IEICE TRANS. INF. & SYST., VOL.E102–D, NO.3 MARCH 2019

PAPER Special Section on Foundations of Computer Science — Algorithm, Theory of Computation, and their Applications —

Exact Learning of Primitive Formal Systems Defining Labeled Ordered Tree Languages via Queries

Tomoyuki UCHIDA[†a)], **Satoshi MATSUMOTO**[††], **Takayoshi SHOUDAI**[†††], **Yusuke SUZUKI**[†], *and* **Tetsuhiro MIYAHARA**[†], ***Members***

SUMMARY A formal graph system (FGS) is a logic programming system that directly manipulates graphs by dealing with graph patterns instead of terms of first-order predicate logic. In this paper, based on an FGS, we introduce a primitive formal ordered tree system (pFOTS) as a formal system defining labeled ordered tree languages. A pFOTS program is a finite set of graph rewriting rules. A logic program is well-known to be suitable to represent background knowledge. The query learning model is an established mathematical model of learning via queries in computational learning theory. In this learning model, we show the exact learnability of a pFOTS program consisting of one graph rewriting rule and background knowledge defined by a pFOTS program using a polynomial number of queries.

1. Introduction

Graph grammar (see [1]) has been applied to a wide range of fields including pattern recognition and image analysis. Uchida et al. [2] introduced a framework called a formal graph system (FGS) as a graph grammar system. An FGS is a logic programming system that deals with term graph patterns instead of terms of first-order predicate logic.

Large amounts of graph-structured data are now accessible on the Internet. Graph-structured data having tree structures, such as HTML/XML files, glycan data and parsing structures of natural languages, are called tree-structured data and can be represented by ordered trees. A *tree language* is a set of ordered trees. In order to represent structural features from tree-structured data, we propose an ordered term tree pattern [3], [4], which is a tree pattern that has an ordered tree structure and some internal structured variables. A variable of an ordered term tree pattern has a variable label and can be replaced with an arbitrary ordered tree by hyperedge replacement (see [1]) according to the variable label. Computational learning theory is studied in the field of foundations of computer science. In computational learning theory, the learnability of tree languages has been studied intensively [5]. We consider the problem of identifying an unknown tree language from a specified class of tree languages. In this paper, we use an ordered term tree pattern t_* as an expression of a target tree language L_* such that L_* is equal to the tree language defined using t_*. We showed in [4] that the class of tree languages defined by ordered term tree patterns is polynomial time inductively inferable from positive data. The query learning model of Angluin [6] is an established mathematical model of learning via queries in computational learning theory. In this learning model, a learning algorithm accesses oracles, which answer specific types of queries, and collects information about a target.

First of all, we introduce a *primitive* graph rewriting rule of the form $p_0(t_0) \leftarrow p_1(t_1), p_2(t_2), \ldots, p_n(t_n)$ $(n \geq 0)$ satisfying the following conditions. (1) If $n \geq 1$, then t_0 is an ordered term tree pattern all of whose variables are distinct. Otherwise, t_0 is a tree consisting of two vertices and one edge between them. (2) For each i $(1 \leq i \leq n)$, t_i is an ordered term tree pattern consisting of two vertices and one variable between them. (3) For any i $(1 \leq i \leq n)$, the variable label in t_i appears in t_0. Here, for i $(0 \leq i \leq n)$, the symbols p_i and $p_i(t_i)$ are called a predicate symbol and a literal, respectively. A primitive graph rewriting rule of the form $p(t_0) \leftarrow$ is called a *fact*.

Secondly, as a special type of FGS defining labeled ordered tree languages obtained by replacing variables of ordered term tree patterns with specified ordered trees, we introduce a *primitive formal ordered tree system (pFOTS) program*, which is a finite set of primitive graph rewriting rules. Moreover, for a pFOTS program Γ, we give a condition, called *the predicate symbol identifiable condition (PSI condition)*, on two different edge labels of two facts in Γ in order to identify one predicate symbol. According to the manner of logic programming system, the tree language defined using a pFOTS program Γ and its predicate symbol r, denoted by $\mathcal{L}(\Gamma, r)$, is defined as the set of all trees T such that there exists a derivation starting from the unit goal '$\leftarrow r(T)$' and ending with the empty goal '$\square$' derived using Γ (see [2]). In Fig. 1, as examples we give a pFOTS program Γ_{en}, a new graph rewriting rule α and the tree language $\mathcal{L}(\Gamma_{en} \cup \{\alpha\}, r)$ defined using the pFOTS program Γ_{en}, the graph rewriting rule α and the predicate symbol r. Moreover, we give a derivation $\mathcal{D}$ starting from the unit goal '$\leftarrow r(T)$' and ending with the empty goal '$\square$' derived using Γ_{en}, where T is the leftmost tree displayed in $\mathcal{L}(\Gamma_{en} \cup \{\alpha\}, r)$ in Fig. 1. The language $\mathcal{L}(\Gamma_{en} \cup \{\alpha\}, r)$ represents a set of parse trees of a natural language.

Manuscript received March 30, 2018.
Manuscript revised July 27, 2018.
Manuscript publicized October 30, 2018.

[†]The authors are with Faculty of Information Sciences, Hiroshima City University, Hiroshima-shi, 731–3194 Japan.
[††]The author is with Faculty of Science, Tokai University, Hiratsuka-shi, 259–1292 Japan.
[†††]The author is with Faculty of Contemporary Business, Kyushu International University, Kitakyushu-shi, 805–0062 Japan.
a) E-mail: uchida@hiroshima-cu.ac.jp

DOI: 10.1587/transinf.2018FCP0011

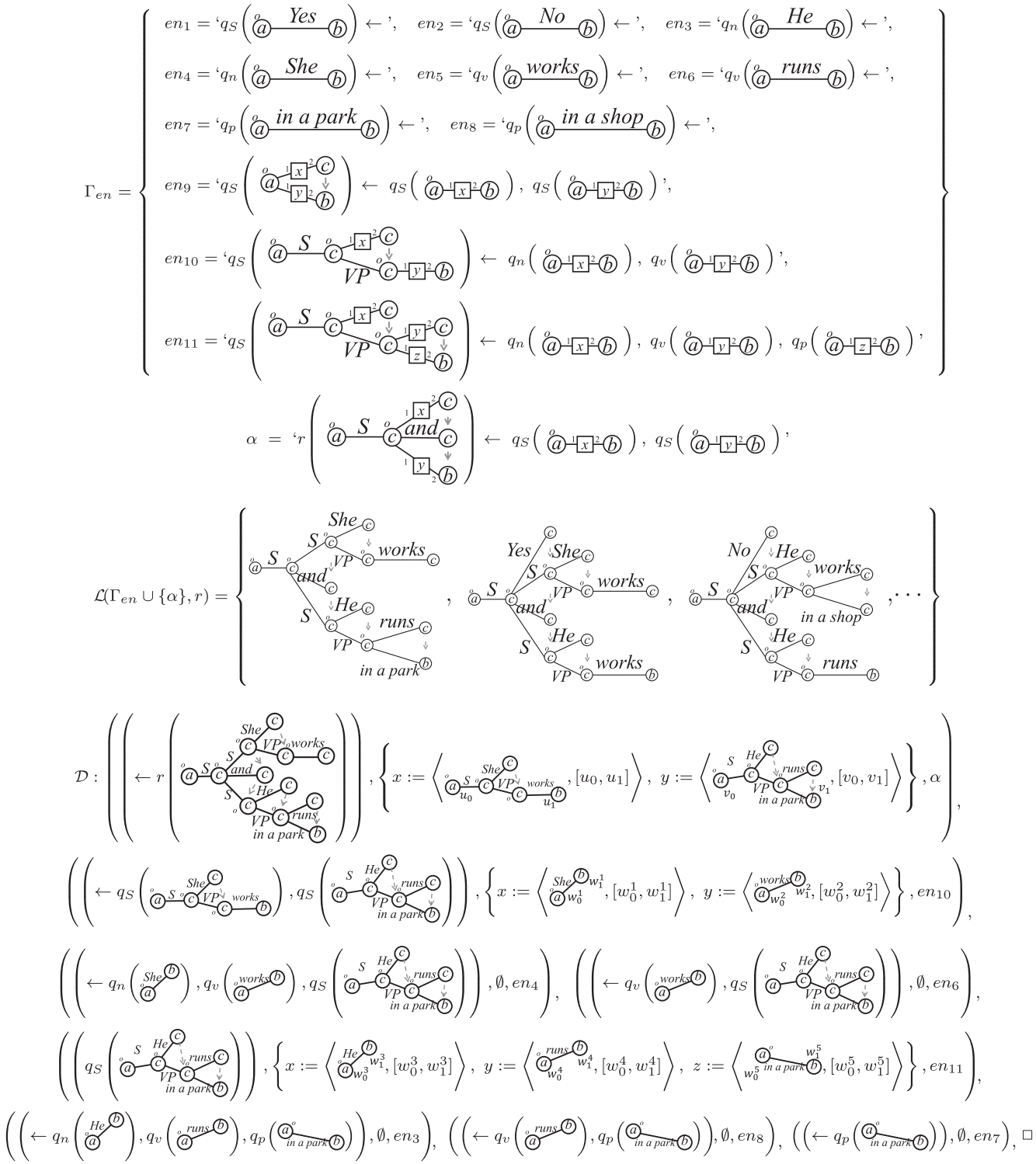

Fig. 1 pFOTS program Γ_{en}, graph rewriting rule α, ordered tree language $\mathcal{L}(\Gamma_{en} \cup \{\alpha\}, r)$, and derivation $\mathcal{D}$ starting from '$\leftarrow r(T)$' ending with the empty goal '$\square$' derived using Γ_{en}. For each tree, the leftmost vertex is the root of the tree. The children of a vertex close to "o" have siblings in the order shown with dashed arrows. A variable $[u, v]$ is denoted with a square connecting u and v which are numbered by 1 and 2, respectively.

An elementary formal system (EFS) [7] is a logic programming system over strings and is regarded as a special type of pFOTS, since any string is expressed by an ordered tree having only one leaf. We assume that our considered class does not contain the empty tree language. Under this assumption, from the insufficiency result [8] for a

special type of EFS, we can see that there exists no learning algorithm identifying a target tree language defined by a pFOTS program using a polynomial number of equivalence, membership and subset queries. This insufficiency result is due to the fact that a positive example is not necessarily obtained by using equivalence and subset queries. A logic program is well-known to be suitable to represent background knowledge. We call an FGS program or a pFOTS program given as background knowledge a *background FGS program* or a *background pFOTS program*, respectively. Let Γ be a pFOTS program as background knowledge satisfying the PSI condition. Thirdly, we present learning algorithms of a target tree language defined using a pFOTS program consisting of one graph rewriting rule and a background pFOTS program Γ satisfying the PSI condition, by using one positive example and a polynomial number of membership queries with respect to the number of edges of the positive example. Our learning methods use some operators based on a background pFOTS program. Therefore, even if the given pFOTS program and the tree language are changed, our learning algorithms find the target graph rewriting rule.

Finally, by giving examples of FGS programs, we discuss the learnability of a target tree language defined using an FGS program consisting of one graph rewriting rule and a background FGS program that does not satisfy at least one of the conditions of a primitive graph rewriting rule and the PSI condition, by using one positive example and a polynomial number of membership queries.

For the learning of ordered tree languages, Suzuki et al. [4] showed that the class of languages defined by ordered term tree patterns is polynomial time inductively inferable from positive data. Matsumoto et al. [9] showed that finite unions of ordered term tree pattern languages are exactly learnable with a polynomial number of queries. For the learning of graph grammars, Okada et al. [10] showed that some classes of graph pattern languages defined by FGS programs are exactly learnable with a polynomial number of equivalence and restricted subset queries. Shoudai et al. [11] showed that the regular FGS languages of bounded degree with the 1-finite context property and bounded treewidth property are learnable from positive data and membership queries with current distributional learning techniques [12].

This paper is organized as follows. In Sect. 2, we introduce an ordered term tree pattern [3], [4], a primitive formal ordered tree system and the PSI condition on a pFOTS program, and explain the query learning model [6] briefly. In Sect. 3, we propose a learning algorithm that exactly identifies a target tree language defined using one graph rewriting rule and a background pFOTS program having only one predicate symbol. In Sect. 4, we present a learning algorithm that identifies a target tree language defined using one graph rewriting rule and a background pFOTS program having two or more predicate symbols. In Sect. 5, under the assumption that a background FGS program Ξ is not a pFOTS program, we discuss whether or not a target tree language defined using one graph rewriting rule and Ξ is exactly identified by using one positive example and a polynomial number of membership queries. This paper is the full version of the paper [13], with proofs, extended results and complete definitions.

2. Preliminaries

2.1 Term Tree Patterns

Let Σ and Λ be finite alphabets, and X an infinite alphabet. We assume $(\Sigma \cup \Lambda) \cap X = \emptyset$. Let T be a rooted ordered tree (a *tree*, for short). We denote by $V(T)$ and $E(T)$ the vertex set and the edge set of T, respectively. We denote by (u_0, u_1) the edge between u_0 and u_1 such that u_1 is a child of u_0. For a set or list S, the number of elements in S is denoted by $|S|$.

Definition 1: Let T be a tree and H_T a subset of $E(T)$. A *term tree pattern t obtained from T and H_T* is a 3-tuple (V, E, H) where $V = V(T)$, $E = E(T) \setminus H_T$ and $H = H_T$. An element (u_0, u_1) in H_T is called a *variable* of t and denoted by $[u_0, u_1]$ instead of (u_0, u_1). Every vertex and edge of T is labeled with a symbol in Σ and Λ, respectively. Every variable $h = [u_0, u_1]$ is labeled with a symbol x in X. Such a symbol x is called a *variable label* of h. We call u_0 the *parent port* of h and u_1 the *child port* of h.

For a term tree pattern t obtained from T and H_T, if T and H_T are clear from the context, we omit them, i.e., we write a term tree pattern t simply. For a term tree pattern t, we denote by $V(t)$, $E(t)$, and $H(t)$ the vertex set, the edge set, and the variable set of t, respectively. Moreover we denote the vertex label of $v \in V(t)$ by $\varphi_t(v) \in \Sigma$, the edge label of $e \in E(t)$ by $\psi_t(e) \in \Lambda$, and the variable label of $h \in H(t)$ by $\lambda_t(h) \in X$. For two vertices $u, u' \in V(t)$, we say that u is the *parent* of u' in t if u is the parent of u' in T. Similarly we say that u' is a *child* of u in t if u' is a child of u in T. For a vertex $u \in V(t)$ with no child, we call u a *leaf* of t. We define the order of the children of each internal vertex u in t as the order of the children of u in T.

Definition 2: We say that two term tree patterns t and t' are *isomorphic*, denoted by $t \cong t'$, if there is a bijection f from $V(t)$ to $V(t')$ such that for any $u, u', v, v' \in V(t)$ the following conditions 1–7 hold:

1. the root of t is mapped to the root of t' by f,
2. u is the next sibling of u' if and only if $f(u)$ is the next sibling of $f(u')$,
3. $(u, u') \in E(t)$ if and only if $(f(u), f(u')) \in E(t')$,
4. $[u, u'] \in H(t)$ if and only if $[f(u), f(u')] \in H(t')$,
5. $\varphi_t(u) = \varphi_{t'}(f(u))$,
6. $\psi_t((u, u')) = \psi_{t'}((f(u), f(u')))$, and
7. $\lambda_t([u, u']) = \lambda_t([v, v'])$ if and only if $\lambda_{t'}([f(u), f(u')]) = \lambda_{t'}([f(v), f(v')])$.

A term tree pattern t is *linear* if all variables in t have mutually distinct variable labels in X. The set of all linear term tree patterns is denoted by $\mathcal{LOTT}$. A term tree pattern is said to be *primitive* if it consists of two vertices and one variable between them. We regard a term tree pattern t with

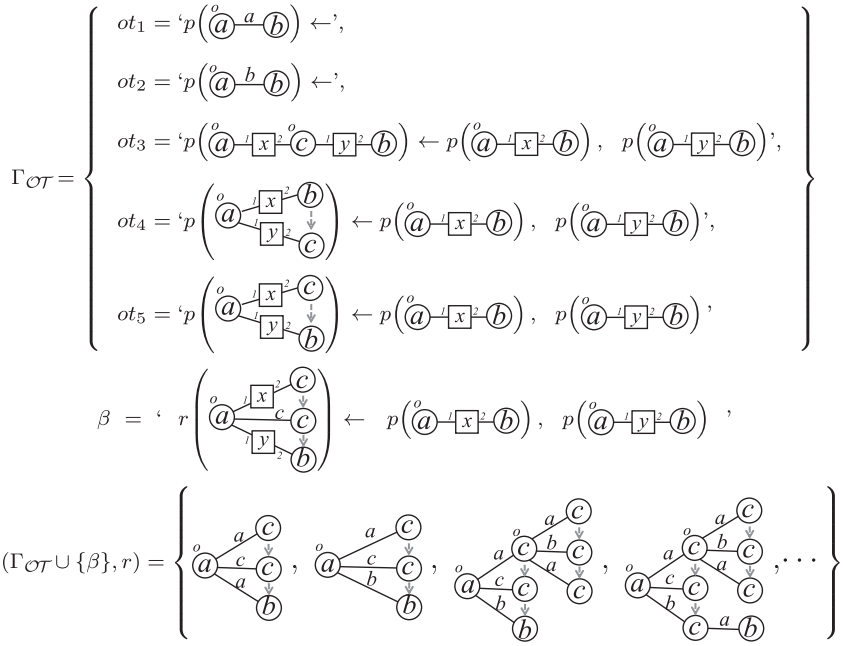

Fig. 2 pFOTS program $\Gamma_{\mathcal{OT}}$, graph rewriting rule β and the ordered tree language $L(\Gamma_{\mathcal{OT}} \cup \{\beta\}, r)$ defined using the pFOTS program $\Gamma_{\mathcal{OT}} \cup \{\beta\}$ and predicate symbol r.

$H(t) = \emptyset$ as a tree. We denote by $\mathcal{OT}$ the set of all trees.

Let T be a tree having at least two vertices and t a term tree pattern having at least one variable. Let x be a variable label in t and $\sigma = [u_0, u_1]$ a list of two distinct vertices in T where u_0 is the root of T and u_1 is a leaf of T. Then, the form $x := \langle T, \sigma \rangle$ is called a *binding* for x. A new term tree pattern t' is obtained by applying the binding $x := \langle T, \sigma \rangle$ to t in the following way. For each variable $h = [v_0, v_1]$ labeled with x in t, we attach a copy T' of T to t by removing h from $H(t)$ and by identifying the vertices v_0 and v_1 with the vertices u'_0 and u'_1 of T', respectively, where the vertices u'_0 and u'_1 of T' correspond to the vertices u_0 and u_1 of T, respectively. We define a new sibling ordering on every vertex v in t' in a natural way [4], [14]. A *substitution* θ is a finite set of bindings for distinct variable labels. The term tree pattern obtained from t by applying all bindings in θ to t simultaneously is denoted by $t\theta$. For a term tree pattern t, the term tree pattern language of t, denoted by $L(t)$, is the set of all trees obtained from t by replacing all variables in t with arbitrary trees.

2.2 Primitive Formal Ordered Tree System

Let Π be a set of unary predicate symbols. We assign a list of two distinct vertex labels in Σ to every predicate symbol p in Π. The list is called the *pointer* of p and denoted by *pointer*(p). An *atom* is an expression of the form $p(t)$. We use *pointer*(p) to specify how to bind a tree to a variable of t in an atom $p(t)$. The first label of *pointer*(p) specifies the root of t and the second specifies one of the leaves of t. In Sect. 2.1, we gave the definition of a binding $x := \langle T, \sigma \rangle$. In any binding, we need to give the list σ of vertices. Roughly speaking, if T is a tree that is generated by a predicate symbol p, the first (resp. second) vertex of σ is the vertex of T that has the first (resp. second) label of *pointer*(p). In Figs. 1 and 2, all predicate symbols $q_s, q_n, \ldots$ have the same pointer (a, b).

Let $A, B_1, \ldots, B_n$ be atoms ($n \geq 0$). A *graph rewriting rule* (*rule*, for short) is a clause of the form $A \leftarrow B_1, \ldots, B_n$. The atom A is called the *head* and the part $B_1, \ldots, B_n$ is called the *body* of the rule. A rule is called a *fact* if $n = 0$. We denote by $o(t, x)$ the number of variables in t labeled with x.

Definition 3: A rule $p(t) \leftarrow q_1(t_1), \ldots, q_n(t_n)$ is said to be *primitive* if the following conditions 1–3 hold.

1. If $n = 0$, i.e., the rule is a fact, then t is a tree consisting of two vertices and one edge between them.
2. If $n \geq 1$, then the following conditions are satisfied.
   a. t is a linear term tree pattern such that $|V(t)| > 2$ holds.
   b. Every t_i ($1 \leq i \leq n$) is a primitive term tree pattern.
3. For every variable $x \in X$, $o(t, x) = o(t_1, x) + \cdots + o(t_n, x) \leq 1$.

For example, all rules in Γ_{en} and the rule α in Fig. 1 are primitive. Moreover, all rules in $\Gamma_{\mathcal{OT}}$ and a rule β in Fig. 2 are also primitive.

Definition 4: A finite set Γ of primitive rules is said to be a *primitive Formal Ordered Tree System program* (*pFOTS program*, for short) if for each predicate symbol p appearing in Γ, there is at least one fact whose predicate symbol is p.

For a rule $\alpha =$‘$p(t) \leftarrow q_1(t_1), \ldots, q_n(t_n)$’, we denote by $\Pi^h(\alpha)$ and $\Pi^b(\alpha)$ the sets of all predicate symbols in

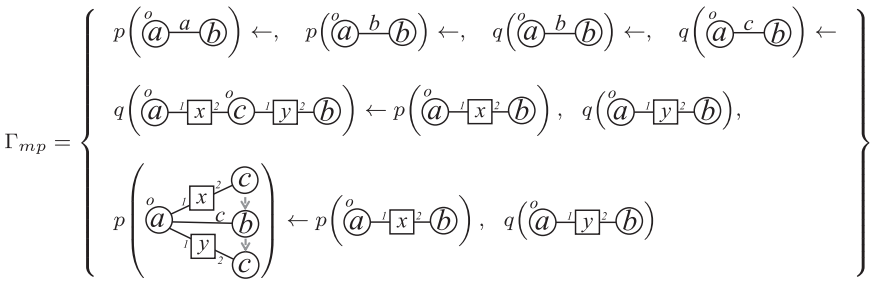

Fig. 3 pFOTS program Γ_{mp}

the head and body of α, respectively, that is, $\Pi^h(\alpha) = \{p\}$ and $\Pi^b(\alpha) = \{q_1, \ldots, q_n\}$. For a pFOTS program Γ, we denote by $\Pi(\Gamma)$ the set of all predicate symbols in Γ, that is, $\Pi(\Gamma) = \bigcup_{\alpha\in\Gamma} \Pi^h(\alpha) \cup \Pi^b(\alpha)$. For any predicate symbol $p \in \Pi(\Gamma)$, we denote by $\Lambda(\Gamma, p)$ the set of all edge labels in the facts in Γ whose predicate symbols are p. Figure 2 shows a pFOTS program $\Gamma_{\mathcal{OT}}$, where $\Pi(\Gamma_{\mathcal{OT}}) = \{p\}$ and $\Lambda(\Gamma_{\mathcal{OT}}, p) = \{a, b\}$.

A pFOTS program Γ satisfies *the predicate symbol identifiable condition* (*PSI condition*, for short) if for each predicate symbol $p \in \Pi(\Gamma)$, there exist two edge labels $a_1, a_2 \in \Lambda$ $(a_1 \neq a_2)$ such that $P(a_1) \cap P(a_2) = \{p\}$, where $P(a) = \{q \in \Pi(\Gamma) \mid$ '$q(T) \leftarrow$ ' $\in \Gamma$ s.t. T has the edge label $a\}$. A pair of such edge labels like a_1 and a_2 is called a *predicate identifier* of p. For example, for the pFOTS program Γ_{mp} in Fig. 3, since $P(a) = \{p\}$, $P(b) = \{p, q\}$ and $P(c) = \{q\}$, for two edge labels $a, b \in \Lambda$, $P(a) \cap P(b) = \{p\}$ and $P(b) \cap P(c) = \{q\}$ hold. Then, the predicate identifiers of p and q are the pairs (a, b) and (b, c), respectively. From the definition of the PSI condition, we remark that, if a pFOTS program Γ satisfies the PSI condition, then for every predicate symbol $p \in \Pi(\Gamma)$, Γ has at least two facts having p as their predicate symbols. This means that, if $|\Pi(\Gamma)| = \{p\}$ holds, Γ has at least two facts whose predicate symbol is p.

For a pFOTS program Γ and a predicate symbol $r \in \Pi \setminus \Pi(\Gamma)$, we denote by $p\mathcal{GRR}(\Gamma, r)$ the set of all primitive rules α such that the predicate symbol of the head of α is r, i.e., $r \in \Pi^h(\alpha)$ and if α is not a fact, then $\Pi^b(\alpha) \subseteq \Pi(\Gamma)$ holds.

Let t_1 and t_2 be linear term tree patterns. Let p and q be predicate symbols. A substitution θ is called a *unifier* of atoms $p(t_1)$ and $q(t_2)$ if p and q are the same predicate symbol and $t_1\theta \cong t_2\theta$ holds. A *goal* is a rule of the form $\leftarrow B_1, \ldots, B_m$ $(m \geq 0)$. It is called a *unit goal* if $m = 1$ and the *empty goal* if $m = 0$. For a rule α, we denote by $var(\alpha)$ the set of all variable labels in α. Let α be a rule '$p_0(t_0^\alpha) \leftarrow p_1(t_1^\alpha), \ldots, p_k(t_k^\alpha)$' and β a rule '$p_0(t_0^\beta) \leftarrow p_1(t_1^\beta), \ldots, p_k(t_k^\beta)$'. We say that β is a *variant* of α if there are two substitutions θ, θ' and a permutation $\xi : \{1, \ldots, k\} \to \{1, \ldots, k\}$ such that $t_0^\alpha\theta \cong t_0^\beta$ and $t_0^\alpha \cong t_0^\beta\theta'$ and, for any i $(1 \leq i \leq k)$, $t_i^\alpha\theta \cong t_{\xi(i)}^\beta$ and $t_i^\alpha \cong t_{\xi(i)}^\beta\theta'$ hold.

Definition 5: ([2], [15]) Let Γ be a pFOTS program and β a goal. A *derivation* from β derived using Γ is a sequence of 3-tuples $(\beta_i, \theta_i, \alpha_i)$ $(i = 0, 1, \ldots)$ satisfying the following conditions:

1. β_i is a goal, θ_i is a substitution, and α_i is a variant of a rule in Γ.
2. $\beta_0 = \beta$.
3. $var(\alpha_i) \cap var(\alpha_j) = \emptyset$ for any i, j $(i \neq j)$ and $var(\alpha_i) \cap var(\beta) = \emptyset$ for any i.
4. We assume that there is a procedure, say $\mathcal{A}$, which chooses an atom from the body of a goal. Let β_i be a goal $\leftarrow A_1, \ldots, A_k$ and α_i a rule $A \leftarrow B_1, \ldots, B_q$. Let A_m $(1 \leq m \leq k)$ be an atom in the body of β_i, which is chosen by $\mathcal{A}$. Then θ_i is a unifier of A and A_m, and β_{i+1} is a goal $\leftarrow A_1\theta_i, \ldots, A_{m-1}\theta_i, B_1\theta_i, \ldots, B_q\theta_i, A_{m+1}\theta_i, \ldots, A_k\theta_i$.

Definition 6: ([2], [15]) Let Γ be a pFOTS program and β_0 a goal. A *refutation* derived using Γ is a finite derivation from β_0 derived using Γ ending with the empty goal. Let $F = \{(\beta_i, \theta_i, \alpha_i)\}_{0\leq i\leq \ell}$ be a refutation from a unit goal β derived using Γ. A *refutation tree* $\mathcal{T}_{\ell+1}$ of F is a tree satisfying the following conditions:

1. Every vertex is labeled with a unit goal or the empty goal.
2. The label of the root is the unit goal $\beta_0 = \beta$.
3. Every leaf is labeled with the empty goal.
4. $\mathcal{T}_0$ is a tree consisting of only one vertex labeled with β. For any $i = 0, \ldots, \ell$, $\mathcal{T}_{i+1}$ is a tree obtained from $\mathcal{T}_i$ by applying $(\beta_i, \theta_i, \alpha_i)$ to $\mathcal{T}_i$ as follows: let β_i be a goal $\leftarrow A_1^i, \ldots, A_{n_i}^i$. We assume that $\mathcal{T}_i$ has a leaf labeled with $\leftarrow A_j^i$ $(1 \leq j \leq n_i)$. Let α_i be a rule $A^i \leftarrow B_1^i, \ldots, B_{q_i}^i$ $(q_i \geq 0)$ and θ_i a unifier of A^i and A_j^i. If $q_i = 0$, that is, α_i is a fact $A^i \leftarrow$, then $\mathcal{T}_{i+1}$ is obtained by adding a new vertex labeled with empty goal, as the child of the vertex that is labeled with $\leftarrow A_j^i$. Otherwise, $\mathcal{T}_{i+1}$ is obtained by adding new q_i vertices labeled with $\leftarrow B_1^i\theta_i, \ldots, \leftarrow B_{q_i}^i\theta_i$, respectively, as the children of the vertex that is labeled with $\leftarrow A_j^i$.

Figure 4 shows an example of a refutation from a unit goal derived using the pFOTS program $\Gamma_{\mathcal{OT}}$ in Fig. 2 and its refutation tree.

For a pFOTS program Γ and a predicate symbol p in $\Pi(\Gamma)$, we denote by $L(\Gamma, p)$ the set of all trees $T \in \mathcal{OT}$ such that there exists a refutation from the unit goal $\leftarrow p(T)$ derived using Γ. We say that a subset $L \subseteq \mathcal{OT}$ is a *pFOTS lan-*

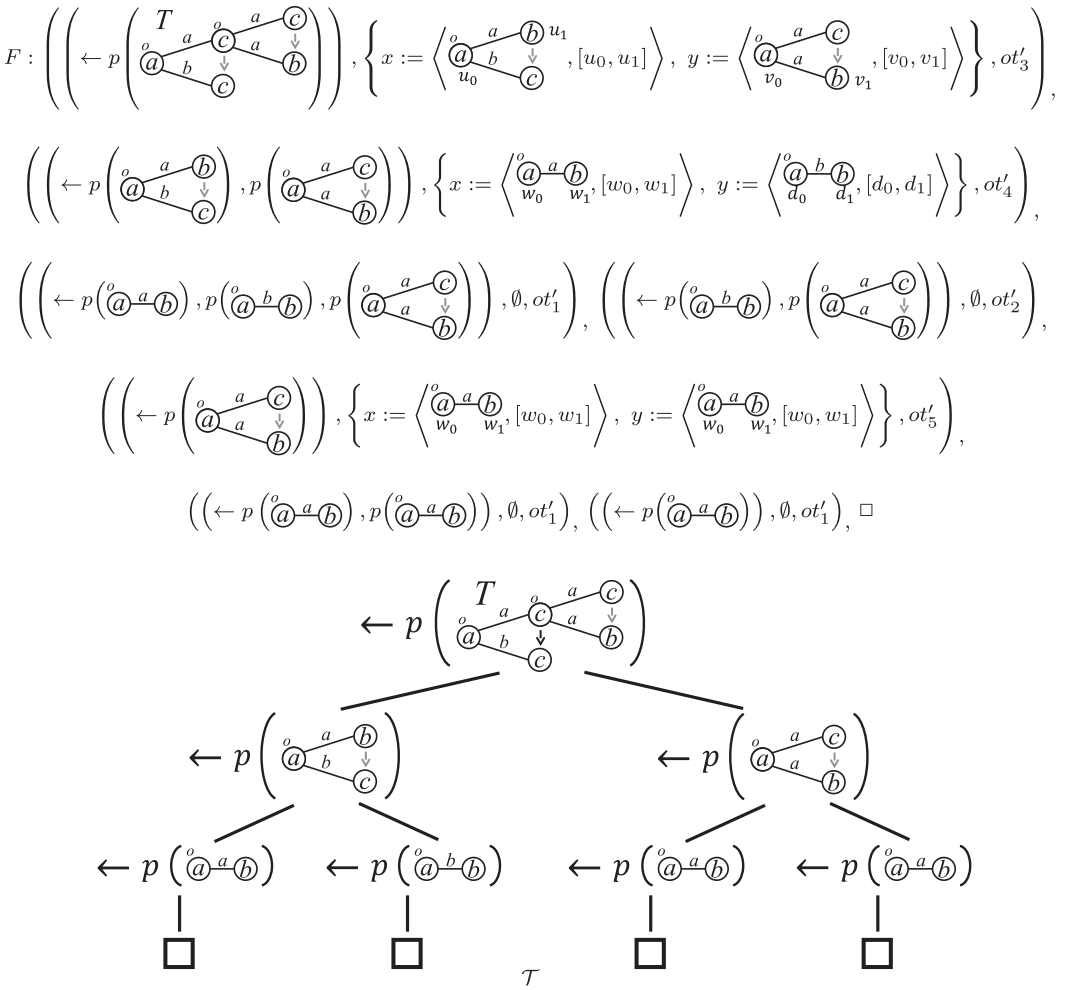

Fig. 4 Tree T in $\mathcal{LOTT}$, refutation F from the unit goal $\leftarrow p(T)$ derived using $\Gamma_{\mathcal{OT}}$ and refutation tree $\mathcal{T}$ of F. For each i ($1 \le i \le 5$), ot'_i is a variant of a rule ot_i in Fig. 2. The square box with thick line denotes the empty goal.

guage if there exists a pFOTS program Γ and its predicate symbol p such that $L = L(\Gamma, p)$ holds. We note that for any pFOTS program Γ and any predicate symbol $p \in \Pi$, $L(\Gamma, p) \neq \emptyset$ holds, since Γ has at least one fact whose predicate symbol is p from Definition 4 of a pFOTS program. It is easy to see that for the pFOTS program $\Gamma_{\mathcal{OT}}$ in Fig. 2, $L(\Gamma_{\mathcal{OT}}, p) = \mathcal{OT}$ holds. Furthermore, Fig. 2 shows the pFOTS language $L(\Gamma_{\mathcal{OT}} \cup \{\beta\}, r)$ defined using the pFOTS program $\Gamma_{\mathcal{OT}} \cup \{\beta\}$.

2.3 Query Learning Model

Let $\mathcal{L}$ be a class consisting of sets of trees such that each set in $\mathcal{L}$ has its own representation of finite length. Let $\mathcal{R}$ be the set of representations for all sets of trees in $\mathcal{L}$. For each representation $\pi \in \mathcal{R}$, we denote by $L(\pi)$ the set of trees that is represented by π. For the set $\mathcal{LOTT}$ of representations, $\mathcal{L}(\mathcal{LOTT}) = \{L(t) \mid t \in \mathcal{LOTT}\}$ is the class defined using $\mathcal{LOTT}$. Let Γ be a pFOTS program that can be considered background knowledge and r a predicate symbol with $r \in \Pi \setminus \Pi(\Gamma)$. Then, for the set $p\mathcal{GRR}(\Gamma, r)$ of representations, $p\mathcal{GRRL}(\Gamma, r) = \{L(\Gamma \cup \{\alpha\}, r) \mid \alpha \in p\mathcal{GRR}(\Gamma, r)\}$ is the class defined using $p\mathcal{GRR}(\Gamma, r)$. Hereafter we call a pFOTS program given as background knowledge a *background pFOTS program*.

We consider the query learning framework proposed by Angluin [6]. Let $L_* \in \mathcal{L}$ be a learning target. Let $\pi_* \in \mathcal{R}$ be one of the representations of L_*, i.e., $L_* = L(\pi_*)$. A tree $T \in \mathcal{OT}$ is said to be a *positive example* of L_* if $T \in L_*$. A learning algorithm can access *oracles* that will answer queries about the target L_*. We consider the following tree kinds of query.

Membership query (**MQ**): The input is a tree $T \in \mathcal{OT}$. The output is `yes` if $T \in L_*$, and no otherwise.

Restricted subset query (**rSQ**): The input is a representation $\pi \in \mathcal{R}$. The output is `yes` if $L(\pi) \subseteq L(\pi_*)$, and no otherwise.

Equivalence query (**EQ**): The input is a representation $\pi \in \mathcal{R}$. If $L(\pi) = L(\pi_*)$, the output is `yes` otherwise the output is a tree in $(L(\pi) \setminus L(\pi_*)) \cup (L(\pi_*) \setminus L(\pi))$, called a *counterexample*.

A learning algorithm $\mathcal{A}$ is said to *exactly identify a tar-*

get $L_* \in \mathcal{L}$ if $\mathcal{A}$ outputs a representation $\pi \in \mathcal{R}$ satisfying $L(\pi) = L(\pi_*)$. In the next section, we will present a learning algorithm that exactly identifies a target $L_* \in p\mathcal{GRRL}(\Gamma, r)$ using one positive example and a polynomial number of membership queries, that is, the algorithm outputs a primitive rule $\alpha_* \in p\mathcal{GRR}(\Gamma, r)$ satisfying $L(\Gamma \cup \{\alpha_*\}, r) = L_*$.

3. Exact Learning of Tree Languages with Background pFOTS Programs Having Only One Predicate Symbol via Queries

In this paper, for a background pFOTS program Γ and the set $p\mathcal{GRR}(\Gamma, r)$ of representations with $r \in \Pi \setminus \Pi(\Gamma)$, we consider the learnability of the class $p\mathcal{GRRL}(\Gamma, r)$ using one positive example and a polynomial number of membership queries. From Theorem 16 in [4], if a background pFOTS program Γ satisfies the PSI condition, a target tree language L_* in $p\mathcal{GRRL}(\Gamma, r)$ is identified by identifying a rule α_* in $p\mathcal{GRR}(\Gamma, r)$ such that $L_* = L(\Gamma \cup \{\alpha_*\}, r)$ holds. Hence, by using one positive example and a polynomial number of membership queries, we construct a primitive rule $\alpha_* \in p\mathcal{GRR}(\Gamma, r)$ that satisfies $L_* = L(\Gamma \cup \{\alpha_*\}, r)$ on the basis of the following strategy.

(1) Construct contraction operators from Γ.
(2) Construct a minimal tree $T_{\min}$ with respect to the number of edges in L_* by recursively applying the constructed contraction operators to a given positive example in L_*.
(3) Construct a target primitive rule α_* from $T_{\min}$ such that $L_* = L(\Gamma \cup \{\alpha_*\}, r)$ holds as follows.
   (a) Identify a term tree pattern in the head of α_* by replacing some edges in $T_{\min}$ with appropriate variables.
   (b) Identify all predicate symbols in the body of α_*.

For a background pFOTS program Γ satisfying the PSI condition and the set $p\mathcal{GRR}(\Gamma, r)$ of representations with $r \in \Pi \setminus \Pi(\Gamma)$, in Sect. 3.2 and Sect. 4, we give two query learning algorithms based on the above strategy for two cases of $|\Pi(\Gamma)| = 1$ and $|\Pi(\Gamma)| \geq 2$, respectively. In the case of $|\Pi(\Gamma)| = 1$ (Sect. 3.2), it is not necessary to identify predicate symbols in the body of a target primitive rule α_*, that is, process (3)-(b) of the strategy is unnecessary. On the other hand, in the case of $|\Pi(\Gamma)| \geq 2$ (Sect. 4), we have to identify predicate symbols in the body of a target primitive rule α_* since, $L(\Gamma, p) \neq L(\Gamma, q)$ holds for two distinct predicate symbols $p, q \in \Pi(\Gamma)$. To make our results easier to understand, we explain (1) and (2) of the above strategy in detail in Sect. 3.1. Moreover, in Sect. 3.2, we show results for the case of $|\Pi(\Gamma)| = 1$. Finally, in Sect. 4, we show results for the case of $|\Pi(\Gamma)| \geq 2$.

3.1 Algorithm for Finding Minimum Positive Example by Contraction Operators

In this subsection, we explain our ideas for (1) and (2) in the above strategy.

Let Γ be a pFOTS program. For convenience, i.e., in order to describe our learning algorithm simply, we use two kinds of special symbols that are not in $\Sigma \cup \Lambda$. For any $p \in \Pi(\Gamma)$, we use the symbol ϵ to specify two vertex labels in $pointer(p)$, and the symbols "$?_p$" to specify the edge labels in $\Lambda(\Gamma, p)$. Let p be a predicate symbol of the head of a rule in Γ. Let $\alpha =$ '$p(t_0^\alpha) \leftarrow p_1(t_1^\alpha), \ldots, p_n(t_n^\alpha)$' $(n \geq 1)$ be a primitive rule in Γ and $\beta =$ '$p(t^\beta) \leftarrow$ ' a fact in Γ. Let $H(t_0^\alpha) = \{h_1, \ldots, h_n\}$. Without loss of generality, we assume that for any i $(1 \leq i \leq n)$, the variable label of h_i is the same as that of the unique variable of t_i^α. Let P_α be the tree obtained from t_0^α in the following way: each h_i $(1 \leq i \leq n)$ is replaced with an edge whose label is the special symbol "$?_{p_i}$", and each vertex label in $pointer(p)$ is changed to the special symbol ϵ. Let Q_β be the tree obtained from t^β by replacing the labels of two vertices of t^β with the special symbol ϵ. The pair (P_α, Q_β) is called the *contraction pair* of α and β. For a pFOTS program Γ and a predicate symbol $p \in \Pi(\Gamma)$, the *contraction pair set* of Γ and p, denoted by $CPS(\Gamma, p)$, is the set of all contraction pairs of α and β that satisfy the following conditions 1–3:

1. α is a primitive rule in Γ with $\Pi^b(\alpha) \neq \emptyset$,
2. β is a fact in Γ such that the edge label appearing in the head of β is the lexicographically first symbol in $\Lambda(\Gamma, p)$,
3. $\Pi^h(\alpha) = \Pi^h(\beta) = \{p\}$.

The *contraction pair set* of Γ, denoted by $CPS(\Gamma)$, is the set $\bigcup_{p \in \Pi(\Gamma)} CPS(\Gamma, p)$.

Example 1: For the pFOTS program Γ_{en} in Fig. 1, we present the contraction pair set $CPS(\Gamma_{en}) = \{(P_1^{en}, Q_{No}^{en}), (P_2^{en}, Q_{No}^{en}), (P_3^{en}, Q_{No}^{en})\}$ of Γ_{en}, where $P_1^{en}, P_2^{en}, P_3^{en}$ and Q_{No}^{en} are shown in Fig. 5.

Example 2: For the pFOTS program $\Gamma_{\mathcal{OT}}$ in Fig. 2, we present the contraction pair set $CPS(\Gamma_{\mathcal{OT}}) = \{(P_1^{\mathcal{OT}}, Q_a^{\mathcal{OT}}), (P_2^{\mathcal{OT}}, Q_a^{\mathcal{OT}}), (P_3^{\mathcal{OT}}, Q_a^{\mathcal{OT}})\}$ of $\Gamma_{\mathcal{OT}}$, where $P_1^{\mathcal{OT}}$, $P_2^{\mathcal{OT}}$, $P_3^{\mathcal{OT}}$ and $Q_a^{\mathcal{OT}}$ are shown in Fig. 6.

Let T be a tree in $\mathcal{OT}$. A subgraph S of T is said to be a *subtree* of T if S is a tree. A vertex u in $V(S)$ is called a *boundary vertex* of S if u is adjacent to a vertex in $V(T) \setminus V(S)$. A subtree S of T is said to be a 2-*port subtree* of T if the root of S is either the root of T or a boundary vertex of S, and all leaves of S except at most one boundary vertex are leaves of T. Let r_S be the root of S and ℓ_S the boundary vertex that is a leaf of S if it exists.

For a 2-port subtree S of T and a contraction pair (P, Q) of Γ, we say that S *matches* P, denoted by $S \approx P$, if there is a bijection $f : V(S) \to V(P)$ that satisfies the following conditions 1–5:

1. the root of S is mapped to the root of P by f,
2. for any u and u' in $V(S)$, u is the next sibling of u' if and only if $f(u)$ is the next sibling of $f(u')$,
3. for any u and u' in $V(S)$, $(u, u') \in E(S)$ if and only if $(f(u), f(u')) \in E(P)$,

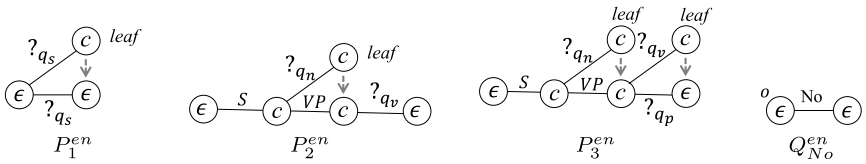

Fig. 5 Trees $P_1^{en}, P_2^{en}, P_3^{en}$ and Q_{No}^{en} constructed from Γ_{en} in Fig. 1.

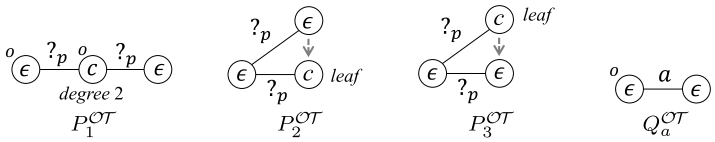

Fig. 6 Trees $P_1^{\mathcal{OT}}, P_2^{\mathcal{OT}}, P_3^{\mathcal{OT}}$ and $Q_a^{\mathcal{OT}}$ constructed from $\Gamma_{\mathcal{OT}}$ in Fig. 2.

4. for any $u \in V(S)$, if u is neither r_S nor ℓ_S, then $\varphi_S(u) = \varphi_P(f(u))$ holds, and
5. for any $(u, u') \in E(S)$, if $\psi_P((f(u), f(u'))) =$"$?_p$" for some $p \in \Pi(\Gamma)$, then $\psi_S((u, u')) \in \Lambda(\Gamma, p)$ holds, otherwise $\psi_S((u, u')) = \psi_P((f(u), f(u')))$.

Note that if ℓ_S exists, then $\varphi_P(f(\ell_S)) = \epsilon$ holds.

Let (P, Q) be a contraction pair in $CPS(\Gamma)$. A *contraction operator*, denoted by $\triangleright_{(P,Q)}(T, S)$, is a function that returns a new tree from a tree T and a 2-port subtree S of T as input. If $S \not\approx P$ holds, we define the tree $\triangleright_{(P,Q)}(T, S)$ as T. We assume that $S \approx P$ holds. Let f be a bijection from $V(S)$ to $V(P)$ realizing $S \approx P$. Let ℓ'_S be the leaf of S such that $\varphi_P(f(\ell'_S)) = \epsilon$ holds. If ℓ_S exists, then $\ell'_S = \ell_S$ holds. The tree $\triangleright_{(P,Q)}(T, S)$ is obtained from T by removing the vertices in $V(S) \setminus \{r_S, \ell'_S\}$ and the edges in $E(S)$, and by identifying r_S of S with the root of Q and ℓ'_S of S with the leaf of Q.

The following lemma on pFOTS plays an important role in our algorithms.

Lemma 1: Let Γ be a pFOTS program, r a predicate symbol in $\Pi \setminus \Pi(\Gamma)$, α a primitive rule in $pG\mathcal{RR}(\Gamma, r)$, and t_α a linear term tree pattern appearing in the head of α. For any tree $T \in L(\Gamma \cup \{\alpha\}, r)$, if $|V(T)| > |V(t_\alpha)|$ holds, then there are a 2-port subtree S of T and a contraction pair $(P, Q) \in CPS(\Gamma)$ such that $\triangleright_{(P,Q)}(T, S) \in L(\Gamma \cup \{\alpha\}, r)$ holds.

Proof. Since $T \in L(\Gamma \cup \{\alpha\}, r)$, there is a refutation tree $\mathcal{T}$ from the unit goal $\leftarrow r(T)$. Let $v_{\mathcal{T}}$ be the root of $\mathcal{T}$. Since $|V(T)| > |V(t_\alpha)|$, there is a proper descendant χ of $v_{\mathcal{T}}$ all of whose children in $\mathcal{T}$ are labeled with facts in Γ. Let $\leftarrow p(S')$ be the label of χ for a tree $S' \in \mathcal{OT}$ and a predicate symbol $p \in \Pi(\Gamma)$. Then there are a rule $p(t_0) \leftarrow q_1(t_1), \ldots, q_n(t_n)$ and a substitution θ such that $S' \cong t_0\theta$ holds and all $t_i\theta$ $(1 \le i \le n)$ are trees of only one edge. From the definition of the contraction pair set of Γ and p, there is a contraction pair (P, Q) in $CPS(\Gamma, p)$ such that $S' \approx P$ holds. From the definition of a refutation tree, there is a 2-port subtree S of T such that S is isomorphic to S' except the two vertex labels of S' specified by $pointer(p)$. Therefore $S \approx P$ holds. The refutation tree of the tree $\triangleright_{(P,Q)}(T, S)$ is obtained from $\mathcal{T}$ by removing all the children and grandchildren of χ, and by replacing χ with the vertex χ' having the unit goal $\leftarrow p(Q)$ as its label and its child having the empty goal as its label. Therefore the tree $\triangleright_{(P,Q)}(T, S) \in L(\Gamma \cup \{\alpha\}, r)$ holds. □

Procedure 1 CONTRACTION

Input: A tree T in the target L_* and the contraction pair set $CPS(\Gamma)$ of Γ.
Output: A tree $T_{\min} \in L(t_*)$ s.t. $|V(T_{\min})| \le |V(T')|$ holds for all $T' \in L_*$.

```
 1: T_min := T;
 2: repeat
 3:     for all (P, Q) ∈ CPS(Γ) do
 4:         for all subtrees S of T_min satisfying S ≈ P do
 5:             T_tmp := ▷_(P,Q)(T_min, S);
 6:             if |V(T_tmp)| < |V(T_min)| then
 7:                 if MQ(T_tmp) = yes then
 8:                     T_min := T_tmp;
 9:                 end if
10:             end if
11:         end for
12:     end for
13: until T_min does not change;
14: output T_min;
```

Lemma 2: Let Γ be a pFOTS program, r a predicate symbol in $\Pi \setminus \Pi(\Gamma)$, α a primitive rule in $pG\mathcal{RR}(\Gamma, r)$, and t_α a linear term tree pattern appearing in the head of α. For any tree $T \in L(\Gamma \cup \{\alpha\}, r)$ and the contraction pair set $CPS(\Gamma)$ of Γ, Procedure 1 (CONTRACTION) finds a tree $T_{\min} \in L(\Gamma \cup \{\alpha\}, r)$ with $|V(T_{\min})| = |V(t_\alpha)|$ using $O(n^2)$ membership queries where $n = |V(T)|$.

Proof. We will prove that, given any tree $T \in L(\Gamma \cup \{\alpha\}, r)$ and the contraction pair set $CPS(\Gamma)$ of Γ as inputs, Procedure 1 (CONTRACTION) outputs a tree $T_{\min}$ obtained from T such that $|V(T_{\min})| = |V(t_\alpha)|$ holds. We assume that $|V(T_{\min})| > |V(t_\alpha)|$ holds. From Lemma 1, there are a 2-port subtree S of $T_{\min}$ and a contraction pair $(P, Q) \in CPS(\Gamma)$ such that $\triangleright_{(P,Q)}(T_{\min}, S) \in L(\Gamma \cup \{\alpha\}, r)$ and $|V(\triangleright_{(P,Q)}(T_{\min}, S))| < |V(T_{\min})|$ hold. Therefore if $|V(T_{\min})| > |V(t_\alpha)|$, the repeat-loop of Procedure 1 does not finish. Hence $|V(T_{\min})| = |V(t_\alpha)|$ holds. The number of vertices of $T_{\min}$ decreases by at least one every iteration at lines 2–13 of Procedure 1. The for-loop at lines 4–11 of Procedure 1 enumerates $O(n)$ subtrees, since the number of subtrees of T that are nearly isomorphic to the first element of one pair of $CPS(\Gamma)$ is $O(|CPS(\Gamma)| \cdot n)$. Note that $|CPS(\Gamma)| \le |\Gamma|$ holds. Then, since

Algorithm 2 LEARNING$_{\langle\Gamma,r\rangle}$

Input: A tree $T \in \mathcal{OT}$ that is a positive example of a target tree language L_*, i.e., $T \in L_*$.
Output: A primitive rule $\alpha \in p\mathcal{GRR}(\Gamma, r)$ satisfying $L(\Gamma \cup \{\alpha\}, r) = L_*$.
1: // *Preprocessing Step*
2: Make the contraction pair set $CPS(\Gamma)$ of Γ;
3: // *Contraction Step*
4: $T_{\min} :=$ CONTRACTION$(T, CPS(\Gamma))$; (Procedure 1)
5: // *Variable Replacement Step*
6: $t :=$ VARIABLE_REPLACEMENT$(T_{\min})$; (Procedure 3)
7: // *Postprocessing Step*
8: $\alpha :=$ MAKE_RULE(t); (Procedure 4)
9: **output** α;

Procedure 3 VARIABLE_REPLACEMENT

Input: A tree $T_{\min}$ in the target L_*.
Output: A term tree pattern $t \in \mathcal{LOTT}$.
1: $t := T_{\min}$;
2: **for all** $e \in E(T_{\min})$ **do**
3: $\quad T_{tmp} := T_{\min}$;
4: $\quad \psi_{T_{tmp}}(e) := a$, where $a \in \Lambda(\Gamma, p) \setminus \{\psi_{T_{\min}}(e)\}$;
5: $\quad$ **if** $\mathbf{MQ}(T_{tmp}) =$ `yes` **then**
6: $\quad\quad h_e := [u, u']$, where $e = (u, u')$;
7: $\quad\quad t_{tmp} := (V(t), E(t) \setminus \{e\}, H(t) \cup \{h_e\})$;
8: $\quad\quad \lambda_{t_{tmp}}(h_e) := x$, where $x \in X \setminus \{\lambda_t(h) \mid h \in H(t)\}$;
9: $\quad\quad t := t_{tmp}$;
10: $\quad$ **end if**
11: **end for**
12: **output** t;

the number of primitive rules in Γ is constant, Procedure 1 uses $O(n^2)$ membership queries. □

3.2 Learning Algorithm Given Background pFOTS Program Having Only One Predicate Symbol

Let Γ be a background pFOTS program such that $\Pi(\Gamma) = \{p\}$ holds and r a predicate symbol in $\Pi \setminus \Pi(\Gamma)$. For a target tree language $L_* \in p\mathcal{GRRL}(\Gamma, r)$ with $r \in \Pi \setminus \Pi(\Gamma)$, Algorithm 2 (LEARNING$_{\langle\Gamma,r\rangle}$) finds a primitive rule $\alpha \in p\mathcal{GRR}(\Gamma, r)$ satisfying $L_* = L(\Gamma \cup \{\alpha\}, r)$ by using one positive example and membership queries in the following way. First of all, Algorithm 2 makes the contraction pair set $CPS(\Gamma)$ of Γ from the pFOTS program Γ. Secondly, in Procedure 1 (CONTRACTION), Algorithm 2 constructs the minimal tree $T_{\min}$ in the target language L_* by recursively applying contraction operators defined using $CPS(\Gamma)$ to the tree given as one positive example. Thirdly, in Procedure 3 (VARIABLE_REPLACEMENT), Algorithm 2 constructs a term tree pattern t obtained from $T_{\min}$ by recursively replacing edges of t with variables using membership queries. Finally, in Procedure 4 (MAKE_RULE), Algorithm 2 makes the primitive rule $\alpha \in p\mathcal{GRR}(\Gamma, r)$ from t.

We will prove the correctness and the time complexity of Algorithm 2 (LEARNING$_{\langle\Gamma,r\rangle}$) when for a background pFOTS program Γ, $|\Pi(\Gamma)| = 1$ holds. From Lemma 1, we have the following theorem.

Theorem 1: Let Γ be a fixed pFOTS program such that Γ has at least two facts and $|\Pi(\Gamma)| = 1$ holds, r a predicate symbol in $\Pi \setminus \Pi(\Gamma)$, and L_* a target tree language in $p\mathcal{GRRL}(\Gamma, r)$. Then, a primitive rule $\alpha \in p\mathcal{GRR}(\Gamma, r)$ satisfying $L_* = L(\Gamma \cup \{\alpha\}, r)$ is exactly identified using $O(n^2)$ membership queries and one positive example $T \in L_*$, where $n = |V(T)|$.

Proof. Algorithm 2 (LEARNING$_{\langle\Gamma,r\rangle}$) consists of four steps. In *Preprocessing Step*, the contraction pair set $CPS(\Gamma)$ of Γ is created. In *Contraction Step*, Algorithm 2 decreases the size of $T_{\min}$ using the contraction operators constructed from $CPS(\Gamma)$ repeatedly while $\mathbf{MQ}(T_{tmp})$ returns `yes`. After that, in *Variable Replacement Step*, Algorithm 2 makes a linear term tree pattern t by replacing edges of $T_{\min}$ with new variables if $\mathbf{MQ}(T_{tmp})$ returns `yes`. Then, in *Postprocessing Step*, Algorithm 2 makes a primitive rule α from t such that the predicate symbol of the head of α is r and $L_* = L(\Gamma \cup \{\alpha\}, r)$ holds.

Let α_* be a primitive rule in $p\mathcal{GRR}(\Gamma, r)$ such that $L_* = L(\Gamma \cup \{\alpha_*\}, r)$ holds, and t_* the linear term tree pattern appearing in the head of α_*. From Lemma 2, in *Contraction Step*, for a tree $T \in L_*$ and the contraction pair set $CPS(\Gamma)$ of Γ, Algorithm 2 finds a tree $T_{\min} \in L_*$ with $|V(T_{\min})| = |V(t_*)|$ using $O(n^2)$ membership queries. Here, we will prove that after *Variable Replacement Step*, $t \cong t_*$ holds. We see that after *Contraction Step*, there is a bijection $f : V(T_{\min}) \to V(t_*)$ such that for any $u, u' \in V(T_{\min})$, $(u, u') \in E(T_{\min})$ if and only if $(f(u), f(u')) \in E(t_*)$ or $[f(u), f(u')] \in H(t_*)$. In particular, if $(f(u), f(u')) \in E(t_*)$, then $\psi_{T_{\min}}((u, u')) = \psi_{t_*}((f(u), f(u')))$ holds. In the for-loop at lines 2–11 of Procedure 3, for an edge $e = (u, u') \in E(T_{\min})$, if $(f(u), f(u')) \in E(t_*)$, $\mathbf{MQ}(T_{tmp})$ does not return `yes`. That is, if $\mathbf{MQ}(T_{tmp})$ returns `yes`, then $[f(u), f(u')] \in H(t_*)$ holds. Conversely, if $[f(u), f(u')] \in H(t_*)$, whatever the edge label of $e = (u, u')$ is, $\mathbf{MQ}(T_{tmp})$ returns `yes`. Therefore, we have $t \cong t_*$. Trivially, in *Variable Replacement Step*, Procedure 3 uses at most n membership queries. Then Algorithm 2 (LEARNING$_{\langle\Gamma,r\rangle}$) uses $O(n^2)$ membership queries. Hence, the theorem holds. □

From Theorem 1, when the pFOTS program $\Gamma_{\mathcal{OT}}$, which defines the set of all ordered trees, is given as background knowledge, we can see that the following corollary holds.

Corollary 1: Let $\Gamma_{\mathcal{OT}}$ be the pFOTS program in Fig. 2 and r a predicate symbol in $\Pi \setminus \Pi(\Gamma_{\mathcal{OT}})$. Then, Algorithm 2 (LEARNING$_{\langle\Gamma_{\mathcal{OT}},r\rangle}$) exactly identifies a target tree language in

Procedure 4 MAKE_RULE

Input: A linear term tree pattern $t \in \mathcal{LOTT}$.
Output: A primitive rule $\alpha \in p\mathcal{GRR}(\Gamma, r)$.
1: Let $H(t) = \{h_1, \ldots, h_\ell\}$;
2: **for** $i := 1$ **to** ℓ **do**
3: $\quad$ Let u_i and v_i be distinct new vertices;
4: $\quad t_i := (\{u_i, v_i\}, \emptyset, \{[u_i, v_i]\})$;
5: $\quad (\varphi_{t_i}(u_i), \varphi_{t_i}(v_i)) :=$ *pointer*(p);
6: $\quad \lambda_{t_i}([u_i, v_i]) := \lambda_t(h_i)$;
7: **end for**
8: **output** $\alpha =$ ' $r(t) \leftarrow p(t_1), \ldots, p(t_\ell)$ ';

Algorithm 5 LEARNINGPLURALPRED$_{\langle\Gamma,r\rangle}$

Input: A tree $T \in \mathcal{OT}$ that is a positive example of a target tree language L_*, i.e., $T \in L_*$.
Output: A primitive rule $\alpha \in p\mathcal{GRR}(\Gamma, r)$ satisfying $L(\Gamma \cup \{\alpha\}, r) = L_*$.
1: // *Preprocessing Step*
2: Make the contraction pair set $CPS(\Gamma)$ of Γ;
3: // *Contraction Step*
4: $T_{\min}$:= CONTRACTION$(T, CPS(\Gamma))$; (Procedure 1)
5: // *Variable Replacement Step*
6: $(t, \mathcal{X})$:= VARIABLE_REPLACEMENT$(T_{\min})$; (Procedure 6)
7: // *Postprocessing Step*
8: α := MAKE_RULE$(t, \mathcal{X})$; (Procedure 7)
9: **output** α;

$\mathcal{L}(\mathcal{LOTT})$ using one positive example T and $O(n^2)$ membership queries, where $n = |V(T)|$.

4. Exact Learning of Tree Languages with Background pFOTS Programs Having Two or More Predicate Symbols with Queries

In this section, for a background pFOTS program Γ satisfying the PSI condition and the set $p\mathcal{GRR}(\Gamma, r)$ of representations with $r \in \Pi \setminus \Pi(\Gamma)$, we consider the learnability of the class $p\mathcal{GRRL}(\Gamma, r)$ in the query learning model under the assumption of $|\Pi(\Gamma)| \geq 2$. If a background pFOTS program Γ have two or more predicate symbols such as Γ_{en} with $|\Pi(\Gamma_{en})| = 4$ in Fig. 1, it is necessary to determine the predicate symbols in the body of a target rule. We identify those predicate symbols in *Variable Replacement Step*, while identifying the variables of the term tree pattern in the head of the target rule.

Let Γ be a pFOTS program such that Γ satisfies the PSI condition and $|\Pi(\Gamma)| \geq 2$ holds. We can present a learning algorithm, denoted by Algorithm 5 (LEARNINGPLURALPRED$_{\langle\Gamma,r\rangle}$), by replacing Procedures 3 and 4 of Algorithm 2 with Procedures 6 and 7, respectively. By using Algorithm 5, given a pFOTS program having two or more predicate symbols and satisfying the PSI condition as background knowledge, we have the following theorem.

Theorem 2: Let Γ be a background pFOTS program such that Γ satisfies the PSI condition and $|\Pi(\Gamma)| \geq 2$, r a predicate symbol in $\Pi \setminus \Pi(\Gamma)$, and L_* a target tree language in $p\mathcal{GRRL}(\Gamma, r)$. Then, a primitive rule $\alpha \in p\mathcal{GRR}(\Gamma, r)$ satisfying $L_* = L(\Gamma \cup \{\alpha\}, r)$ is exactly identified using one positive example $T \in L_*$ and $O(n^2)$ membership queries with $n = |V(T)|$.

Proof. In a similar way to Theorem 1, the correctness of Algorithm 5 (LEARNINGPLURALPRED$_{\langle\Gamma,r\rangle}$) can be shown. Here we will estimate the time complexity of Algorithm 5. Since Γ is a pFOTS program satisfying the PSI condition, *Variable Replacement Step* is recursively applied to a tree $T_{\min}$ obtained after *Contraction Step* at most $2 \cdot V(T_{\min})$ times. After that, we obtain a primitive rule α by using Procedure 7 such that $L(\Gamma \cup \{\alpha\}, r) = L_*$ holds. Hence, since $|V(T_{\min})| \leq n$, α is exactly identified using $O(|\Gamma| \cdot n^2)$ membership queries and one positive example. Since $|\Gamma|$ is constant, the statement holds. □

Procedure 6 VARIABLE_REPLACEMENT

Input: A tree $T_{\min}$ in the target L_*.
Output: A term tree pattern $t \in \mathcal{LOTT}$ and a collection of sets of variable labels $\{X_p\}_{p \in \Pi(\Gamma)}$.
1: $t := T_{\min}$;
2: **for all** predicate symbols $p \in \Pi(\Gamma)$ **do**
3: Let (a_p, b_p) be a predicate identifier of p;
4: $X_p := \emptyset$;
5: **end for**
6: **for all** $e \in E(T_{\min})$ **do**
7: $T^1_{tmp} := T_{\min}$;
8: $T^2_{tmp} := T_{\min}$;
9: **for all** predicate symbols $p \in \Pi(\Gamma)$ **do**
10: $\psi_{T^1_{tmp}}(e) := a_p$ and $\psi_{T^2_{tmp}}(e) := b_p$;
11: **if** $\mathbf{MQ}(T^1_{tmp})$ = `yes` and $\mathbf{MQ}(T^2_{tmp})$ = `yes` **then**
12: $h_e := [u, u']$, where $e = (u, u')$;
13: $t_{tmp} := (V(t), E(t) \setminus \{e\}, H(t) \cup \{h_e\})$;
14: $\lambda_{t_{tmp}}(h_e) := x$, where $x \in X \setminus \{\lambda_t(h) \mid h \in H(t)\}$;
15: $X_p := X_p \cup \{x\}$;
16: $t := t_{tmp}$;
17: break;
18: **end if**
19: **end for**
20: **end for**
21: **output** t **and** $\{X_p\}_{p \in \Pi(\Gamma)}$;

Procedure 7 MAKE_RULE

Input: A linear term tree pattern $t \in \mathcal{LOTT}$ and a collection $\mathcal{X}$ of sets of variable labels $\{X_p\}_{p \in \Pi(\Gamma)}$.
Output: A primitive rule $\alpha \in p\mathcal{GRR}(\Gamma, r)$.
1: Let $H(t) = \{h_1, \ldots, h_\ell\}$;
2: **for** $i := 1$ **to** ℓ **do**
3: Let p_i be a predicate symbol in $\Pi(\Gamma)$ s.t. $\lambda_t(h_i) \in X_{p_i}$;
4: Let u_i and v_i be distinct new vertices;
5: $t_i := (\{u_i, v_i\}, \emptyset, \{[u_i, v_i]\})$;
6: $(\varphi_{t_i}(u_i), \varphi_{t_i}(v_i)) := pointer(p_i)$;
7: $\lambda_{t_i}([u_i, v_i]) := \lambda_t(h_i)$;
8: **end for**
9: **output** $\alpha =$ ' $r(t) \leftarrow p_1(t_1), \ldots, p_\ell(t_\ell)$ ';

We can extend pFOTS programs to deal with the class of ordered tree languages in case that $|\Lambda|$ is infinite. by using the special atom determining whether or not the edge label is in Λ. Therefore, it is easy to see that Theorems 1 and 2 hold if $|\Lambda|$ is infinite.

5. Discussion

In this section, under some relaxed conditions of pFOTS programs, that is, the assumption that background knowledge Γ has rules that do not satisfy at least one of the conditions 1 and 2 in Definition 3, we discuss whether or not a target tree language $L_* = L(\Gamma \cup \{\alpha_*\}, r)$ with $\alpha_* \in p\mathcal{GRR}(\Gamma, r)$ is exactly identified using one positive example and a polynomial number of membership queries.

To identify a target tree language L_* in $p\mathcal{GRRL}(\Gamma, r)$ with background knowledge Γ and $r \in \Pi \setminus \Pi(\Gamma)$, we need to construct a linear term tree pattern t_* in the head of a target

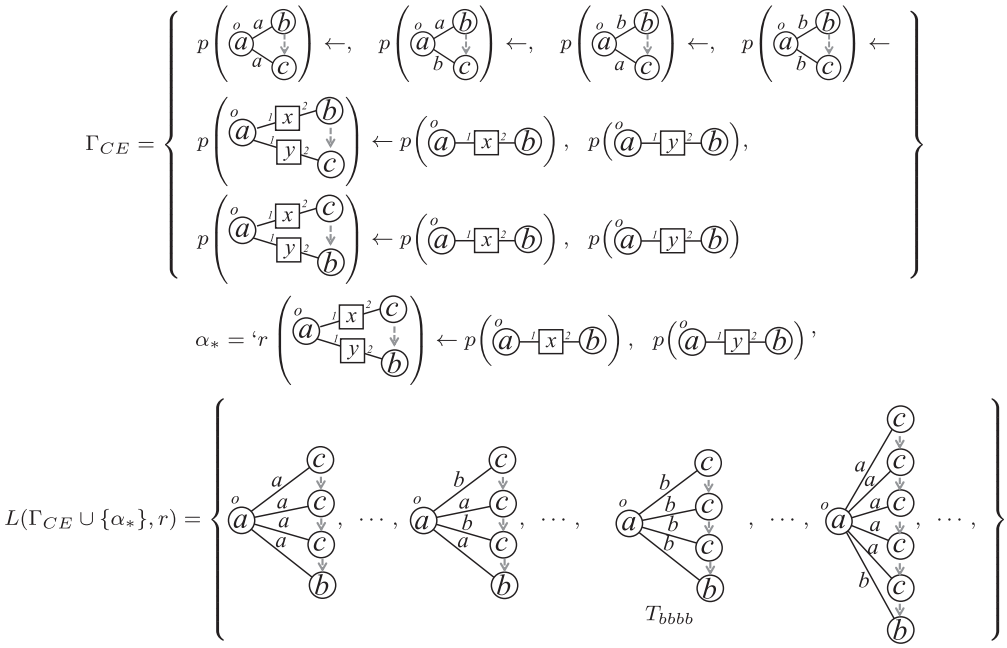

Fig. 7 FGS program Γ_{CE}, which is not a pFOTS program, target primitive rule α_* and language $L(\Gamma_{CE} \cup \{\alpha_*\}, r)$

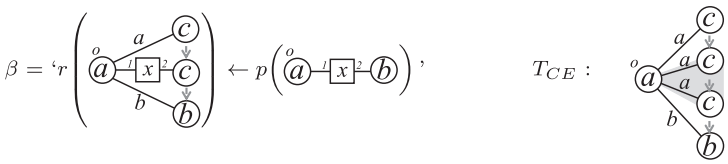

Fig. 8 Candidate primitive rule β and a tree T_{CE} in $\mathcal{OT}$

primitive rule α_* such that $L_* = L(\Gamma \cup \{\alpha_*\}, r)$ holds. If Γ is a pFOTS program, we can see that $|E(t_*)| + |H(t_*)| = |E(T_{min})|$ holds, where T_{min} is a minimal tree such that there exists a refutation from the unit goal $\leftarrow r(T_{min})$ derived using $\Gamma \cup \{\alpha_*\}$. Therefore, we can construct a target term tree pattern t_* obtained from $E(T_{min})$ by replacing edges with variables recursively while membership queries return "yes". We consider the case that Γ has a fact '$p(t) \leftarrow$' such that t consists of two or more edges. Let T_{min} be a minimal tree such that there exists a refutation F from the unit goal $\leftarrow r(T_{min})$ derived using $\Gamma \cup \{\alpha_*\}$ satisfying $(\beta, \theta, `p(t) \leftarrow ') \in F$. Then, we can see that $|E(t_*)| + |H(t_*)| < |E(T_{min})|$ holds. Therefore, we need to construct a linear term tree pattern t_* from T_{min} by replacing subtrees isomorphic to trees of facts in Γ with variables using a polynomial number of membership queries with respect to $|E(T_{min})|$. Consider the FGS program Γ_{CE} to be background knowledge and the target rule α_* in Fig. 7. We remark that any fact in Γ_{CE} has a tree consisting of two edges, that is, none of the facts in Γ_{CE} satisfy the condition 1 of Definition 3. Suppose that the minimal tree T_{CE} in Fig. 8 is constructed by recursively applying the contraction operators to a positive example, after *Contraction Step* (or T_{CE} is given as one positive example). In *Variable Replacement Step*, the subtree of T_{CE} that is described in the gray region is tried to be replaced with a variable by replacing it with each tree appearing in facts of Γ_{CE} and inquiring to oracle as membership query. Since any membership query answers "yes", the candidate primitive rule β in Fig. 8 can be constructed. However, $L(\Gamma_{CE} \cup \{\beta\}, r) \neq L(\Gamma_{CE} \cup \{\alpha_*\}, r)$ holds, since $T_{bbbb} \in L(\Gamma_{CE} \cup \{\alpha_*\}, r)$ but $T_{bbbb} \notin L(\Gamma_{CE} \cup \{\beta\}, r)$. This example shows that identification of the target language may fail even if we recursively apply the variable replacements to T_{min} while membership queries return "yes". For a similar reason, we can see that identification of a target tree language may fail for any background knowledge containing rules having variables consisting of 3 ports or more fail.

Moreover, let a pFOTS program Γ have at most one fact for each predicate symbol in $\Pi(\Gamma)$. We can easily construct a target primitive rule α_* without any membership query, if Γ satisfies the following condition. For each predicate symbol p in $\Pi(\Gamma)$, the edge label in $\Lambda(\Gamma, p)$ does not appear in a target graph rewriting rule α_* or any rule in Γ except a fact whose predicate symbol is p. Otherwise, since we can use membership queries only, we need to identify edges replaced with variables by using subtrees instead of edge labels. If Γ is a pFOTS program such that $L(\Gamma, p)$ equals the set of all ordered trees having only one edge label, we conjecture that, using a similar strategy to that of the proof of Theorem 22 in [4], a target tree language defined

using a pFOTS program such as Γ and a predicate symbol $r \in \Pi \setminus \Pi(\Gamma)$ can be identified by using one positive example and a polynomial number of membership queries with respect to the number of edges of the positive example. Otherwise, that is, if Γ is an arbitrary pFOTS program, to identify a target tree language L_* by using one positive example and a polynomial number of membership queries, we need a new strategy for constructing an appropriate primitive rule α such that $L(\Gamma \cup \{\alpha\}, r) = L_*$ holds.

To identify a target tree language defined using a background FGS program Ξ, such as Γ_{CE} and one graph rewriting rule in $p\mathcal{GRR}(\Xi, r)$, by using one positive example and a polynomial number of membership queries, Procedures 3 of $\textsc{Learning}_{\langle \Xi, r \rangle}$ or 6 of $\textsc{LearningPluralPred}_{\langle \Xi, r \rangle}$ is necessary to be modified.

6. Conclusions

We have introduced a primitive formal ordered tree system (pFOTS) as a formal system defining ordered tree languages based on FGS [2]. For a pFOTS program Γ as background knowledge, we have shown the exact learnability of the class $p\mathcal{GRRL}(\Gamma, r)$ of tree languages defined using pFOTS programs each of which consists of one primitive graph rewriting rule and Γ in cases of $|\Pi(\Gamma)| = 1$ and $|\Pi(\Gamma)| \geq 2$, by using one positive example and a polynomial number of membership queries. Moreover, by giving an FGS program Γ_{CE} whose facts do not satisfy the condition 1 of Definition 3 as background knowledge, we have shown that a target tree language defined using Γ_{CE} and a primitive rule cannot be identified, by our learning algorithm using one positive example and a polynomial number of membership queries.

As future work, we will consider the exact learnability of tree languages defined using pFOTS programs as background knowledge in case that the number of facts for each predicate symbol is only one, by using one positive example and membership queries. Let a pFOTS program Γ have at most one fact for each predicate symbol in $\Pi(\Gamma)$. Kato et al. [8] showed that any target language of all strings defined using a special type of EFS [7] is identified, by using a polynomial number of membership and superset queries. Since any string is expressed by an ordered tree having only one leaf, we will consider the learnability of tree languages defined using pFOTS programs without background knowledge by expanding Kato's results to pFOTS programs. Moreover we will expand our results to pFOTS programs without background knowledge using more powerful queries such as equivalence queries, subset queries, superset queries and predicate membership queries (see [6], [16]).

Acknowledgements

This study was partially supported by Grant-in-Aid for Scientific Research (C) (Grant Numbers JP15K00312, JP15K00313, JP17K00321) from Japan Society for the Promotion of Science (JSPS).

References

[1] G. Rozenberg, ed., Handbook of Graph Grammars and Computing by Graph Transformation: Volume I. Foundations, World Scientific Publishing Co., Inc., River Edge, NJ, USA, 1997.
[2] T. Uchida, T. Shoudai, and S. Miyano, "Parallel algorithms for refutation tree problem on formal graph systems," IEICE Trans. Inf. & Syst., vol.E78-D, no.2, pp.99–112, 1995.
[3] S. Matsumoto, Y. Hayashi, and T. Shoudai, "Polynomial time inductive inference of regular term tree languages from positive data," Proc. ALT-97, Springer-Verlag, LNAI, vol.1316, pp.212–227, 1997.
[4] Y. Suzuki, T. Shoudai, T. Uchida, and T. Miyahara, "Ordered term tree languages which are polynomial time inductively inferable from positive data," Theoretical Computer Science, vol.350, no.1, pp.63–90, 2006.
[5] J. Björklund and H. Fernau, "Learning tree languages," in Topics in Grammatical Inference, ed. J. Heinz and J.M. Sempere, ch. 7, pp.173–213, Springer-Verlag, 2016.
[6] D. Angluin, "Queries and concept learning," Machine Learning, vol.2, no.4, pp.319–342, 1988.
[7] S. Arikawa, T. Shinohara, and A. Yamamoto, "Learning elementary formal systems," Theoretical Computer Science, vol.95, no.1, pp.97–113, 1992.
[8] H. Kato, S. Matsumoto, and T. Miyahara, "Learning of elementary formal systems with two clauses using queries," IEICE Trans. Inf. & Syst., vol.E92-D, no.2, pp.172–180, 2009.
[9] S. Matsumoto, T. Shoudai, T. Uchida, T. Miyahara, and Y. Suzuki, "Learning of finite unions of tree patterns with internal structured variables from queries," IEICE Trans. Inf. & Syst., vol.E91-D, no.2, pp.222–230, 2008.
[10] R. Okada, S. Matsumoto, T. Uchida, Y. Suzuki, and T. Shoudai, "Exact learning of finite unions of graph patterns from queries," Proc. ALT 2007, LNAI, vol.4754, pp.298–312, Springer, 2007.
[11] T. Shoudai, S. Matsumoto, and Y. Suzuki, "Distributional learning of regular formal graph system of bounded degree," Proc. ILP 2016, LNAI, vol.10326, pp.68–80, Springer, 2016.
[12] A. Clark and R. Yoshinaka, "Distributional learning of context-free and multiple context-free grammars," in Topics in Grammatical Inference, ed. J. Heinz and J.M. Sempere, ch. 6, pp.143–172, Springer-Verlag, 2016.
[13] T. Uchida, S. Matsumoto, T. Shoudai, Y. Suzuki, and T. Miyahara, "Learning of primitive formal systems defining labeled ordered tree languages via queries," Late Breaking Papers of the 27th International Conference on Inductive Logic Programming (LBP-ILP 2017), pp.61–66, CEUR-WS.org, http://ceur-ws.org/Vol-2085/, 2018.
[14] Y. Suzuki, T. Shoudai, T. Uchida, and T. Miyahara, "An efficient pattern matching algorithm for ordered term tree patterns," IEICE Trans. Fundamentals, vol.E98-A, no.6, pp.1197–1211, 2015.
[15] J.W. Lloyd, Foundations of Logic Programming, Springer-Verlag, 1993.
[16] H. Sakamoto, K. Hirata, and H. Arimura, "Learning elementary formal systems with queries," Theoretical Computer Science, vol.298, no.1, pp.21–50, 2003.

Tomoyuki Uchida received the B.S. degree in Mathematics, the M.S. and Dr. Sci. degrees in Information Systems all from Kyushu University, in 1989, 1991 and 1994, respectively. Currently, he is an associate professor of Graduate School of Information Sciences, Hiroshima City University. His research interests include data mining from semistructured data, algorithmic graph theory and algorithmic learning theory.

Satoshi Matsumoto is an associate professor of Department of Mathematical Sciences, Tokai University, Kanagawa, Japan. He received the B.S. degree in Mathematics, the M.S. and Dr. Sci. degrees in Information Systems all from Kyushu University, Fukuoka, Japan in 1993, 1995 and 1998, respectively. His research interests include algorithmic learning theory.

Takayoshi Shoudai received the B.S. in 1986, the M.S. degree in 1988 in Mathematics and the Dr. Sci. in 1993 in Information Science all from Kyushu University. Currently, he is a professor of Faculty of Contemporary Business, Kyushu International University. His research interests include graph algorithms, computational learning theory, and data mining.

Yusuke Suzuki received the B.S. degree in Physics, the M.S. and Dr. Sci. degrees in Informatics all from Kyushu University, in 2000, 2002 and 2007, respectively. He is currently a research associate of Graduate School of Information Sciences, Hiroshima City University, Hiroshima, Japan. His research interests include machine learning and data mining.

Tetsuhiro Miyahara is an associate professor of Graduate School of Information Sciences, Hiroshima City University, Hiroshima, Japan. He received the B.S. degree in Mathematics, the M.S. and Dr. Sci. degrees in Information Systems all from Kyushu University, Fukuoka, Japan in 1984, 1986 and 1996, respectively. His research interests include algorithmic learning theory, knowledge discovery and machine learning.