(D_n)\rvert$ is of polynomial size w.r.t. $\sum_{i=1}^n \lvert G_i\rvert$ at the n-th step. Thus, from Lemma 1, the n-th hypothesis (S_n,δ_n,p_n) is output by Algorithm 1 with polynomial update time w.r.t $\sum_{i=1}^n \lvert G_i\rvert$. □

4 Conclusions

We have considered the problem of learning FGS languages from the viewpoint of the computational learning theory. First, we introduced the class 1-FCP-$\mathcal{RFGSL}$ of regular FGS languages of bounded degree and treewidth with 1-finite context property (1-FCP). We also presented an algorithm for learning class 1-FCP-$\mathcal{RFGSL}$ by using current distributional learning techniques [11]. Finally, we showed that class 1-FCP-$\mathcal{RFGSL}$ can be identifiable in the limit with polynomial time update by using membership queries from positive data. This result will lead us to develop new techniques for learning other classes of FGS languages with distributional properties.

Clark et al. [2,3] and Yoshinaka [11,12] discussed the learnabilities of the class of languages of context-free grammars with the finite kernel property (FKP) and finite context property (FCP). As future work, we will consider the polynomial time learnabilities of the class of regular FGS languages with the FKP and FCP.

References

1. Chiang, D., Andreas, J., Bauer, D., Hermann, K.M., Jones, B., Knight, K.: Parsing graphs with hyperedge replacement grammars. In: Proceeding of ACL 2013, pp. 924–932. Association for Computational Linguistic (2013)
2. Clark, A.: A learnable representation for syntax using residuated lattices. In: Groote, P., Egg, M., Kallmeyer, L. (eds.) FG 2009. LNCS, vol. 5591, pp. 183–198. Springer, Heidelberg (2011). doi:10.1007/978-3-642-20169-1_12
3. Clark, A., Eyraud, R.: Polynomial identification in the limit of substitutable context-free languages. J. Mach. Learn. Res. **8**, 1725–1745 (2007)
4. Drewes, F., Kreowski, H.J., Habel, A.: Hyperedge replacement graph grammars. In: Handbook of Graph Grammars and Computing by Graph Transformation, vol. 1, pp. 95–162. World Scientific (1997)
5. Gildea, D.: Grammar factorization by tree decomposition. Comput. Linguist. **37**(10), 231–248 (2011)
6. Hara, S., Shoudai, T.: Polynomial time MAT learning of c-deterministic regular formal graph systems. In: Proceeding IIAI-AAI 2014, pp. 204–211. IEEE (2014)
7. Kasprzik, A., Yoshinaka, R.: Distributional learning of simple context-free tree grammars. In: Kivinen, J., Szepesvári, C., Ukkonen, E., Zeugmann, T. (eds.) ALT 2011. LNCS (LNAI), vol. 6925, pp. 398–412. Springer, Heidelberg (2011). doi:10.1007/978-3-642-24412-4_31
8. Lautemann, C.: The complexity of graph languages generated by hyperedge replacement. Acta Informatica **27**(5), 399–421 (1990)
9. Okada, R., Matsumoto, S., Uchida, T., Suzuki, Y., Shoudai, T.: Exact learning of finite unions of graph patterns from queries. In: Hutter, M., Servedio, R.A., Takimoto, E. (eds.) ALT 2007. LNCS (LNAI), vol. 4754, pp. 298–312. Springer, Heidelberg (2007). doi:10.1007/978-3-540-75225-7_25
10. Uchida, T., Shoudai, T., Miyano, S.: Parallel algorithms for refutation tree problem on formal graph systems. IEICE Trans. Inf. Syst. **E78–D**(2), 99–112 (1995)

11. Yoshinaka, R.: Integration of the dual approaches in the distributional learning of context-free grammars. In: Dediu, A.-H., Martín-Vide, C. (eds.) LATA 2012. LNCS, vol. 7183, pp. 538–550. Springer, Heidelberg (2012). doi:10.1007/978-3-642-28332-1_46
12. Yoshinaka, R.: General perspective on distributionally learnable classes. In: Proceeding of MoL 2015, pp. 87–98. Association for Computational Linguistic (2015)