Learning of Primitive Formal Systems Defining Labelled Ordered Tree Languages via Queries

Tomoyuki Uchida
Graduate School of Information Sciences,
Hiroshima City University, Japan
uchida@hiroshima-cu.ac.jp

Satoshi Matsumoto
Faculty of Science,
Tokai University, Japan
matsumoto@tokai-u.jp

Takayoshi Shoudai
Faculty of Contemporary Business,
Kyushu International University, Japan
shoudai@isb.kiu.ac.jp

Yusuke Suzuki
Graduate School of Information Sciences,
Hiroshima City University, Japan
y-suzuki@hiroshima-cu.ac.jp

Tetsuhiro Miyahara
Graduate School of Information Sciences,
Hiroshima City University, Japan
miyares18@info.hiroshima-cu.ac.jp

Abstract.

A formal graph system (FGS) is a logic programming system that directly manipulates graphs by dealing with term graph patterns instead of terms of first-order predicate logic. In this paper, based on FGS, we introduce a primitive formal ordered tree system (pFOTS) as a formal system defining labelled 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 with a pFOTS program using a polynomial number of queries.

1 Introduction

Graph grammar (see [6]) has been applied to a wide range of fields including pattern recognition and image analysis. Uchida et al. [10] introduced a framework called a formal graph system (FGS) as a graph grammar. An FGS is a logic programming system that directly manipulates graphs by dealing with term graph patterns instead of terms of first-order predicate logic.

Recently, many graph structured data become accessible on Internet. Graph structured data such as XML files having tree structures are called tree structured data. Ordered tree can represent such graph data such as XML files, glycan data and parsing structures of natural languages. In order to represent structural features

In: Nicolas Lachiche, Christel Vrain (eds.): Late Breaking Papers of ILP 2017, Oéans, France, September 4-6, 2017, published at http://ceur-ws.org

from tree structured data, we propose an ordered term tree pattern [3, 9], which is a tree pattern t such that (i) t has ordered tree structures, and (ii) t has 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 according to the variable label. Computational learning theory is an important research area in inductive logic programming and machine learning. In the computational learning theory, we showed in [9] that the class of languages defined by ordered term tree patterns is polynomial time inductively inferable from positive data. The query learning model is the exact learning model of Angluin [1], which 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.

In this paper, based on FGS, we introduce a primitive formal ordered tree system (pFOTS) as a formal system defining labelled ordered tree languages. A pFOTS program is a finite set of graph rewriting rules that replace variables of ordered term tree patterns with specified ordered trees. A logic program is well-known to be suitable to represent background knowledge. In this paper, we consider a pFOTS program to be background knowledge. We propose a learning method of new graph rewriting rules under background knowledge defined by a pFOTS program. In Fig.1, we give a pFOTS program $\Gamma_{\mathcal{OT}}$, graph rewriting rule α and language $\mathcal{L}(\Gamma_{\mathcal{OT}} \cup \{\alpha\}, r)$ defined using the pFOTS program $\Gamma_{\mathcal{OT}} \cup \{\alpha\}$ and predicate symbol r as examples. In this paper, we propose leaning methods which find the target graph rewriting rule α by given background knowledge $\Gamma_{\mathcal{OT}}$ and the given target language $\mathcal{L}(\Gamma_{\mathcal{OT}} \cup \{\alpha\}, r)$. Our learning methods use some operators based on the given pFOTS program. Therefore, even if the given pFOTS program and the language are changed, our leaning algorithms will find the target graph rewriting rule.

In this paper, we show that if background knowledge is given by a pFOTS program which has one predicate symbol, then one graph rewriting rule is exactly learnable with one positive example and a polynomial number of membership queries w.r.t. the size of the positive example. Moreover, we show the learnability of one graph rewriting rule under pFOTS program having multiple predicate symbols as background knowledge.

For the learning of ordered tree languages, Suzuki et al. [9] showed that the class of languages defined by ordered term tree patterns is polynomial time inductively inferable from positive data. Matsumoto et al. [4] 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. [5] showed that some classes of graph pattern languages defined by FGS are exactly learnable with a polynomial number of equivalence and restricted subset queries. Shoudai et al. [8] 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 [2].

This paper is organized as follows. In Sections 2, we introduce a primitive formal ordered tree system (pFOTS) as a formal system defining ordered tree languages based on FGS [10]. In Section 3, we introduce the query learning model [1] and consider the exact learnability of one graph rewriting rule under pFOTS program as background knowledge in the query learning model. Moreover, we show the exact learnability of one graph rewriting rule under pFOTS program having multiple predicate symbols as background knowledge. Finally, in Section 4, we conclude this paper and discuss future work.

2 Primitive Formal Ordered Tree Systems (pFOTS)

Let Σ and Λ be finite alphabets whose elements are called *node-labels* and *edge-labels*, respectively. Let $\mathcal{X}$ be an infinite alphabet whose element is called a *variable label*. We assume that $\Lambda \cap \mathcal{X} = \emptyset$. A node- and edge-labelled ordered tree $t = (V_t, E_t)$ over $\langle \Sigma, \Lambda \cup \mathcal{X} \rangle$ is an ordered tree that has a node-labelling function $\psi_t : V_t \to \Sigma$ and an edge-labelling function $\varphi_t : E_t \to \Lambda \cup \mathcal{X}$. An edge labelled with a variable label in $\mathcal{X}$ is called a *variable*. For any $x \in \mathcal{X}$, $o(t, x)$ denotes the number of variables in t labelled with x. A node- and edge-labelled ordered tree t over $\langle \Sigma, \Lambda \cup \mathcal{X} \rangle$ is said to be a *linear term tree pattern* over $\langle \Sigma, \Lambda \cup \mathcal{X} \rangle$ if, for any $x \in \mathcal{X}$, $o(t, x) \leq 1$ holds. For a node $v \in V_t$, except the root of t, and parent $u \in V_t$ of v, the edge e between u and v is denoted as (u, v) if $\varphi_t(e) \in \Lambda$ and as $\langle u, v \rangle$ if $\varphi_t(e) \in \mathcal{X}$, respectively. Hereafter, an edge of t means an edge whose label is in Λ. A linear term tree pattern over $\langle \Sigma, \Lambda \cup \mathcal{X} \rangle$ with no variable is a node- and edge-labelled ordered tree over $\langle \Sigma, \Lambda \rangle$ and is simply called a *tree* over $\langle \Sigma, \Lambda \rangle$. The term $\mathcal{OT}$ denotes the set of all trees over $\langle \Sigma, \Lambda \rangle$. Hereafter, a linear term tree pattern over $\langle \Sigma, \Lambda \cup \mathcal{X} \rangle$ is simply called a *term tree pattern*. A term tree pattern is said to be *primitive* if it consists of two nodes and one variable between them.

Let f and g be term tree patterns with at least two nodes. Let $\sigma = [u, v]$ be a pair of the root u and a leaf v of g and x a variable label in $\mathcal{X}$. The form $x := [g, \sigma]$ is called a *binding* over $\langle \Sigma, \Lambda \cup \mathcal{X} \rangle$. A new term tree pattern,

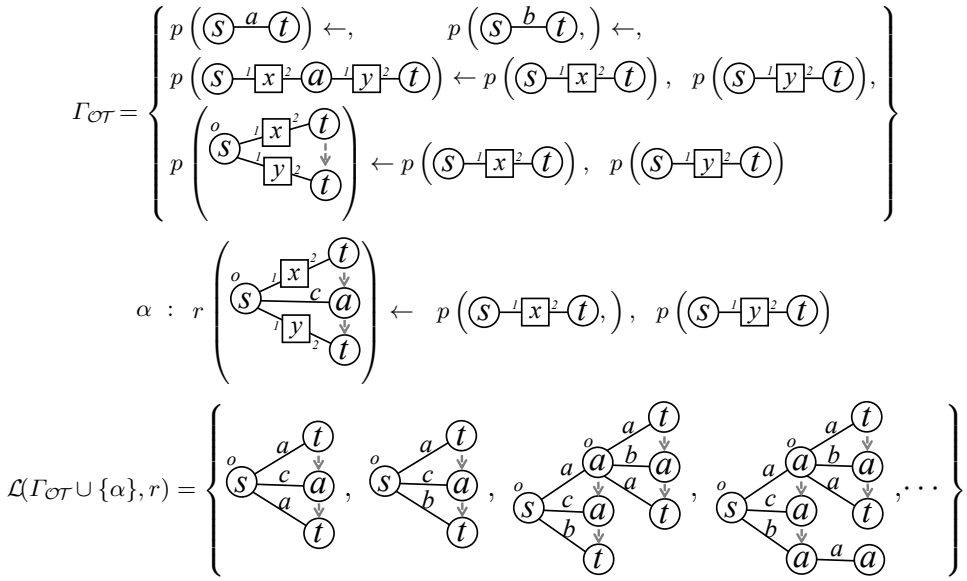

Fig. 1. A pFOTS program $\Gamma_{\mathcal{OT}}$, a graph rewriting rule α, and the labelled ordered tree language $\mathcal{L}(\Gamma_{\mathcal{OT}} \cup \{\alpha\}, r)$ defined using pFOTS program $\Gamma_{\mathcal{OT}} \cup \{\alpha\}$ and predicate symbol r. The symbol o over internal nodes indicates that nodes have children in order shown with broken arrows. Variables denoted with squares each of which connects to two nodes ordered by numbers on lines.

denoted as $f\{x := [g, \sigma]\}$, is obtained by applying the binding $x := [g, \sigma]$ to f as follows. Let $e = \langle s, t \rangle$ be a variable in f with the variable label x, i.e., $\varphi_f(e) = x$. Let g' be a copy of g, and u' and v' be the nodes of g' corresponding to u and v of g, respectively. For $e = \langle s, t \rangle$, we attach g' to f by removing e from f, then identifying s with u' and t with v'. For all internal nodes of $f\{x := [g, \sigma]\}$ to have the ordered children, the children of s need to be reordered (refer to [9]). A *substitution* θ is a finite set of bindings $\{x_1 := [g_1, \sigma_1],\ x_2 := [g_2, \sigma_2], \ldots,\ x_n := [g_n, \sigma_n]\}$, where x_is are mutually distinct variable labels in $\mathcal{X}$. For a term tree pattern f and a substitution θ, we denote the term tree pattern obtained from f and θ denoted as $f\theta$ by applying all bindings in θ to f simultaneously. Let Π be a set of unary predicate symbols. We assume that each predicate symbol $p \in \Pi$ is assigned a pair of two distinct two symbols in Σ. The pair is denoted as $pointer(p)$. Let p be a unary predicate symbol in Π and t a term tree pattern. An *atom* is an expression of the form $p(t)$. Let $A, B_1, B_2, \ldots, B_n$ be atoms, where $n \geq 0$. A *graph rewriting rule* (*rule*) over $\langle \Pi, \Sigma, \Lambda \cup \mathcal{X} \rangle$ is a clause of the form $A \leftarrow B_1, B_2, \ldots, B_n$. Atom A is called the *head* and the right part $B_1, B_2, \ldots, B_n$ is called the *body* of the rule. If $n = 0$, the rule is called a *fact*. A rule $p(t) \leftarrow q_1(f_1),\ q_2(f_2), \ldots,\ q_n(f_n)$ is said to be *primitive* if the following conditions (1)–(3) hold: (1) if $n \geq 1$, f_i are primitive term tree patterns over $\langle \{a \mid a \in pointer(q_i)\}, \mathcal{X} \rangle$ for all i $(1 \leq i \leq n)$, (2) if $n = 0$, t is a tree over $\langle \{a \mid a \in pointer(p)\}, \Lambda \rangle$ consisting of two nodes and the edge between them; otherwise t is a term tree pattern such that each symbol in $pointer(p)$ appears in t, and (3) for any variable $x \in \mathcal{X}$, $o(t, x) = 1$ if and only if $o(f_1, x) + o(f_2, x) + \cdots + o(f_n, x) = 1$. For example, the rule α in Fig. 1 is primitive. The term $p\mathcal{GRR}$ denotes the set of all primitive rules over $\langle \Pi, \Sigma, \Lambda \cup \mathcal{X} \rangle$. For a *rule* $\alpha =$'$p(t) \leftarrow q_1(f_1),\ q_2(f_2), \ldots,\ q_n(f_n)$', let $\Pi^h(\alpha) = \{p\}$ and $\Pi^b(\alpha) = \{q_1, q_2, \ldots, q_n\}$. A finite set Γ of primitive rules is called a *primitive formal ordered tree system program* (*pFOTS program*) if Γ has only one predicate symbol in Π. We give a pFOTS program $\Gamma_{\mathcal{OT}} \cup \{\alpha\}$ in Fig. 1 as an example. For an atom $p(g)$, a rule $A \leftarrow B_1, \ldots, B_n$ and a substitution θ, we define $p(g)\theta = p(g\theta)$ and $(A \leftarrow B_1, \ldots, B_n)\theta = A\theta \leftarrow B_1\theta, \ldots, B_n\theta$. Let Γ be a pFOTS program. The relation $\Gamma \vdash C$ for a rule C is inductively defined as follows. (1) If $C \in \Gamma$, then $\Gamma \vdash C$. (2) If $\Gamma \vdash C$, then $\Gamma \vdash C\theta$ for any substitution θ. (3) If $\Gamma \vdash A \leftarrow B_1, \ldots, B_i, \ldots, B_n$ and $\Gamma \vdash B_i \leftarrow C_1, \ldots, C_m$, then $\Gamma \vdash A \leftarrow B_1, \ \ldots, B_{i-1}, C_1, \ldots, C_m, B_{i+1}, \ldots, B_n$. For a pFOTS program Γ and its predicate symbol p in Π, $\mathcal{L}(\Gamma, p)$ denotes the subset $\{g \in \mathcal{OT} \mid \Gamma \vdash p(g) \leftarrow\}$ of $\mathcal{OT}$. We say that a subset $L \subseteq \mathcal{OT}$ is a *pFOTS language* if there exists a pFOTS program Γ and its predicate symbol p such that $L = \mathcal{L}(\Gamma, p)$ holds. In Fig. 1, we give the pFOTS language $\mathcal{L}(\Gamma_{\mathcal{OT}} \cup \{\alpha\}, r)$ defined by pFOTS program $\Gamma_{\mathcal{OT}} \cup \{\alpha\}$.

Algorithm 1 *LEARNING_pFOTS*

Input: A tree $t \in \mathcal{OT}$.
Output: A rule $\alpha \in p\mathcal{GRR}(\Gamma, r)$ such that $\mathcal{L}(\Gamma \cup \{\alpha\}, r) = \mathcal{L}(\Gamma \cup \{\alpha_*\}, q)$.
Assumption: A pFOTS Γ is known as BK. Predicate symbols r and q are in $\Pi \setminus \Pi(\Gamma)$ and the target α_* is in $p\mathcal{GRR}(\Gamma, q)$.

repeat
  for all $f \in \{t' \in \mathcal{OT} \mid t \rhd_\Gamma t'\}$ **do**
    if MQ$(f) =$ `yes` **then** $t := f$; **break**; **end if**
  end for
until t does not change;
$f := t$; $n := 0$;
for all edges e_f in f **do**
  Let μ be an edge label that is different from the edge label of e_f;
  Let f' be a tree obtained from f by replacing e_f with an edge labeled with μ;
  if MQ$(f') =$ `yes` **then**
    $n := n + 1$;
    Replace the edge e_t of t that corresponds to e_f with a new variable h_n;
  end if
end for
output $\alpha := r(t) \leftarrow p(h_1), p(h_2), \ldots, p(h_n)$;
// We regard variable h_i as a primitive term tree pattern consisting of h_i $(i = 1, \ldots, n)$.

3 Learning of pFOTS via Queries

For a pFOTS program Γ, $\Pi^h(\Gamma) = \bigcup_{\alpha \in \Gamma} \Pi^h(\alpha)$, $\Pi^b(\Gamma) = \bigcup_{\alpha \in \Gamma} \Pi^b(\alpha)$ and $\Pi(\Gamma) = \Pi^h(\Gamma) \cup \Pi^b(\Gamma)$. For a predicate symbol $r \in \Pi \setminus \Pi(\Gamma)$, let $p\mathcal{GRR}(\Gamma, r)$ be the set of all primitive rules α such that $\Pi^h(\alpha) = \{r\}$ and $\Pi^b(\alpha) \subseteq \Pi^h(\Gamma)$. We assume that our learning algorithm knows a fixed pFOTS program Γ, which is referred to as BK, in advance as background knowledge. Then, we consider the learnability of the class $\{\mathcal{L}(\Gamma \cup \{\alpha\}, r) \mid \alpha \in p\mathcal{GRR}(\Gamma, r)\}$ via queries, where $r \in \Pi \setminus \Pi(\Gamma)$. The rule α_* denotes the rule in $p\mathcal{GRR}(\Gamma, q)$ to be identified, which is called the *target*, where $q \in \Pi \setminus \Pi(\Gamma)$. Any tree in $\mathcal{L}(\Gamma \cup \{\alpha_*\}, q)$ is said to be a *positive example*. In the query learning model [1], a learning algorithm can access *oracles* that will answer queries about the target α_*. We consider the following three queries (let $r, q \in \Pi \setminus \Pi(\Gamma)$). **Membership query**(**MQ**): The input is a tree $t \in \mathcal{OT}$. The output is `yes` if $t \in \mathcal{L}(\Gamma \cup \{\alpha_*\}, r)$; otherwise, `no`. **Restricted subset query**(**rSQ**): The input is a rule $\alpha \in p\mathcal{GRR}(\Gamma, r)$. The output of a restricted subset query is `yes` if $\mathcal{L}(\Gamma \cup \{\alpha\}, r) \subseteq \mathcal{L}(\Gamma \cup \{\alpha_*\}, q)$; otherwise, `no`. **Equivalence query**(**EQ**): The input is a rule α in $p\mathcal{GRR}(\Gamma, r)$. The output of an equivalence query is `yes` if $\mathcal{L}(\Gamma \cup \{\alpha\}, r) = \mathcal{L}(\Gamma \cup \{\alpha_*\}, q)$: otherwise, a tree, called a *counterexample*, in $(\mathcal{L}(\Gamma \cup \{\alpha\}, r) \cup \mathcal{L}(\Gamma \cup \{\alpha_*\}, q)) \setminus (\mathcal{L}(\Gamma \cup \{\alpha\}, r) \cap \mathcal{L}(\Gamma \cup \{\alpha_*\}, q))$. A learning algorithm A is said to *exactly identify the target* $\alpha_* \in p\mathcal{GRR}(\Gamma, q)$ if A outputs a rule $\alpha \in p\mathcal{GRR}(\Gamma, r)$ such that $r \in \Pi \setminus \Pi(\Gamma)$ and $\mathcal{L}(\Gamma \cup \{\alpha\}, r) = \mathcal{L}(\Gamma \cup \{\alpha_*\}, q)$. In the case that a target is a finite subset of $p\mathcal{GRR}(\Gamma, r)$, the above queries and exact identifications are defined in a similar way.

Let Γ be a pFOTS program with $\Pi(\Gamma) = \{p\}$ and $\mathcal{F}(\Gamma)$ denote the set of all facts in Γ. For a predicate symbol $r \in \Pi \setminus \{p\}$ and a rule $\alpha =$ '$p(s) \leftarrow p(s_1), \ldots, p(s_n)$' $\in \Gamma$, let $Tr^p_r(\alpha) =$ '$r(s) \leftarrow p(s_1), \ldots, p(s_n)$'. Let f, g be trees and $e = (u, v)$ an edge in f with $\varphi_f(e) \in \Lambda$. Let f' be the term tree pattern obtained from f by changing the edge label $\varphi_f(e)$ of e to a variable label $x \in \mathcal{X}$, i.e., $\varphi_{f'}(e) = x$. Then, we write $g \rhd^e_\Gamma f$ if there exists a substitution $\theta = \{x := [t, \sigma_t]\}$ such that the following three conditions hold: (1) $f'\theta$ is isomorphic to g, (2) t is in $\bigcup_{\alpha \in W} \mathcal{L}(\mathcal{F}(\Gamma) \cup \{Tr^p_r(\alpha)\}, r)$, where $W = \Gamma \setminus \mathcal{F}(\Gamma)$, and (3) σ_t is the pair (u, v) of two nodes u, v of t with $(\psi_t(u), \psi_t(v)) = pointer(p)$. Moreover, we write $g \rhd_\Gamma f$ if there exists an edge e in f such that $g \rhd^e_\Gamma f$ holds. By using Algorithm 1, denoted as *LEARNING_pFOTS*, we have the following theorem.

Theorem 1. *Let $|\Lambda| \geq 2$,Γ be a fixed pFOTS program that is background knowledge with $|\Pi(\Gamma)| = 1$, and α_* a target primitive rule in $p\mathcal{GRR}(\Gamma, r)$, where r is a predicate symbol in $\Pi \setminus \Pi(\Gamma)$. A primitive rule $\alpha \in p\mathcal{GRR}(\Gamma, r')$ satisfying $L(\Gamma \cup \{\alpha\}, r') = L(\Gamma \cup \{\alpha_*\}, r)$ is exactly identified using $O(n^2)$ membership queries and one positive example $T \in L(\Gamma \cup \{\alpha_*\}, r)$, where n is the number of nodes in T.*

A pFOTS program Γ satisfies *PSI condition* 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 $a\}$. Given pFOTS program $\Gamma_1, \ldots, \Gamma_K$ $(K \geq 1)$ satisfies PSI condition as background knowledge, we have the following corollary from Theorem 1.

Table 1. Summary on learnability of pFOTSs

	BK: $\Gamma_{\mathcal{OT}}$, Target: rule α	BK: Γ, Target: rule α	
		$\lvert\Pi(\Gamma)\rvert = 1$	$\lvert\Pi(\Gamma)\rvert \geq 2$
$\lvert\Lambda\rvert = 1$	PII [9]	Open	
$2 \leq \lvert\Lambda\rvert < \infty$		OP & MQ [This paper; Th. 1]	OP & MQ if Cond. PSI holds [This paper; Cor. 1]
$\lvert\Lambda\rvert = \infty$			

	BK: $\Gamma_1 \cup \cdots \cup \Gamma_K$, Target: rules $\alpha_1, \ldots, \alpha_N$, where $\lvert\Pi(\Gamma_k)\rvert = 1$ $(1 \leq k \leq K)$ and $\Pi(\Gamma_i) \cap \Pi(\Gamma_j) = \emptyset$ $(1 \leq i < j \leq K)$		BK: none Target: Γ
	$K > 1, N = 1$	$K = 1, N > 1$	
$\lvert\Lambda\rvert = 1$	OP & MQ if Cond. PSI holds [This paper; Cor. 2]	Open	ILPU & MQ if Cond. 1 holds [8]
$2 \leq \lvert\Lambda\rvert < \infty$			
$\lvert\Lambda\rvert = \infty$		EQ & rSQ [4, 5]	

Corollary 1. *Let $\Gamma_1, \ldots, \Gamma_K$ be fixed pFOTS programs such that $\bigcup_{1 \leq j \leq K} \Gamma_j$ satisfies PSI condition, $|\Lambda| \geq 2$, $|\Pi(\Gamma_i)| = 1$ for any i $(1 \leq i \leq K)$ and $\Pi(\Gamma_i) \cap \Pi(\Gamma_j) = \emptyset$ for any i, j $(1 \leq i < j \leq K)$. Let α_* be a target primitive rule in $p\mathcal{GRR}(\Gamma_1 \cup \cdots \cup \Gamma_K, r)$, where r is a predicate symbol in $\Pi \setminus \bigcup_{1 \leq i \leq K} \Pi(\Gamma_i)$. A primitive rule $\alpha \in p\mathcal{GRR}(\Gamma_1 \cup \cdots \cup \Gamma_K, r')$ satisfying $L(\Gamma_1 \cup \cdots \cup \Gamma_K \cup \{\alpha\}, r') = L(\Gamma_1 \cup \cdots \cup \Gamma_K \cup \{\alpha_*\}, r)$ is exactly identified using one positive example $T \in L(\Gamma_1 \cup \cdots \cup \Gamma_K \cup \{\alpha_*\}, r)$ and $O(n^2)$ membership queries, where n is the number of nodes in T.*

Next, given a pFOTS program having multiple predicate symbols and satisfying PSI condition as background knowledge, from Theorem 1, we have the following corollary.

Corollary 2. *Let Γ be a fixed pFOTS program such that Γ satisfies PSI condition, $|\Lambda| \geq 2$ and $|\Pi(\Gamma)| \geq 2$. Let α_* be a target primitive rule in $p\mathcal{GRR}(\Gamma, r)$ with $r \in \Pi \setminus \Pi(\Gamma)$. A primitive rule α in $p\mathcal{GRR}(\Gamma, r')$ satisfying $L(\Gamma \cup \{\alpha\}, r') = L(\Gamma \cup \{\alpha_*\}, r)$ is exactly identified using one positive example $T \in L(\Gamma \cup \{\alpha_*\}, r)$ and $O(n^2)$ membership queries, where n is the number of nodes in T.*

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

We summarize related work and the results discussed in this paper in Table 1. The terms PII and ILPU stand for the learning models "polynomial-time inductive inference from positive data" and "identification in the limit by polynomial-time update from positive data," respectively. The term OP means that the learning algorithm in the result requires exactly one positive example. We say that a pFOTS program Γ satisfies the Condition 1 if Γ has the 1-finite context property with Chomsky normal form and bounded treewidth property [8].

4 Conclusions

We have introduced a primitive formal ordered tree system (pFOTS) as a formal system defining the ordered tree languages based on FGS [10]. For a fixed pFOTS program Γ as background knowledge, we have shown the exact learnability of tree languages defined using pFOTS programs $\Gamma \cup \{\alpha_*\}$ such that α_* is a target primitive graph rewriting rule by using one positive example and a polynomial number of membership queries. For future work, for a fixed pFOTS program as background knowledge that has only one edge label, we will consider the exact learnability of tree languages defined by pFOTS programs using one positive example and membership queries, and tree languages defined by pFOTS programs without background knowledge using other queries such as equivalence queries, restricted subset queries, and predicate membership queries (see [1, 7]).

Acknowledgments

We would like to thank the anonymous referees for their helpful comments. 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. Angluin, D.: Queries and concept learning. Machine Learning 2(4), 319–342 (1988)
2. Clark, A., Yoshinaka, R.: Distributional learning of context-free and multiple context-free grammars. In: Topics in Grammatical Inference, pp. 143–172. Springer-Verlag Berlin Heidelberg (2016)
3. Matsumoto, S., Hayashi, Y., Shoudai, T.: Polynomial time inductive inference of regular term tree languages from positive data. Proc. ALT-97, Springer-Verlag, LNAI 1316 pp. 212–227 (1997)
4. Matsumoto, S., Shoudai, T., Uchida, T., Miyahara, T., Suzuki, Y.: Learning of finite unions of tree patterns with internal structured variables from queries. IEICE Trans. Inf. & Syst. E91-D(2), 222–230 (2008)
5. Okada, R., Matsumoto, S., Uchida, T., Suzuki, Y., Shoudai, T.: Exact learning of finite unions of graph patterns from queries. In: Proc. ALT 2007, LNAI 4754. pp. 298–312. Springer (2007)
6. Rozenberg, G. (ed.): Handbook of Graph Grammars and Computing by Graph Transformation: Volume I. Foundations. World Scientific Publishing Co., Inc., River Edge, NJ, USA (1997)
7. Sakamoto, H., Hirata, K., Arimura, H.: Learning elementary formal systems with queries. Theoretical Computer Science 298(1), 21–50 (2003)
8. Shoudai, T., Matsumoto, S., Suzuki, Y.: Distributional learning of regular formal graph system of bounded degree. In: Proc. ILP 2016, LNAI 10326. pp. 68–80. Springer (2016)
9. Suzuki, Y., Shoudai, T., Uchida, T., Miyahara, T.: Ordered term tree languages which are polynomial time inductively inferable from positive data. Theoretical Computer Science 350(1), 63–90 (2006)
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)