数理解析研究所講究録 1325 巻 2003 年 122-127



# Polynomial Time Learnabilities of Tree Patterns with Internal Structured Variables from Queries

松本 哲志 (Satoshi Matsumoto)
東海大学 理学部 情報数理学科
Department of Mathematical Sciences, Tokai University
matumoto@ss.u-tokai.ac.jp

鈴木 祐介 (Yusuke Suzuki), 正代 隆義 (Takayoshi Shoudai)
九州大学大学院 システム情報科学 { 府, 研究院 } 情報理学 { 専攻, 部門 }
Department of Informatics, Kyushu University
{y-suzuki,shoudai}@i.kyushu-u.ac.jp

内田 智之 (Tomoyuki Uchida)†, 宮原 哲浩 (Tetsuhiro Miyahara)††
† 広島市立大学 情報科学部 情報メディア工学科
†† 広島市立大学 情報科学部 知能情報システム工学科
†Department of Computer and Media Technologies, Hiroshima City University,
††Department of Intelligent Systems, Hiroshima City University
{uchida@cs,miyahara@its}.hiroshima-cu.ac.jp

**Abstract**

We give the polynomial time learnabilities of two classes of ordered tree patterns with internal structured variables, in the query learning model of Angluin (1988). An ordered tree pattern with internal structured variables, called a term tree, is a rooted tree pattern which consists of tree structures, ordered children and internal structured variables. A term tree is suited for representing structural features in semistructured or tree structured data such as HTML/XML files. We show the polynomial time learnabilities of two classes of term trees using membership and restricted subset queries and one positive example.

## 1 Introduction

Large amount of Web documents such as HTML/XML files are available. Such documents are called semistructured data and considered tree structured data, which are represented by rooted trees with ordered children and edge labels [1]. As an example of a representation of tree structured data, we give a rooted tree $T$ in Fig. 1. This work is motivated from data mining of tree structured patterns from semistructured data.

As a representation of a tree structured pattern, we use an ordered tree pattern with internal structured variables, called a *term tree*. A term tree is a rooted tree pattern which consists of tree structures, ordered children and internal structured variables. A variable in a term tree is a list of vertices and it can be substituted by an arbitrary tree. A term tree is more powerful than or incomparable to other representations of tree structured patterns, which were proposed in computational learning theory, such as ordered tree patterns [2] and ordered gapped tree patterns [7]. We can show that a term tree is more powerful than an ordered tree pattern, which is also called a first order term in formal logic. Consider the example in Fig. 1. The tree pattern $f(b,x,g(a,z),y)$ can be represented by the term tree $s$, but the term tree $t$ cannot be represented by any ordered tree pattern because of the existence of internal structured variables represented by $x_2$ and $x_3$ in $t$. The variable represented by $x_3$ in $t$ is a list of vertices $[v_6, v_7, v_9]$.

For a set of edge labels $\Lambda$, the *term tree language* $L_\Lambda(t)$ of a term tree $t$ with $\Lambda$, which denotes the representing power of $t$, is the set of all labeled trees which are obtained from $t$ by substituting arbitrary labeled trees for all variables in $t$. The subtrees which are obtained from $t$ by removing the variables in $t$ represent the common subtree structures in the trees in $L_\Lambda(t)$. A term tree $t$ is said to be *regular* if all variable labels in $t$ are mutually distinct.

$\mathcal{OTT}_\Lambda$ denotes the set of all regular term trees with $\Lambda$ as a set of edge labels. Let $t$ be a term tree in $\mathcal{OTT}_\Lambda$. For a variable $h = [u_0, u_1, \dots, u_l]$ in $t$, we define $parent(h) = u_0$ and $child(h) = \{u_1, \dots, u_l\}$. We denote by $z\mathcal{OTT}_\Lambda$ the set of all term trees $t \in \mathcal{OTT}_\Lambda$ with a variable set $H_t$ such that $parent(h_1) \notin child(h_2)$ for any $h_1$ and $h_2$ in $H_t$.

In query learning model, a learning algorithm accesses to oracles, which answer specific kinds of queries, and collect information about a target term tree $t_*$. We consider the following oracles. ***Membership oracle***: The input is a term tree $t$ having no variable. The output is "*yes*" if $t \in L_\Lambda(t_*)$, and "*no*" otherwise. ***Restricted subset oracle***: The input is a term tree $t$ in $\mathcal{OTT}_\Lambda$. The output is "*yes*" if $L_\Lambda(t) \subseteq L_\Lambda(t_*)$, and "*no*" otherwise. The former is called a ***membership query*** and the latter is called a ***restricted subset query***. In this model, a learning algorithm is said to ***exactly identify*** a target term tree $t_*$ if it outputs a term tree $t$ such that $L_\Lambda(t) = L_\Lambda(t_*)$ and halts, after it uses some queries.

In this paper, we assume $|\Lambda| \geq 2$. We show that any term tree in $\mathcal{OTT}_\Lambda$ is exactly identifiable in polynomial time using at most $n^2 + n$ membership queries, at most $n$ restricted subset queries and one positive example, where $n$ is the size of the positive example. Moreover, we show that any term tree in $z\mathcal{OTT}_\Lambda$ is exactly identifiable in polynomial time using at most $n^2 + 2n$ membership queries and one positive example, where $n$ is the size of the positive example.

As our previous works, we showed the learnabilities of graph structured patterns[8], term tress with unordered children [10] in the framework of polynomial time inductive inference from positive data [4]. Also our work [11] showed that the class $\mathcal{OTT}_\Lambda^1$, a subclass of $\mathcal{OTT}_\Lambda$, is polynomial time inductively inferable from positive data. As an application [9], we gave a data mining method from semistructured data by using a learning algorithm for term trees. As other related works, the works [2, 3, 6, 7] showed the learnabilities of tree structured patterns in query learning model. The tree structured patterns and learning models of this work are incomparable to those of all the other related works.

This paper is organized as follows. In Section 2, we explain term trees as tree structured patterns. In Section 3, we explain the query learning model. In Section 4, we show the above learnabilities of the two classes $\mathcal{OTT}_\Lambda$ and $z\mathcal{OTT}_\Lambda$.

## 2 Preliminaries

Let $T = (V_T, E_T)$ be an ordered tree with a vertex set $V_T$ and an edge set $E_T$. A list $h = [u_0, u_1, \dots, u_\ell]$ of vertices in $V_T$ is called a ***variable*** of $T$ if $u_1, \dots, u_\ell$ are consecutive children of $u_0$, i.e., $u_0$ is the parent of $u_1, \dots, u_\ell$ and $u_{j+1}$ is the next sibling of $u_j$ for any $j$ with $1 \leq j < \ell$. We call $u_0$ the ***parent port*** of the variable $h$ and $u_1, \dots, u_\ell$ the ***child ports*** of $h$. Two variables $h = [u_0, u_1, \dots, u_\ell]$ and $h' = [u'_0, u'_1, \dots, u'_{\ell'}]$ are said to be ***disjoint*** if $\{u_1, \dots, u_\ell\} \cap \{u'_1, \dots, u'_{\ell'}\} = \emptyset$. For a set $S$, we denote by $|S|$ the number of elements in $S$.

**Definition 1** Let $T = (V_T, E_T)$ be an ordered tree and $H_T$ a set of pairwise disjoint variables of $T$. An *ordered term tree obtained from $T$ and $H_T$* is a triplet $t = (V_t, E_t, H_t)$, where $V_t = V_T$, $E_t = E_T - \bigcup_{[u_0,u_1,\dots,u_\ell] \in H_T} \{\{u_0, u_i\} \in E_T \mid 1 \leq i \leq \ell\}$ and $H_t = H_T$.

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 $T$. In particular, 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 vertex $u$ in $t$ as the order of the children of $u$ in $T$. We often omit the description of the ordered tree $T$ and variable set $H_T$ because we can find them from the triplet $t = (V_t, E_t, H_t)$. We define the ***size*** of $t$ as the number of vertices in $t$ and denote it by $|t|$.

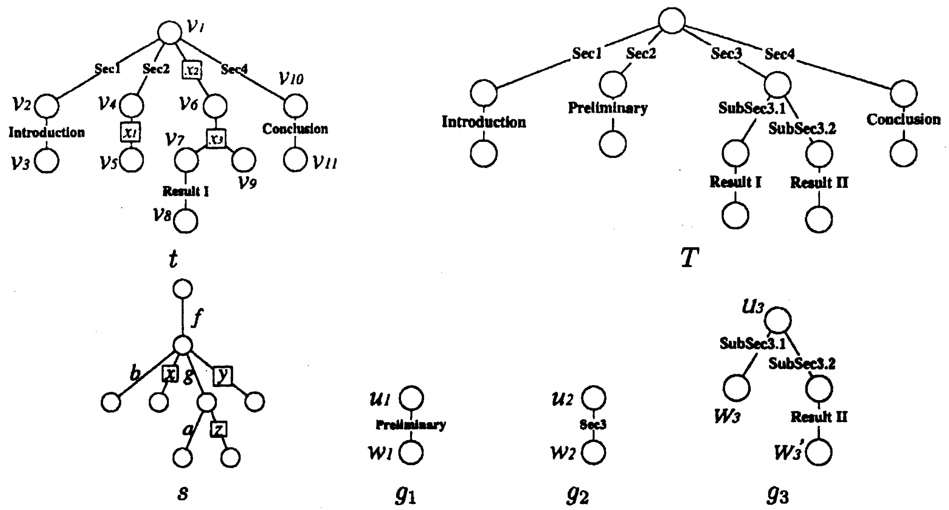


Figure 1: A term tree $t$ explains a tree $T$. A term tree $s$ represents the tree pattern $f(b,x,g(a,z),y)$. A variable is represented by a box with lines to its elements. The label inside a box is the variable label of the variable.

For example, the ordered term tree $t$ in Fig. 1 is obtained from the tree $T=(V_T,E_T)$ and the set of variables $H_T$ defined as follows. $V_T=\{v_1,\dots,v_{11}\}$, $E_T=\{\{v_1,v_2\},\ \{v_2,v_3\},\ \{v_1,v_4\},\ \{v_4,v_5\},\ \{v_1,v_6\},\ \{v_6,v_7\},\ \{v_7,v_8\},\ \{v_6,v_9\},\ \{v_1,v_{10}\},\ \{v_{10},v_{11}\}\}$ with the root $v_1$ and the sibling relation displayed in Fig. 1. $H_T=\{[v_4,v_5],[v_1,v_6],[v_6,v_7,v_9]\}$.

For any ordered term tree $t$, a vertex $u$ of $t$, and two children $u'$ and $u''$ of $u$, we write $u' <^t_u u''$ if $u'$ is smaller than $u''$ in the order of the children of $u$. We assume that every edge and variable of an ordered term tree is labeled with some words from specified languages. A label of a variable is called a *variable label*. $\Lambda$ and X denote a set of edge labels and a set of variable labels, respectively, where $\Lambda\cap X=\phi$. An ordered term tree $t=(V_t,E_t,H_t)$ is called *regular* if all variables in $H_t$ have mutually distinct variable labels in $X$.

**Note.** In this paper, we treat only regular ordered term trees, and then we call a regular ordered term tree a *term tree*, simply. In particular, an ordered term tree with no variable is called a *ground term tree* and considered to be a tree with ordered children.

$\mathcal{OT}_\Lambda$ denotes the set of all ground term trees with $\Lambda$ as a set of edge labels. Let $\mathcal{OTT}_\Lambda$ be the set of all term trees. In particular, for a positive integer $L$, we denote by $\mathcal{OTT}^L_\Lambda$ the set of all term trees $t$ with $\Lambda$ as a set of edge labels such that each variable in $t$ has at most $L$ child ports.

Let $f=(V_f,E_f,H_f)$ and $g=(V_g,E_g,H_g)$ be term trees. We say that $f$ and $g$ are *isomorphic*, denoted by $f\equiv g$, if there is a bijection $\varphi$ from $V_f$ to $V_g$ such that (i) the root of $f$ is mapped to the root of $g$ by $\varphi$, (ii) $\{u,u'\}\in E_f$ if and only if $\{\varphi(u),\varphi(u')\}\in E_g$ and the two edges have the same edge label, (iii) $[u_0,u_1,\dots,u_\ell]\in H_f$ if and only if $[\varphi(u_0),\varphi(u_1),\dots,\varphi(u_\ell)]\in H_g$, and (iv) for any vertex $u$ in $f$ which has more than one child, and for any two children $u'$ and $u''$ of $u$, $u' <^f_u u''$ if and only if $\varphi(u') <^g_{\varphi(u)} \varphi(u'')$. We say that an edge $\{u,u'\}\in E_f$ *corresponds* to an edge $\{v,v'\}\in E_g$ if $v=\varphi(u)$ and $v'=\varphi(u')$.

Let $f$ and $g$ be term trees with at least two vertices. Let $h=[v_0,v_1,\dots,v_\ell]$ be a variable in $f$ with the variable label $x$ and $\sigma=[u_0,u_1,\dots,u_\ell]$ a list of $\ell+1$ distinct vertices in $g$ where $u_0$ is the root of $g$ and $u_1,\dots,u_\ell$ are leaves of $g$. The form $x:=[g,\sigma]$ is called a *binding* for $x$. A new term tree $f'=f\{x:=[g,\sigma]\}$ is obtained by applying the binding $x:=[g,\sigma]$ to $f$ in the following way. For the variable $h=[v_0,v_1,\dots,v_\ell]$, we attach $g$ to $f$ by removing the variable $h$ from $H_f$ and by identifying the vertices $v_0,v_1,\dots,v_\ell$ with the vertices $u_0,u_1,\dots,u_\ell$ of $g$ in this order. We define a new ordering $<^{f'}_v$ on every vertex $v$ in $f'$ in the following natural way. Suppose that $v$ has more than one child and let $v'$ and $v''$ be two children of $v$ in $f'$. We note that $v_i=u_i$ for any $0\le i\le\ell$. (1) If $v,v',v''\in V_g$ and $v' <^g_v v''$, then $v' <^{f'}_v v''$. (2) If $v,v',v''\in V_f$ and $v' <^f_v v''$, then $v' <^{f'}_v v''$. (3) If $v=v_0(=u_0)$,

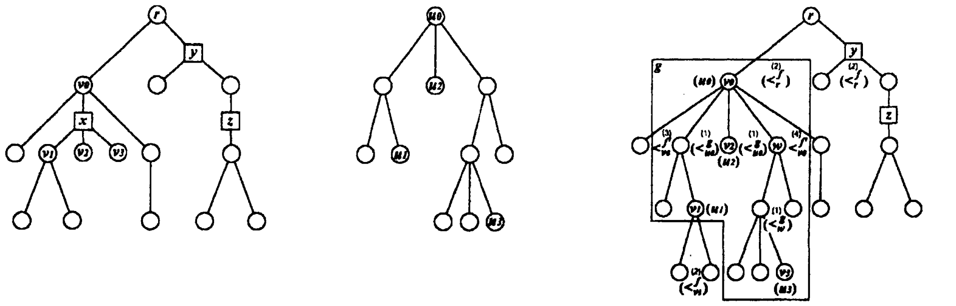


Figure 2: The new ordering on vertices in the term tree $f' = f\{x := [g, [u_0, u_1, u_2, u_3]]\}$.

$v' \in V_f - \{v_1, \ldots, v_\ell\}$, $v'' \in V_g$, and $v' <_v^f v_1$, then $v' <_v^{f'} v''$. (4) If $v = v_0 (= u_0)$, $v' \in V_f - \{v_1, \ldots, v_\ell\}$, $v'' \in V_g$, and $v_\ell <_v^f v'$, then $v'' <_v^{f'} v'$. In Fig. 2, we give an example of the new ordering on vertices in a term tree.

A ***substitution*** $\theta$ is a finite collection of bindings $\{x_1 := [g_1, \sigma_1], \cdots, x_n := [g_n, \sigma_n]\}$, where $x_i$'s are mutually distinct variable labels in $X$. The term tree $f\theta$, called the ***instance*** of $f$ by $\theta$, is obtained by applying all the bindings $x_i := [g_i, \sigma_i]$ on $f$.We define the root of the resulting term tree $f\theta$ as the root of $f$. Consider the examples in Fig. 1. An example of a term tree $t$ is given. Let $\theta = \{x_1 := [g_1, [u_1, w_1]], x_2 := [g_2, [u_2, w_2]], x_3 := [g_3, [u_3, w_3, w_3']]\}$ be a substitution, where $g_1$, $g_2$, and $g_3$ are ground term trees in Fig. 1. Then the instance $t\theta$ of the term tree $t$ by $\theta$ is isomorphic to the tree $T$ in Fig. 1. Let $t$ and $t'$ be term trees. We write $t \preceq t'$ if there exists a substitution $\theta$ such that $t \equiv t'\theta$. If $t \preceq t'$ and $t \not\equiv t'$, then write $t \prec t'$. Let $\Lambda$ be a set of edge labels. The ***term tree language*** $L_\Lambda(t)$ of a term tree $t \in \mathcal{OTT}_\Lambda$ is $\{s \in \mathcal{OT}_\Lambda \mid s \preceq t\}$.

# 3 Learning model

In this paper, let $t_*$ be a term tree in $\mathcal{OTT}_\Lambda$ to be identified, and we say that the term tree $t_*$ is a ***target***. A ground term tree $t$ is called a ***positive example*** of $L_\Lambda(t_*)$ if $t$ is in $L_\Lambda(t_*)$.

We introduce the exact learning model via queries due to Angluin [5]. In this model, learning algorithms can access to ***oracles*** that answer specific kinds of queries about the unknown term tree language $L_\Lambda(t_*)$. We consider the following oracles: (1) ***Membership oracle*** $Mem_{t_*}$: The input is a ground term tree $t$. The output is "***yes***" if $t$ is in $L_\Lambda(t_*)$, and "***no***" otherwise. The query is called a ***membership query***. (2) ***Restricted subset oracle*** $rSub_{t_*}$: The input is a term tree $t$ in $\mathcal{OTT}_\Lambda$. The output is "***yes***" if $L_\Lambda(t) \subseteq L_\Lambda(t_*)$, and "***no***" otherwise. The query is called a ***restricted subset query***.

A learning algorithm $\mathcal{A}$ may collect information about membership and restricted subset queries of $L_\Lambda(t_*)$. We say that a learning algorithm ***exactly identifies*** a target $t_*$ if it outputs a term tree $t$ in $\mathcal{OTT}_\Lambda$ with $L_\Lambda(t) = L_\Lambda(t_*)$ and halts after it uses some queries.

# 4 Learning using membership and restricted subset queries

We introduce an operation of a contraction which reduces the number of edges in a ground term tree. In this paper, we assume $|\Lambda| \geq 2$.

**Definition 2** Let $t = (V_t, E_t, H_t)$ be a ground term tree and $e = \{u, v\}$ an edge in $E_t$. We define the contraction of $e$ to $t$ as the following operation: If $v$ has children $v_1, \ldots, v_l$, then the operation removes $v$ from $V_t$ and replaces $\{u, v\}, \{v, v_1\}, \ldots, \{v, v_l\}$ in $E_t$ with new edges $\{u, v_1\}, \ldots, \{u, v_l\}$, that is, $E_t = E_t \cup \{\{u, v_1\}, \ldots, \{u, v_l\}\} - \{\{u, v\}, \{v, v_1\}, \ldots, \{v, v_l\}\}$. Otherwise, the operation removes $v$ from $V_t$ and $e$ from $E_t$. We denote by $t \backslash \{e\}$ the term tree obtained from $t$ by applying the contraction

**Algorithm** *CONTRACTION*;
*Given*: An oracle $Mem_{t_*}$ for a target $t_*$ in $\mathcal{OTT}_\Lambda$ and a positive example $t$ in $L_\Lambda(t_*)$;
*Output*: A term tree $r$ in $\mathcal{OTT}^1_\Lambda$ with $r \equiv port(t_*)$;
**begin**
  **repeat**
    **foreach** edge $e$ in $t$ **do begin** Let $t' := t \backslash \{e\}$;
      **if** $Mem_{t_*}(t')$ = "*yes*" **then begin** $t := t'$; **break**; **end**; **end**;
  **until** $t$ does not change;
  Let $r = (V_r, E_r, H_r)$ be $t$;
  **foreach** edge $e = \{u, v\}$ in $t$ **do begin**
    Let $t'$ be a term tree obtained from $t$ by replacing the label of $e$ with another label;
    **if** $Mem_{t_*}(t')$ = "*yes*" **then begin**
      Let $\{u', v'\}$ be an edge in $r$ which corresponds to $\{u, v\}$ in $t$.
      $E_r := E_r - \{\{u', v'\}\}$; $H_r := H_r \cup \{[u', v']\}$; **end**;
  **end**; **output** $r$;
**end**

Figure 3: **Algorithm** *CONTRACTION*

**Algorithm** *LEARN_OTT*;
*Given*: Oracles $rSub_{t_*}$ and $Mem_{t_*}$ for a target $t_*$ in $\mathcal{OTT}_\Lambda$ and a positive example $t$ in $L_\Lambda(t_*)$;
*Output*: A term tree $r$ in $\mathcal{OTT}_\Lambda$ with $L_\Lambda(r) = L_\Lambda(t_*)$;
**begin**
  $r := CONTRACTION(t)$ using $Mem_{t_*}$;
  let $H = [h_1, \ldots, h_l]$ be the sequence of variables in $r$ by the breath-first search order;
  $i := 1$; flag:=false;
  **while** $H$ is not empty **do begin** Let $S = \{h_i\}$;
    **repeat**
      Let $h_i = [u_i, v_i]$ and $h_{i+1} = [u_{i+1}, v_{i+1}]$;
      **if** $v_{i+1}$ is the next sibling of $v_i$ **then begin**
        Let $S := S \cup \{h_{i+1}\}$; $r' := replace(r, S)$;
        **if** $rSub_{t_*}(r')$ = "*yes*" **then begin** flag := false; $i := i + 1$; **end**
        **else begin** flag := true; $S := S - \{h_{i+1}\}$; **end**; **end**
      **else** flag := true;
    **until** flag
    $r := replace(r, S)$; remove the variables in $S$ from $H$; $i := i + 1$;
  **end**; **output** $r$;
**end**

Figure 4: **Algorithm** *LEARN_OTT*

Let $t = (V_t, E_t, H_t)$ be a term tree in $\mathcal{OTT}_\Lambda$. We denote by $port(t)$ the term tree obtained from $t$ by replacing any variable $[v_0, v_1, \ldots, v_k]$ with $k$ variables $[v_0, v_1], \ldots, [v_0, v_k]$. Hence, for $t \in \mathcal{OTT}_\Lambda$, $port(t)$ is in $\mathcal{OTT}^1_\Lambda$.

Let $k$ be a positive integer, $t = (V_t, E_t, H_t)$ a term tree in $\mathcal{OTT}_\Lambda$ and $h_1 = [v_0, v_1], \ldots, h_k = [v_0, v_k]$ variables in $H_t$, where $v_{i+1}$ is the next sibling of $v_i$ for each $i = 1, \ldots, k-1$. We denote by $replace(t, \{h_1, \ldots, h_k\})$ the term tree obtained from $t$ by replacing the variables $h_1, \ldots, h_k$ with a variable $h = [v_0, v_1, \ldots, v_k]$. That is, $replace(t, \{h_1, \ldots, h_k\}) = (V_t, E_t, H'_t)$, where $H'_t = H_t \cup \{h\} - \{h_1, \ldots, h_k\}$.

**Theorem 1** *The algorithm LEARN_OTT in Fig. 4 exactly identifies any term tree $t_*$ in $\mathcal{OTT}_\Lambda$ in polynomial time using at most $n^2 + n$ membership queries, at most $n$ restricted subset queries and one positive example $t$ in $L_\Lambda(t_*)$, where $n = |t|$.*

From Theorem 1, we can identify any term tree in $\mathcal{OTT}_\Lambda$ using membership, restricted subset queries and a positive example. Next, we show that any term tree in some subset of $\mathcal{OTT}_\Lambda$ is identifiable using membership queries and a positive example.

Let $t = (V_t, E_t, H_t)$ be a term tree in $\mathcal{OTT}_\Lambda$ and $h = [u_0, u_1, \ldots, u_k]$ a variable in $H_t$. We denote by *parent*$(h)$ the parent node of $h$ and by *child*$(h)$ the set of the child ports of $h$, that is, *parent*$(h) = u_0$ and *child*$(h) = \{u_1, \ldots, u_k\}$. We denote by $z\mathcal{OTT}_\Lambda$ the set of all term trees $t = (V_t, E_t, H_t)$ in $\mathcal{OTT}_\Lambda$ such that *parent*$(h_1) \notin$ *child*$(h_2)$ for any $h_1$ and $h_2$ in $H_t$.

**Theorem 2** *Any term tree $t_*$ in $z\mathcal{OTT}_\Lambda$ is exactly identifiable in polynomial time using at most $n^2 + 2n$ membership queries and one positive example $t$ in $L_\Lambda(t_*)$, where $n = |t|$.*

## 5 Conclusions

We have considered the polynomial time learnabilities of $\mathcal{OTT}_\Lambda$ and $z\mathcal{OTT}_\Lambda$ in the query learning model. In this paper, we assume $|\Lambda| \geq 2$. We have shown that any term tree in $\mathcal{OTT}_\Lambda$ is exactly identifiable using at most $n^2 + n$ membership queries, at most $n$ restricted subset queries and one positive example, where $n$ is the size of the positive example. Moreover, we have shown that any term tree in $z\mathcal{OTT}_\Lambda$ is exactly identifiable using at most $n^2 + 2n$ membership queries and one positive example, where $n$ is the size of the positive example.

Suzuki et al. [11, 12] have shown that the learnabilities of $\mathcal{OTT}^1_\Lambda$ and $\mathcal{OTT}_\Lambda$ in the framework of polynomial time inductive inference from positive data, where $|\Lambda| \geq 1$. Thus, we will study the learnability of finite unions of term trees in $\mathcal{OTT}^1_\Lambda$ in the same framework.

## References


[1] S. Abiteboul, P. Buneman, and D. Suciu. *Data on the Web: From Relations to Semistructured Data and XML.* Morgan Kaufmann, 2000.

[2] T. R. Amoth, P. Cull, and P. Tadepalli. Exact learning of tree patterns from queries and counterexamples. *Proc. COLT-98, ACM Press*, pages 175–186, 1998.

[3] T. R. Amoth, P. Cull, and P. Tadepalli. Exact learning of unordered tree patterns from queries. *Proc. COLT-99, ACM Press*, pages 323–332, 1999.

[4] D. Angluin. Finding pattern common to a set of strings. *Journal of Computer and System Sciences*, 21:46–62, 1980.

[5] D. Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.

[6] H. Arimura, H. Ishizaka, and T. Shinohara. Learning unions of tree patterns using queries. *Proc. ALT-95, Springer-Verlag, LNAI 997*, pages 66–79, 1995.

[7] H. Arimura, H. Sakamoto, and S. Arikawa. Efficient learning of semi-structured data from queries. *Proc. ALT-2001, Springer-Verlag, LNAI 2225*, pages 315–331, 2001.

[8] 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 1316*, pages 212–227, 1997.

[9] T. Miyahara, Y. Suzuki, T. Shoudai, T. Uchida, K. Takahashi, and H. Ueda. Discovery of frequent tag tree patterns in semistructured web documents. *Proc. PAKDD-2002, Springer-Verlag, LNAI 2336*, pages 341–355, 2002.

[10] T. Shoudai, T. Uchida, and T. Miyahara. Polynomial time algorithms for finding unordered tree patterns with internal variables. *Proc. FCT-2001, Springer-Verlag, LNCS 2138*, pages 335–346, 2001.

[11] Y. Suzuki, R. Akanuma, T. Shoudai, T. Miyahara, and T. Uchida. Polynomial time inductive inference of ordered tree patterns with internal structured variables from positive data. *Proc. COLT-2002, Springer-Verlag, LNAI 2375*, pages 169–184, 2002.

[12] Y. Suzuki, T. Shoudai, T. Uchida, and T. Miyahara. Ordered term tree languages which are polynomial time inductively inferable from positive data. *Proc. ALT-2002, Springer-Verlag, LNAI 2533*, pages 188–202, 2002.