LEARNING TREE LANGUAGES FROM TEXT

Henning Fernau[1]

Abstract. We study the problem of learning regular tree languages from text. We show that the framework of function distinguishability, as introduced by the author in [*Theoret. Comput. Sci.* **290** (2003) 1679–1711], can be generalized from the case of string languages towards tree languages. This provides a large source of identifiable classes of regular tree languages. Each of these classes can be characterized in various ways. Moreover, we present a generic inference algorithm with polynomial update time and prove its correctness. In this way, we generalize previous works of Angluin, Sakakibara and ourselves. Moreover, we show that this way all regular tree languages can be approximately identified.

Mathematics Subject Classification. 68Q32, 68Q45.

1. Introduction

Grammatical inference (GI) – also called grammar induction – mostly focussed on learning *string* languages, although there are many practical motivations for studying formally specified sets not being comprised of words, as well:

- Linguists are often interested in the dependencies of different parts of a sentence. This sort of dependencies is likely to be reflected in the derivation *trees* of a context-free grammar which captures the main syntactical features of the language in question. Moreover, the meaning attached to a sentence is in close relation to the tree reflecting its analysis. So, tree languages are quite important to linguists. Applications of GI in this setting are reported, *e.g.*, in [6, 38].
- Derivation trees play an important role when studying the use of GI in connection with programming languages, starting from [12]. The whole area of *programming by examples*, sometimes also called *programming by demonstration*, shows the importance of learning context-free languages,

[1] Fachbereich IV, Abteilung Informatik, Universität Trier, 54286 Trier, Germany;
`fernau@uni-trier.de`

at best annotated by derivation trees. Comprehensive introductions into this fields are [13, 46].

- Recently, tree language inference was proposed as a tool in *inductive logic programming* (ILP) [5], this way (again) underlining the intimate relationship between ILP and GI, as already seen, *e.g.*, in [8].
- The quite natural relationship between tree structures and databases gave rise to several more practic-oriented papers in that area; a starting point for reading may be [14].
- In the flourishing area of developing tools that help deal with XML documents, trees can play their rôle, as well, as indicated by the works of Neven and others [53], also see [10][1]. Up to now, *e.g.*, tools for creating document type descriptions were all based, one way or other, on regular language learners, see [1, 20, 31, 37].
- In [33] Chapters 3.2 and 6.4, applications of tree language inference to *pattern recognition* can be found. We will report on a particular application area in this respect at the end of this paper.
- Finally, trees can be equivalently interpreted as *terms*, so that the study of the inference of tree languages might also contribute to the learning of term algebras, a topic touched in [42, 43, 56].

Hence, the study of the automatic inference of tree languages is well motivated. Besides [29, 41–43], the classes of regular tree languages presented in this paper are the first ones which are characterized by well-defined restrictions on automata, an important issue in GI [36][2]. Another characteristic is that we are dealing with inferring *deterministic* automata. In a certain sense, this line of research therefore contrasts with the research undertaken to learn probabilistic or stochastic tree automata, see [9, 56, 57, 68, 69]. The importance of this area of research can be likewise seen by the fact that a special workshop on context-free grammar learning was organized for the joint ECML/PKDD event in Dubrovnik in 2003[3]. In his bibliographical survey [35], de la Higuera dedicates one section to this topic, also *cf.* [24].

In some sense, the present paper can be seen as a continuation of Sakakibara's work [60] on the inference of 0-reversible tree languages[4] and of ours on the inference of distinguishable string languages [22]: we will explore how to learn – exactly and approximately – function distinguishable regular tree languages from text, a setting introduced by Gold [32], also known as *identification in the limit from*

---

[1]More references can be found on Neven's homepage `http://alpha.uhasselt.be/~fneven/publsn2.html`. We also refer to works of G. Gottlob's group in Vienna, see, *e.g.*, [11].

[2]After the publication of the conference version of this paper [21], López *et al.* published a paper [50] in which they showed that natural generalizations of the k-reversible languages towards tree languages, as introduced by Angluin [3] in the string case, lead to tree language classes that can be efficiently learned from text. Besides minor technical differences, the present work also generalizes that of López *et al.* [50], as already in the string case functional distinguishable languages were designed to generalize k-reversible languages.

[3]see `http://ilk.uvt.nl/~mvzaanen/ECMLPKDD/workshop.html`

[4]Ideas similar to Sakakibara's are contained in [18]. Another early work on tree language inference that is worth mentioning is [28].

positive samples. Both [22] and [60] are based on Angluin's paper [3], which was ground-breaking for the revival of the interest in learning regular languages from text, exhibiting the first non-trivial and interesting classes of regular languages which can be learned in this scenario. This paper therefore aims to lay solid theoretical foundations for works on applications of learning regular tree languages, as there are plenty of them as outlined above.

In a nutshell, a function distinguishable regular tree language is given by a bottom-up deterministic tree automaton such that any occurring backward non-determinism can be resolved by means of an "oracle" called distinguishing function. The 0-reversible tree languages considered by Sakakibara [60] are a special case where the oracle gives no information at all. We will show the usefulness of our approach by exhibiting "tree-counterparts" of terminal distinguishable and k-reversible languages, which are the most "popular" learnable regular language classes according to Gregor [34].

Note that, from a linguistic point of view, the fact that every context-free language is the yield of a 0-reversible tree language is somewhat misleading, since not all possible syntax trees (which reflect the interesting semantical connections) are 0-reversible. Hence, it is important to go beyond 0-reversible tree languages. As we will see, the tree language classes introduced in this paper are a natural extension of the 0-reversible tree languages.

Let us mention that the learnability of context-free languages and likewise that of regular tree languages was also studied under different learning paradigms, especially in Angluin's model of a "minimally adequate teacher" (query learning), see [4, 5, 7, 16, 17, 40, 52, 59] and when providing both positive and negative information [30].

The paper is organized as follows. In Section 2, the basic definitions of tree languages which we use are summarized. Nonetheless, we assume some basic knowledge and acquaintance with basic notions in formal language theory and grammatical inference as provided, *e.g.*, within the *Handbook of Formal Languages*, in particular in the third volume [58]. Section 3 explains the central notion of this paper, that of function distinguishability in the context of tree languages. In Section 4, several characterizations of the classes of function distinguishable tree languages are derived. Section 5 collects and proves the learnability results of this paper. In Section 6, the corresponding learning algorithms are explained. Section 7 on approximability notably not only extends but supplements Sakakibara's work. The final section explains some further research directions.

2. Definitions

Let $\mathbb{N}$ be the set of nonnegative integers and let $(\mathbb{N}^*, \cdot, \lambda)$ (or simply $\mathbb{N}^*$) be the free monoid generated by $\mathbb{N}$. For $y, x \in \mathbb{N}^*$, write $y \leq x$ iff there is a $z \in \mathbb{N}^*$ with $x = y \cdot z$; "$y < x$" abbreviates: $y \leq x$ and $y \neq x$. $|x|$ denotes the length of x. We are now giving the necessary definitions for trees and tree automata. More details can be found, *e.g.*, in the chapter written by Gécseg and Steinby in [58].

Trees. A *ranked alphabet* V is a finite set of symbols together with a finite *rank relation* $r_V \subset V \times \mathbb{N}$. Define $V_n := \{f \in V \mid (f, n) \in r_V\}$. Since elements in V_n are often considered as *function symbols* (standing for functions of *arity* n), elements in V_0 are also called *constant symbols*. A *tree over* V is a mapping $t : \Delta_t \to V$, where the domain Δ_t is a finite subset of $\mathbb{N}^*$ such that (1) if $x \in \Delta_t$ and $y < x$, then $y \in \Delta_t$; (2) if $y \cdot i \in \Delta_t$, $i \in \mathbb{N}$, then $y \cdot j \in \Delta_t$ for $1 \le j \le i$. An element of Δ_t is also called a *node* of t, where the node λ is the *root* of the tree. Then $t(x) \in V_n$ whenever, for $i \in \mathbb{N}$, $x \cdot i \in \Delta_t$ iff $1 \le i \le n$. If $t(x) = A$, A is the *label* of x. Let V^{t} denote the set of all finite trees over V. By this definition, trees are rooted, directed, acyclic graphs in which every node except the root has one predecessor and the direct successors of any node are linearly ordered from left to right. Interpreting V as a set of function symbols, V^{t} can be identified with the well-formed terms over V. This yields a compact string denotation of trees. A *frontier node* in t is a node $y \in \Delta_t$ such there is no $x \in \Delta_t$ with $y < x$. If $y \in \Delta_t$ is not a frontier node, it is called *interior node*. The *depth* of a tree t is defined as $\mathrm{depth}(t) := \max\{|x| \mid x \in \Delta_t\}$, whereas the *size* of t is given by $|\Delta_t|$. Letters (or constant symbols) will be viewed as trees of size one and depth zero.

Operations on trees. We are now going to define a catenation on trees. Let $\$$ be a new symbol, *i.e.*, $\$ \notin V$, of rank 0. Let $V_{\$}^{\mathsf{t}}$ denote the set of all trees over $V \cup \{\$\}$ which contain exactly one occurrence of label $\$$. By definition, only frontier nodes can carry the label $\$$. For trees $u \in V_{\$}^{\mathsf{t}}$ and $t \in (V^{\mathsf{t}} \cup V_{\$}^{\mathsf{t}})$, we define an operation $\#$ to replace the frontier node labelled with $\$$ of u by t according to

$$u \# t(x) := \begin{cases} u(x), & \text{if } x \in \Delta_u \wedge u(x) \ne \$, \\ t(y), & \text{if } x = z \cdot y \wedge u(z) = \$ \wedge y \in \Delta_t. \end{cases}$$

The set of nodes can then be described as

$$\Delta_{u \# t} = \Delta_u \cup \{x \cdot y \mid u(x) = \$, y \in \Delta_t\}.$$

If $U \subseteq V_{\$}^{\mathsf{t}}$ and $T \subseteq (V^{\mathsf{t}} \cup V_{\$}^{\mathsf{t}})$, then $U \# T := \{u \# t \mid u \in U \wedge t \in T\}$. For $t \in V^{\mathsf{t}}$ and $x \in \Delta_t$, the *subtree* of t at x, denoted by t/x, is defined by $t/x(y) := t(x \cdot y)$ for any $y \in \Delta_{t/x}$, where $\Delta_{t/x} := \{y \mid x \cdot y \in \Delta_t\}$. $\mathrm{ST}(T) := \{t/x \mid t \in T \wedge x \in \Delta_t\}$ is the set of subtrees of trees from $T \subseteq V^{\mathsf{t}}$. Furthermore, for any $t \in V^{\mathsf{t}}$ and any tree language $T \subseteq V^{\mathsf{t}}$, the *quotient* of T and t is defined as:

$$U_T(t) := \begin{cases} \{u \in V_{\$}^{\mathsf{t}} \mid u \# t \in T\}, & \text{if } t \in V^{\mathsf{t}} \setminus V_0, \\ \{t\}, & \text{if } t \in V_0. \end{cases}$$

Tree automata. Let V be a ranked alphabet and m be the maximum rank of the symbols in V. A *(bottom-up) tree automaton* over V is a quadruple $A = (Q, V, \delta, F)$ such that Q is a finite state alphabet (disjoint with V_0), $F \subseteq Q$ is a set of final states, and $\delta = (\delta_0, \ldots, \delta_m)$ is an $m + 1$-tuple of state transition functions, where $\delta_0(a) = \{a\}$ for $a \in V_0$ and $\delta_k : V_k \times (Q \cup V_0)^k \to 2^Q$ for $k = 1, \ldots, m$. In this definition, the constant symbols at the frontier nodes are taken as sort of initial states. Now, a transition relation (also denoted by δ) can be recursively defined

on V^{t} by letting

$$\delta(f(t_1,\ldots,t_k)) := \begin{cases} \delta_0(f), & \text{if } k = 0, \\ \bigcup_{q_i \in \delta(t_i), i=1,\ldots,k} \delta_k(f, q_1, \ldots, q_k), & \text{if } k > 0. \end{cases}$$

A tree t is accepted by A iff $\delta(t) \cap F \neq \emptyset$. The tree language accepted by A is denoted by $T(A)$. A is *deterministic* if each of the functions δ_k maps each possible argument to a set of cardinality of at most one. Deterministic tree automata can be viewed as algorithms for labelling the nodes of a tree with states. Analogously to the string case, it can be shown that nondeterministic and deterministic finite tree automata accept the same class of tree languages, namely the *regular tree languages*, at the expense of a possibly exponential state explosion.

Remark 2.1. Observe that there is some sort of arbitrariness in having $Q \cap V_0 = \emptyset$. Other papers on tree languages even enforce $V_0 \subseteq Q$.

This tiny technical difference has a certain influence on the subsequent definitions, where especially one-node trees attached with symbols from V_0 are not considered as valid subtrees in the construction of the canonical and of the base tree automata. This of course also influences the learnable classes. Our approach (adapted from Sakakibara's definition [60]) is more appropriate when thinking about derivation trees of context-free grammars as being the most important application of trees, since one-node trees labelled with terminal symbols are no valid derivation trees. Fortunately, all major constructions presented in this paper would work with the indicated different definition(s) of tree automata, as well.

The notions of *isomorphic* automata and (state subset induced) *subautomata* can be easily carried over from the well-known string case to the tree case. A state q of a deterministic tree automaton A is *useful* iff there exists a tree t and some node $x \in \Delta_t$ such that $\delta(t/x) = q$ and $\delta(t) \in F$. A deterministic automaton containing only useful states is called *stripped*.

Constructions for tree automata. We need four special constructions of tree automata in our exposition.

Base tree automaton: firstly, we define the analogue of the well-known prefix-tree acceptor in the string case: let I_+ be a finite tree language over V. The *base tree automaton* for I_+, denoted by $Bs(I_+) = (Q, V, \delta, F)$, is defined as follows: $Q = \mathrm{ST}(I_+) \setminus V_0$, $F = I_+$,

$$\delta_k(f, u_1, \ldots, u_k) := f(u_1, \ldots, u_k)$$

for $u_1, \ldots, u_k \in Q \cup V_0$, whenever $f(u_1, \ldots, u_k) \in Q$.
Obviously, $T(Bs(I_+)) = I_+$.

Canonical tree automaton: secondly, we transfer the notion of canonical automaton to the tree case: Let T be a regular tree language over V. The *canonical tree automaton* for T, denoted by $C(T) = (Q, V, \delta, F)$, is defined by: $Q = \{U_T(s) \mid s \in \mathrm{ST}(T) \setminus V_0\}$, $F = \{U_T(t) \mid t \in T\}$, $\delta_k(f, U_T(s_1), \ldots, U_T(s_k)) := U_T(f(s_1, \ldots, s_k))$ if $f(s_1, \ldots, s_k)$ is in $\mathrm{ST}(T)$.

Observe that $C(T)$ is a deterministic stripped automaton which is formed completely analogously to the minimal deterministic string automaton.

Product automaton: as in the string case, the notion of a product automaton can be defined. If $A = (Q, V, \delta, F)$ and $A' = (Q', V, \delta', F')$ are deterministic tree automata, then $A \times A' = (Q \times Q', V, \bar{\delta}, F \times F')$ is the deterministic *product tree automaton* of A and A', where $\bar{\delta}$ is defined by $\bar{\delta}_0(a) = a$ for $a \in V_0$ and

$$\bar{\delta}_k(f, p_1, \ldots, p_k) := (\delta_k(f, q_1, \ldots, q_k), \delta'_k(f, q'_1, \ldots, q'_k))$$

with $f \in V_k$, $q_1, \ldots, q_k \in Q \cup V_0$ and $q'_1, \ldots, q'_k \in Q' \cup V_0$; if $p_i \in V_0$, we have $p_i = q_i = q'_i$, and otherwise, *i.e.*, if $p_i \in Q \times Q'$, we have $p_i = (q_i, q'_i)$.

Quotient automaton: finally, we define quotient automata. A *partition* of a set S is a collection of pairwise disjoint nonempty subsets of S whose union is S. If π is a partition of S, then, for any element $s \in S$, there is a unique element of π containing s, which we denote $B(s, \pi)$ and call the *block* of π containing s. A partition π is said to *refine* another partition π' iff every block of π' is a union of blocks of π. If π is any partition of the state set Q of the automaton $A = (Q, V, \delta, F)$, then the *quotient automaton* $\pi^{-1}A = (\pi^{-1}Q, V, \delta', \pi^{-1}F)$ is given by $\pi^{-1}\hat{Q} = \{ B(q, \pi) \mid q \in \hat{Q} \}$ (for $\hat{Q} \subseteq Q$) and, for $B_1, \ldots, B_k \in \pi^{-1}Q \cup V_0$, $f \in V_k$, $B \in \delta'_k(f, B_1, \ldots, B_k)$ whenever there exist $q \in B$ and $q_i \in B_i \in \pi^{-1}Q$ or $q_i = B_i \in V_0$ for $1 \leq i \leq k$ such that $q \in \delta_k(f, q_1, \ldots, q_k)$.

3. FUNCTION DISTINGUISHABILITY

The main feature of the automata classes which are learnable from text seems to be some sort of "backward determinism" or "reversibility". In the case of string languages, the corresponding notion of reversible languages due to Angluin was generalized in [22] with the aid of distinguishing functions. We will take a similar venue here for the case of tree languages in order to generalize the learnability results of Sakakibara for reversible tree languages. More precisely, we will add a so-called distinguishing function as a kind of "oracle" to make a decision when backward nondeterminism conflicts arise.

Definition 3.1. Let $A_\delta = (Q_\delta, V, \delta, Q_\delta)$ be some deterministic tree automaton; in fact, we only need the functional behaviour of the state transition function δ which we also call a *distinguishing function*, which can be viewed as a partial mapping $V^{\mathrm{t}} \to Q_\delta$. A deterministic tree automaton $A = (Q, V, \bar{\delta}, F)$ is called *δ-distinguishable* if it satisfies the following two properties:

(1) For all states $q \in Q$ and all $x, y \in V^{\mathrm{t}}$ with $\bar{\delta}(x) = \bar{\delta}(y) = q$, we have $\delta(x) = \delta(y)$, *i.e.*, δ is defined on x and on y and the corresponding values coincide. In other words, for $q \in Q$, we may view δ also as a mapping $\delta : Q \to Q_\delta$ by defining $\delta(q) := \delta(x)$ for some x with $\bar{\delta}(x) = q$.

(2) For all $q_1, q_2 \in Q \cup V_0$, $q_1 \neq q_2$, with either (a) $q_1, q_2 \in F$ or (b) there exist $q_3 \in Q$, $k \geq 1$, $1 \leq i < k$, $p_1, \ldots, p_{k-1} \in Q \cup V_0$ and $f \in V_k$ with
$\bar{\delta}_k(f, p_1, \ldots, p_{i-1}, q_1, p_i, \ldots, p_{k-1}) = \bar{\delta}_k(f, p_1, \ldots, p_{i-1}, q_2, p_i, \ldots, p_{k-1}) = q_3$,
we have $\delta(q_1) \neq \delta(q_2)$.

A regular tree language T over V is called δ-*distinguishable* if it is accepted by a δ-distinguishable tree automaton. Let δ-DT denote the class of δ-distinguishable tree languages.

Notice one technical detail: the first item in the definition entails that the domain of $\bar{\delta}$ is a subset of the domain of δ. This means that the choice of a distinguishing function δ usually severely restricts the possible "compatible" δ-distinguishable tree automata, even if we only consider trees with label alphabet V.

However, there are generic ways how to define distinguishing functions that are compatible with any tree automaton, by defining $\delta = (\delta_i)_{i \geq 0}$ with $\delta_i : V_k \times (Q \cup V_0)^i \to Q$ so that δ can be interpreted as a state transition function for any arity. Then, the domain of δ would be all of V^{t}. For each fixed k, $\delta^k = (\delta_0, \ldots, \delta_k)$ defines a distinguishing function, and δ can be seen as a *homogeneous series of distinguishing functions*.

Let us consider some examples to clarify these notions.

Example 3.2. The distinguishing function induced by the trivial tree automaton with just one state leads to a slight generalization of the 0-reversible tree automata studied by Sakakibara [60]; in Sakakibara's model, for each arity $k \geq 1$, there was only one symbol σ_k permitted as label of the interior nodes.

Example 3.3. As a further simple example of distinguishing function, consider the function Ter defined by $Ter_0(a) = \{a\}$ and

$$Ter_k(f, q_1, \ldots, q_k) = \{f\} \cup \bigcup_{j=1}^{k} q_j,$$

where 2^V is the state set of A_{Ter}. Ter is the natural tree-analogue of the terminal distinguishing function basically introduced by Radhakrishnan and Nagaraja in [54]. Ter can be seen as a homogeneous series of distinguishing functions. As exhibited in [22] in the string case, several other similar distinguishing functions can be defined which also yield immediate analogues in the tree case. In particular, analogues to k-terminal distinguishable languages [19] and to the k-reversible languages [3] can be defined by means of the notions of k-roots, k-forks and k-subtrees as defined in [42].

Example 3.4. Without going into technical details here, the important notion of testable languages gives rise to an automaton that keeps track of the k-roots, k-forks and k-subtrees seen so far, and this automaton is finite when k is fixed and the underlying label alphabet V. (In fact, the literature is not completely coherent about this notion neither in the string nor in the tree case, but the basic idea prevails, see [29,41,56,57,68,69].) If $\delta^{V,k}$ denotes the corresponding transition

function, then k-testable languages over V are characterized as those languages that can be accepted by some finite tree automaton with transition function $\delta^{V,k}$. $\delta^{V,k}$ can be also seen as a homogeneous series of distinguishing functions. As it has been worked out for the string case in [19], $\delta^{V,k}$-distinguishable tree languages generalize the k-testable languages over V.

We now shortly discuss two important application areas for tree-based learning algorithms: the learning of string languages and the learning of derivation trees.

Inference algorithms for tree languages can be readily used for the inference of context-free string languages, once the structure of the derivation tree is fixed. We consider two examples of this technique:

- Observe that trees whose domain is contained in $\{1\}^*$ correspond to usual strings, and the notion of distinguishability obtained in this way is nearly the same as the one introduced in [22][5]. With this interpretation, learning algorithms for regular tree languages give rise to learning algorithms for regular string languages.
- If we restrict ourselves to *Ter* applied to derivation trees of even linear grammars (with trivial labels for interior nodes), we basically arrive at a variant of the terminal distinguishable even linear languages discussed in [23] in more detail. Here, focussing on derivation trees for even linear grammars basically entails a reordering of the letters, a technique addressed in particular in [26]. Possible pitfalls in this interpretation are discussed in the following example.

The following example expands a bit more on the description of derivation trees of context-free languages.

Example 3.5. Consider $A = (\{q_0, q_1\}, V, \delta, \{q_1\})$ with

$$V = \{a, *\}, \quad r_V = \{(a, 0), (*, 1), (*, 2), (*, 3)\} \quad \text{and}$$

$$\begin{aligned}
\delta_1(*, a) &= \delta_2(*, a, a) = q_0 \\
\delta_3(*, a, q_0, a) &= \delta_3(*, a, q_1, a) = q_1.
\end{aligned}$$

A is intended to accept a certain class of derivation trees of context-free grammars, namely, those of (even) linear grammars. A is not *Ter*-distinguishable, since it violates the second condition, because $Ter(q_0) = Ter(q_1) = \{*, a\}$. By way of contrast, if we consider an automaton $A' = (\{q_0\}, V, \delta', \{q_0\})$ with the same rank relation r_V and with δ' obtained by merging q_0 and q_1, *i.e.*, $\delta'_i = \delta_i$ for $i = 1, 2$ and $\delta'_3(*, a, q_0, a) = q_0$, then A' is *Ter*-distinguishable. Obviously, many more trees are accepted by A' than by A, *e.g.*, consider the tree t given by the term $*(*(a, a), *(a, a))$, since $\delta(t) = q_0$; this term surely does not correspond to a linear derivation tree.

If we like to keep the property of only accepting the originally intended class of derivation trees of context-free grammars, this is possible by modifying A to cope

---

[5]Except from the treatment of the empty word.

with larger alphabets. Consider, *e.g.*, $A'' = (\{q_0, q_1\}, V = \{a, *, +\}, \delta'', \{q_1\})$ with $r_V = \{(a, 0), (*, 1), (*, 2), (+, 3)\}$ and

$$\begin{aligned}
\delta_1''(*, a) &= \delta_2''(*, a, a) = q_0 \\
\delta_3''(+, a, q_0, a) &= \delta_3''(+, a, q_1, a) = q_1.
\end{aligned}$$

Since $Ter(q_0) = \{a, *\} \neq Ter(q_1) = \{a, *, +\}$, A'' is Ter-distinguishable. Obviously, the automata are constructed for accepting the usual derivation trees of (even-)linear grammars.

4. CHARACTERIZATIONS OF DISTINGUISHABLE TREE LANGUAGES

In this section, we derive as a main result a number of characterizations of each of the tree language classes δ-DT. Due to the similarity to the string case, we will omit some proofs in the following; instead, we refer to [22]. The reader comparing the given proofs with their string analogues will notice that, while the proof ideas are similar, the tree formalism entails more formal complications and care.

So, in the following, let δ be some arbitrary distinguishing function.

Remark 4.1. Any subautomaton of a δ-distinguishable tree automaton A is δ-distinguishable. $\square$

Lemma 4.2. *Let $A = (Q, V, \bar{\delta}, F)$ be a δ-distinguishable tree automaton. Let $u \in V_\$^{\mathsf{t}}$, $\{t_1, t_2\} \subseteq V^{\mathsf{t}}$. $\{u\#t_1, u\#t_2\} \subseteq T(A)$ and $\delta(t_1) = \delta(t_2)$ imply $\bar{\delta}(t_1) = \bar{\delta}(t_2)$.*

Proof. Firstly observe that, since $\delta(t_1) = \delta(t_2)$, we have $\delta(u\#t_1) = \delta(u\#t_2)$ for all $u \in V_\$^{\mathsf{t}}$ and $t_1, t_2 \in V^{\mathsf{t}}$. The proof proceeds via induction on the level ℓ of the node with label \$ in u. If $\ell = 0$, then $u = \$$. Consider the final states $q_i = \bar{\delta}(u\#t_i)$ of A for $i = 1, 2$. Since $\delta(q_1) = \delta(u\#t_1) = \delta(u\#t_2) = \delta(q_2)$, condition 2a for δ-distinguishable tree automata yields $q_1 = q_2$.

Assume the claim holds for $\ell < h$. Consider $u \in V_\$^{\mathsf{t}}$ with label \$ at level h. u can be uniquely represented as $u = u'\#f(s_1, \ldots, s_{i-1}, \$, s_i, \ldots, s_{k-1})$ for some $s_1, \ldots, s_{k-1} \in V^{\mathsf{t}}$ and some $u' \in V_\$^{\mathsf{t}}$ having the node labelled \$ at level $h - 1$. The induction hypothesis yields: if A accepts

$$u\#t_j = u'\#f(s_1, \ldots, s_{i-1}, t_j, s_i, \ldots, s_{k-1}), \quad j = 1, 2,$$

then $\bar{\delta}(f(s_1, \ldots, s_{i-1}, t_1, s_i, \ldots, s_{k-1})) = \bar{\delta}(f(s_1, \ldots, s_{i-1}, t_2, s_i, \ldots, s_{k-1}))$, since $\delta(t_1) = \delta(t_2)$ gives

$$\delta(f(s_1, \ldots, s_{i-1}, t_1, s_i, \ldots, s_{k-1})) = \delta(f(s_1, \ldots, s_{i-1}, t_2, s_i, \ldots, s_{k-1})).$$

Hence,

$$\begin{aligned}
\bar{\delta}_k(f, \bar{\delta}(s_1), \ldots, &\bar{\delta}(s_{i-1}), \bar{\delta}(t_1), \bar{\delta}(s_i), \ldots, \bar{\delta}(s_{k-1})) \\
&= \bar{\delta}_k(f, \bar{\delta}(s_1), \ldots, \bar{\delta}(s_{i-1}), \bar{\delta}(t_2), \bar{\delta}(s_i), \ldots, \bar{\delta}(s_{k-1})),
\end{aligned}$$

so that condition 2b for δ-dist-inguishable tree automata yields $q_1 = q_2$. $\square$

With the help of Lemma 4.2, the following statement is easily shown:

Lemma 4.3. *Let $A = (Q, V, \bar{\delta}, F)$ be a δ-distinguishable tree automaton. Let $\{u_1, u_2\} \subseteq V_{\$}^{\mathsf{t}}$ and $\{t, t'\} \subseteq V^{\mathsf{t}}$. If $\{u_1 \# t, u_2 \# t\} \subseteq T(A)$ and $\delta(t) = \delta(t')$, then $u_1 \# t' \in T(A)$ iff $u_2 \# t' \in T(A)$.*

Proof. Consider $u_1 \# t \in T(A)$. If $u_1 \# t' \in T(A)$, then Lemma 4.2 yields $\bar{\delta}(t) = \bar{\delta}(t')$. Hence, $u_2 \# t' \in T(A)$, because $u_2 \# t \in T(A)$ by assumption. Symmetrically, the other part of the claim follows. $\qquad\square$

Corollary 4.4. *If $T \subseteq V^{\mathsf{t}}$ is δ-distinguishable, then:*

$$\forall t, t' \in V^{\mathsf{t}} \text{ with } \delta(t) = \delta(t') \; \forall u_1, u_2 \in U_T(t) \; : u_1 \in U_T(t') \iff u_2 \in U_T(t').$$

Let $T \subseteq V^{\mathsf{t}}$ be a regular tree language. Let $A(T, \delta)$ denote the stripped subautomaton of $C(T) \times A_\delta$. Obviously, $T(A(T, \delta)) = T$. $A(T, \delta)$ is called the *δ-canonical tree automaton* of T. As the following theorem shows, we can take $A(T, \delta)$ as canonical objects describing δ-DT, since $A(T, \delta)$ is a unique object. Moreover, it is proved that the tree language class δ-DT can be characterized in a number of ways.

Theorem 4.5 (Characterization theorem). *Let $\delta : V^{\mathsf{t}} \to Q_\delta$ be a distinguishing function. Then, the following conditions are equivalent for a regular tree language $T \subseteq V^{\mathsf{t}}$:*

(1) *$T \in \delta$-DT.*
(2) *There is a tree automaton $A = (Q, V, \bar{\delta}, F)$ with $T(A) = T$ satisfying*

$$\forall t_1, t_2 \in V^{\mathsf{t}} \; \forall u \in V_{\$}^{\mathsf{t}} : \; (\{u \# t_1, u \# t_2\} \subseteq T \land \delta(t_1) = \delta(t_2)) \Rightarrow \bar{\delta}(t_1) = \bar{\delta}(t_2) \,.$$

(3) *$\forall t_1, t_2 \in V^{\mathsf{t}} \; \forall u, v \in V_{\$}^{\mathsf{t}} : \; (\{u \# t_1, v \# t_1\} \subseteq T \land \delta(t_1) = \delta(t_2)) \Rightarrow$*
$(u \# t_2 \in T \iff v \# t_2 \in T)$.
(4) *$\forall t_1, t_2 \in V^{\mathsf{t}} \; \forall u, v \in U_T(t_1) : \; \delta(t_1) = \delta(t_2) \Rightarrow$*
$(u \in U_T(t_2) \iff v \in U_T(t_2))$.
(5) *$A(T, \delta)$ is δ-distinguishable.*
(6) *$\forall u_1, u_2 \in V_{\$}^{\mathsf{t}} \; \forall t_1, t_2 \in V^{\mathsf{t}} : \; (\{u_1 \# t_1, u_2 \# t_1\} \subseteq T \land \delta(t_1) = \delta(t_2)) \Rightarrow$*
$U_T(t_1) = U_T(t_2)$.

Proof. 1. $\to$ 2. due to Lemma 4.2. According to the proof of Lemma 4.3, 2. $\to$ 3. The implications 3. $\leftrightarrow$ 4., 5. $\to$ 1. and 6. $\to$ 2. are trivial. 5. $\to$ 6. follows with Lemma 4.2.

We are going to show 4. $\to$ 5. in the following.

Consider a language T satisfying the condition 4. We have to show that $A(T, \delta)$ is δ-distinguishable. By definition, $A(T, \delta)$ is deterministic. Since (a subautomaton of) A_δ can be obtained from $A(T, \delta)$ by simple projection, $\delta(q)$ is well-defined for any state q of $A(T, \delta)$.

We now turn to the second condition of δ-distinguishable automata. Let q_1, q_2 be two states of $A(T, \delta)$ (or constant symbols) with $\hat{q} := \delta(q_1) = \delta(q_2)$. Hence, $q_i = (U_T(t_i), \hat{q})$ for some $t_i \in \mathrm{ST}(T)$.

Consider first case (a), *i.e.*, both q_1 and q_2 are final states. Then, $\{t_1, t_2\} \subseteq T$. Hence, $u := \$ \in U_T(t_i)$ for $i = 1, 2$. By condition 4 of the characterization theorem, we know that, for all $v \in U_T(t_1)$, $v \in U_T(t_2)$. Interchanging the roles of t_1 and t_2, we can conclude that $U_T(t_1) = U_T(t_2)$.

Regarding case (b), assume that there are states (or constant symbols) q_3, $p_1, \ldots, p_{k-1}$ such that

$$\bar{\delta}_k(f, p_1, \ldots, p_{i-1}, q_1, p_i, \ldots, p_{k-1}) = \bar{\delta}_k(f, p_1, \ldots, p_{i-1}, q_2, p_i, \ldots, p_{k-1}) = q_3$$

for some $f \in V_k$ and some $1 \le i < k$. Since $A(T, \delta)$ is stripped, there are a $\hat{u} \in V_\$^{\mathsf{t}}$ and $s_1, \ldots, s_{k-1} \in V^{\mathsf{t}}$ such that

$$\{\hat{u}\# f(s_1, \ldots, s_{i-1}, t_1, s_i, \ldots, s_{k-1}), \hat{u}\# f(s_1, \ldots, s_{i-1}, t_2, s_i, \ldots, s_{k-1})\} \subseteq T.$$

Hence, $u := \hat{u}\# f(s_1, \ldots, s_{i-1}, \$, s_i, \ldots, s_{k-1}) \in U_T(t_i)$ for $i = 1, 2$. Condition 4 of the characterization theorem shows again that $U_T(t_1) = U_T(t_2)$. $\square$

Remark 4.6. As in the string case [22], we could have added further characterizations of δ-DT by means of grammars or Myhill-Nerode like algebraic formulations. We did not do this here in order to avoid unnecessary technical complications. The interested reader is referred to the chapter written by Gécseg and Steinby in [58] and to [45] as regards the corresponding formalisms in the tree case.

The following lemma is useful for proving the correctness of our learning algorithms and is, moreover, a simple characterization of our canonical objects.

Lemma 4.7. *The stripped subautomaton of a δ-distinguishable tree automaton A is isomorphic to $A(T(A), \delta)$.*

Proof. According to Remark 4.1, the stripped subautomaton A' of A is δ-distinguishable. Let $A = (Q, V, \bar{\delta}, F)$ and $A' = (Q', V, \bar{\delta}', F')$. We have to show that, for all $q_1, q_2 \in Q'$ with $\delta(q_1) = \delta(q_2)$,

$$\{u \in V_\$^{\mathsf{t}} \mid \exists t \in V^{\mathsf{t}} \setminus V_0 : \bar{\delta}'(u\#q_1) \in F'\}$$
$$= \{u \in V_\$^{\mathsf{t}} \mid \exists t \in V^{\mathsf{t}} \setminus V_0 : \bar{\delta}'(u\#q_2) \in F'\}$$

implies that $q_1 = q_2$, since then, the mapping $q \mapsto (U_{T(A)}(t), \delta(q))$ for some $t \in V^{\mathsf{t}}$ with $\bar{\delta}'(t) = q$ will supply the required isomorphism.

Since A' is stripped, there are $t_1, t_2 \in V^{\mathsf{t}}$ and $u \in V_\$^{\mathsf{t}}$, $q_1 = \bar{\delta}'(t_1)$, $q_2 = \bar{\delta}'(t_2)$ and $\{u\#t_1, u\#t_2\} \subseteq T(A') = T(A)$. Since A' is δ-distinguishable, $\delta(q_1) = \delta(q_2)$ implies that $\delta(t_1) = \delta(t_2)$. Hence, we can apply Lemma 4.2 to show the result. $\square$

5. Inferrability

Learning from text. The learning model we use is *identification in the limit from positive samples* as proposed by Gold [32], sometimes also called *learning from text*. In this well-established model, a language class $\mathcal{L}$ (defined *via* a class of language

describing devices $\mathcal{D}$ as, *e.g.*, grammars or automata) is said to be *identifiable* if there is a so-called *inference machine* I to which as input an arbitrary language $L \in \mathcal{L}$ may be enumerated (possibly with repetitions) in an arbitrary order, *i.e.*, I receives an infinite input stream of words $E(1)$, $E(2)$, ..., where $E : \mathbb{N} \to L$ is an enumeration of L, *i.e.*, a surjection, and I reacts with an output device stream $D_i \in \mathcal{D}$ such that there is an $N(E)$ so that, for all $n \geq N(E)$, we have $D_n = D_{N(E)}$ and, moreover, the language defined by $D_{N(E)}$ equals L.

In order to ensure the convergence of the hypothesis stream output by a Gold-style learner, we need some well-defined canonical output objects. In the case of δ-DT, this will be the δ-canonical automata introduced above.

Telltale sets. According to [2] Theorem 1, a class $\mathcal{L}$ (Characterized by $\mathcal{D}$) comprised of recursive languages is identifiable iff, for any language description $D \in \mathcal{D}$ a so-called *telltale set* exists, *i.e.*, a finite subset $\chi(D) \subseteq L$ such that L is a minimal language from $\mathcal{L}$ containing $\chi(D)$.

For the tree language class δ-DT and some language $T \in \delta$-DT, consider the corresponding δ-canonical automaton $A(T, \delta) = (Q, V, \bar{\delta}, F)$ and define

$$
\begin{aligned}
\chi(T, \delta) \quad = \quad & \{\, u(q) \# t(q) \mid q \in Q \,\} \\
\cup \quad & \{\, u(\bar{\delta}_k(f, q_1, \ldots, q_k)) \# f(t(q_1), \ldots, t(q_k)) \mid \\
& \qquad q_1, \ldots, q_k \in Q \cup V_0, f \in V_k \,\},
\end{aligned}
$$

where $u(q) \in V_{\$}^{\mathsf{t}}$ and $t(q) \in V^{\mathsf{t}} \setminus V_0$ are (arbitrary) trees each of minimal size satisfying $\bar{\delta}(t(q)) = q$ (if $q \in Q$) and $\bar{\delta}(u(q) \# q) \in F$. If $q \in V_0$, we set $t(q) = q$. Naturally, a finite automaton for $\chi(T, \delta)$ may be computed by some Turing machine which is given $C(T)$ and A_δ as input.

Theorem 5.1 (Telltale property). *For each A_δ and each $T \in \delta$-DT, $\chi(T, \delta)$ is a telltale set of T.*

Proof. Clearly, $\chi(T, \delta) \subseteq T$. Consider some tree language $T' \in \delta$-DT with $\chi(T, \delta) \subseteq T'$. We have to show that $T \subseteq T'$. Let $A(T, \delta) = (Q, V, \bar{\delta}, F)$.

By induction on the height of s, we show

$$
(*) \quad U_{T'}(s) = U_{T'}(t(\bar{\delta}(s))) \quad \text{and} \quad \delta(s) = \delta(t(\bar{\delta}(s)))
$$

for all $s \in \mathrm{ST}(T)$. Note that $(*)$ implies the following: if $s \in T$, *i.e.*, $q_f = \bar{\delta}(s)$ is a final state of $A(T, \delta)$, then $U_{T'}(t(q_f))$ is a final state of $C(T')$, because $t(q_f) \in \chi(T, \delta) \subseteq T'$. Therefore, $(U_{T'}(t(q_f)), \delta(t(q_f)))$ is a final state of $A(T', \delta)$. Due to $(*)$, we conclude that $s \in T'$. Hence, $T \subseteq T'$.

Now, we prove $(*)$. If the height of s is zero, then $s \in V_0$, which means that $s = t(s)$ by definition of $t(\cdot)$. Assume that $(*)$ holds for all trees of depth at most $h \geq 0$. Consider some $s \in \mathrm{ST}(T)$ of depth $h + 1$, *i.e.*, $s = f(s_1, \ldots, s_k)$ for some $f \in V_k$, $s_1, \ldots, s_k \in \mathrm{ST}(T)$; obviously, all s_i are trees of depth at most h. By the induction hypothesis,

$$
U_{T'}(s_i) = U_{T'}(t(\bar{\delta}(s_i))) \quad \text{and} \quad \delta(s_i) = \delta(t(\bar{\delta}(s_i))), \quad 1 \leq i \leq k.
$$

Therefore,

$$
\begin{aligned}
U_{T'}(s) &= U_{T'}(f(s_1,\ldots,s_k)) \\
&= \bar{\delta}_k(f, U_{T'}(s_1),\ldots,U_{T'}(s_k)) \\
&= \bar{\delta}_k(f, U_{T'}(t(\bar{\delta}(s_1))),\ldots,U_{T'}t(\bar{\delta}(s_k))) \\
&= U_{T'}(f(t(\bar{\delta}(s_1)),\ldots,t(\bar{\delta}(s_k))))
\end{aligned}
$$

and

$$
\delta(s) = \delta(f(s_1,\ldots,s_k)) = \delta(f(t(\bar{\delta}(s_1)),\ldots,t(\bar{\delta}(s_k)))).
$$

Define $q' = \bar{\delta}_k(f, \bar{\delta}(s_1),\ldots,\bar{\delta}(s_k))$. Since by definition of the telltale set, both

$$
u(q')\#f(t(\bar{\delta}(s_1)),\ldots,t(\bar{\delta}(s_k))) \quad \text{and} \quad u(q')\#t(q')
$$

are contained in $\chi(T,\delta) \subseteq T'$, Lemma 4.2 yields

$$
U_{T'}(s) = U_{T'}(f(t(\bar{\delta}(s_1)),\ldots,t(\bar{\delta}(s_k)))) = U_{T'}(t(q')),
$$

because

$$
\begin{aligned}
\delta(t(q')) &= \delta(\bar{\delta}_k(f, \bar{\delta}(s_1),\ldots,\bar{\delta}(s_k))) \\
&= \delta(f(t(\bar{\delta}(s_1)),\ldots,t(\bar{\delta}(s_k)))). \quad \square
\end{aligned}
$$

6. INFERENCE ALGORITHMS

For each A_δ, we sketch an algorithm which receives an input sample set $I_+ = \{t_1,\ldots,t_M\}$ (a finite subset of the tree language $T \in \delta\text{-DT}$ to be identified) and finds a minimal language $T' \in \delta\text{-DT}$ which contains I_+. Of course, Theorem 5.1 already guarantees the existence of such an algorithm, but the ad-hoc enumeration algorithm is not very efficient. In contrast, our algorithms will have polynomial update time, but the number of so-called implicit errors of prediction is not polynomially bounded, as explained by Sakakibara for his simpler setting [60].

Our merging state inference algorithm $\delta\text{-}\mathtt{Ident}$ for $\delta\text{-DT}$ now starts with the automaton $A_0 = Bs(I_+) = (Q = \mathrm{ST}(T) \setminus V_0, V, \bar{\delta}, F = I_+)$ on receiving I_+ as input. Then, it subsequently merges two states which cause a conflict to one of the requirements for δ-distinguishable automata. This way, we get a sequence of automata $A_0, A_1, \ldots, A_f$ each of which can be interpreted as a quotient automaton of A_0 by the partition of the state set of A_0 induced by the corresponding merging operation. Let δ^i denote the transition function of A_i. Observe that each A_i is stripped, since A_0 is stripped. Moreover, A_f is δ-distinguishable, as being the last automaton in this chain. In terms of the partitions inducing the mentioned

quotient automata, δ-Ident starts with the trivial partition π_0 of Q into singletons and repeatedly merges two distinct blocks B_1 and B_2 at stage i, $i = 0, \ldots, f-1$ if any of the following conditions is satisfied:

Final state conflict: B_1 and B_2 contain both final states $q_1 \in B_1$, $q_2 \in B_2$ of A_0 with $\delta(q_1) = \delta(q_2)$. This would mean that B_1 and B_2 are both final states (in A_i) with $\delta(B_1) = \delta(B_2)$.

Determinism conflict: there exist two states $q_1 \in B_1$, $q_2 \in B_2$ of the form

$$q_1 = f(p_1, \ldots, p_k) \quad \text{and} \quad q_2 = f(p'_1, \ldots, p'_k)$$

such that, for all $1 \le j \le k$, either $p_j = p'_j \in V_0$ or $B(p_j, \pi_i) = B(p'_j, \pi_i)$. Namely, if this situation occurred, this would mean that

$$B_i = \delta^i(f, B(p_1, \pi_i), \ldots, B(p_k, \pi_i)) \text{ for } i = 1, 2.$$

Backward determinism conflict: there exist two states q_1, q_2 of the form

$$q_1 = f(p_1, \ldots, p_k) \quad \text{and} \quad q_2 = f(p'_1, \ldots, p'_k)$$

with $B(q_1, \pi_i) = B(q_2, \pi_i)$ and an integer $1 \le \ell \le k$ such that, for all $1 \le j \le k$ with $i \ne \ell$, either $p_j = p'_j \in V_0$ or $B(p_j, \pi_i) = B(p'_j, \pi_i)$. Moreover, $p_\ell \in B_1$, $p'_\ell \in B_2$ and $\delta(p_\ell) = \delta(p'_\ell)$.

To be more concrete, consider the following program fragment:

Algorithm 6.1 (δ-Ident).
Input: a nonempty positive sample $I_+ \subseteq V^{\mathrm{t}}$.
Output: $A(T, \delta)$, where T is a minimal δ-distinguishable tree language containing I_+.
*** Initialization
Let $A_0 = (Q \setminus V_0, V, \bar{\delta}, F) = Bs(I_+)$ with $Q = ST(T)$ and $F = I_+$.
Let π_0 be the trivial partition of Q into singletons.
Let LIST contain all unordered pairs $\{q, q'\}$ of final states of Q such that $q \ne q'$ and $\delta(q) = \delta(q')$.
Let $i := 0$.
*** Merging
While LIST$\ne \emptyset$ do begin
    Remove some element $\{q_1, q_2\}$ from LIST.
    Consider the blocks $B_1 = B(q_1, \pi_i)$ and $B_2 = B(q_2, \pi_i)$.
    If $B_1 \ne B_2$, then begin
        Let π_{i+1} be π_i with B_1 and B_2 merged.
        Identify newly produced determinism conflicts & update LIST.
        Identify newly produced backward determinism conflicts & update LIST.
        Increment i by one.
        If $i = |Q| - 1$, then LIST$:= \emptyset$.
    end *** if
end *** while

The conflict resolution can be implemented either by means of an explicit search through all possible conflict situations, which results in an algorithm similar to the one proposed by Sakakibara [60] or by keeping track of the forward and backward transition functions of the automata A_i with the help of union/find techniques, as elaborated in [25] in the case of string language identification; then, the running time of the algorithm is linear for all practical purposes. In either case, we obtain algorithms with polynomially bounded update times, see [60] for the definitions.

Example 6.2. If $I_+ = \{(a, *(a), a), *(a, *(a, a), a), *(a)\}$ is given to Ter-`Ident`, then the automaton A' from Example 3.5 will result.

Moreover, it is possible to design so-called *incremental versions* of the algorithms, where the input sample is fed to the algorithm in an on-line manner.

We now give the ingredients for showing the correctness of δ-`Ident`. The following lemma is crucial in this respect:

Lemma 6.3. *Let $I_+ \subseteq T \in \delta$-$DT$ be given. Let π be the partition of $A_0 = Bs(I_+)$ described by: q_1, q_2 belong to the same block if[6] $U_T(q_1) = U_T(q_2)$ and $\delta(q_1) = \delta(q_2)$. Then, $\pi^{-1}A_0$ is isomorphic to a subautomaton of $A(T, \delta)$.*

Proof. Let $\pi^{-1}A_0 = (\hat{Q}, V, \hat{\delta}, \hat{F})$ and $A(T, \delta) = (Q, V, \bar{\delta}, F)$. By definition, $\hat{Q} = \{B(t, \pi) \mid t \in \mathrm{ST}(I_+) \setminus V_0\}$ and the mapping

$$h : \hat{Q} \to Q, B(t, \pi) \mapsto (U_T(t), \delta(t))$$

is well-defined and injective. If $B_1 \in \hat{F}$, then $B_1 = B(t, \pi)$ for some $t \in I_+ \subseteq T$, and hence, $(U_T(t), \delta(t)) \in F$. Therefore, $h(\hat{F}) \subseteq F$. $\pi^{-1}A_0$ is deterministic, because, if

$$f(s_1, \ldots, s_k), \ f(s'_1, \ldots, s'_k) \in \mathrm{ST}(I_+),$$

with $B(s_i, \pi) = B(s'_i, \pi)$ if $s_i, s'_i \in \mathrm{ST}(I_+) \setminus V_0$ and $s_i = s'_i$ if $s_i, s'_i \in V_0$, then

$$B(f(s_1, \ldots, s_k), \pi) = B(f(s'_1, \ldots, s'_k), \pi)$$

for any $f \in V_k$. h is an automaton morphism, since

$$h(\hat{\delta}_k(f, q_1, \ldots, q_k)) = h(B(f(t_1, \ldots, t_k), \pi))$$

with $t_i = q_i$ if $q_i \in V_0$ and t_i chosen otherwise to satisfy $B(t_i, \pi) = q_i$. Hence,

$$
\begin{aligned}
h(\hat{\delta}_k(f, q_1, \ldots, q_k)) &= (U_T(f(t_1, \ldots, t_k)), \delta(f(t_1, \ldots, t_k))) \\
&= \bar{\delta}_k(f, (U_T(t_1), \delta(t_1)), \ldots, (U_T(t_k), \delta(t_k))).
\end{aligned}
$$

So, h is an isomorphism between $\pi^{-1}A_0$ and a tree subautomaton of $A(T, \delta)$. $\square$

Theorem 6.4. *Fix A_δ. Consider a chain of automata $A_0, A_1, \ldots, A_f$ obtained by applying the sketched algorithm δ-`Ident` on input sample I_+, where $A_0 = Bs(I_+)$.*

---

[6]Recall that $q_i \in \mathrm{ST}(I_+) \setminus V_0$.

Then, we have:

(1) $T(A_0) \subseteq T(A_1) \subseteq \cdots \subseteq T(A_f)$.
(2) A_f *is δ-distinguishable and stripped.*
(3) *The partition π_f of the state set of A_0 corresponding to A_f is the finest partition π of the state set of A_0 such that the quotient automaton $\pi^{-1}A_0$ is δ-distinguishable.*

Proof.

1. is clear, since δ-`Ident` is a merging states algorithm;
2. follows almost by definition;
3. can be shown by induction, proving that each π_i corresponding to A_i refines π, quite analogous to [3] Lemma 25 and [60] Lemma 13.   □

Theorem 6.5. *In the notations of Theorem 6.4, $T(A_f)$ is a minimal δ-distinguishable language containing I_+.*

Proof. Theorem 6.4 states that $T(A_f) \in \delta$-DT and $I_+ = T(A_0) \subseteq T(A_f)$. Consider now an arbitrary language $T \in \delta$-DT containing I_+. We consider the quotient automaton $\pi^{-1}A_0$ defined in Lemma 6.3. This Lemma shows that

$$T(\pi^{-1}A_0) \subseteq T = T(A(T,\delta)).$$

By Remark 4.1, $\pi^{-1}A_0$ is δ-distinguishable, because $A(T,\delta)$ is δ-distinguishable due to Theorem 4.5. Theorem 6.4 yields that π_f refines π, so that

$$T(A_f) = T(\pi_f^{-1}A_0) \subseteq T(\pi^{-1}A_0) \subseteq T.   □$$

Remark 6.6. Up to now, in accordance with the definition of a telltale set, we always spoke about *a minimal δ-distinguishable language* containing the sample I_+. Considering again the previous proof, one sees that there is actually a *unique* minimal language in δ-DT containing I_+, so that we can talk about *the smallest* language in δ-DT containing I_+ in the following. This means that each telltale is in fact a *characteristic sample* as defined in [3].

Theorem 6.5 immediately yields:

Corollary 6.7 (Correctness of δ-`Ident`). *Let $(t_i)_{i\geq 1}$ be an enumeration of $T \in \delta$-DT. Then, the sequence $(\delta$-`Ident`$(\{t_1,\ldots,t_i\}))_{i\geq 1}$ converges to the δ-canonical automaton $A(T,\delta)$.*

Proof. At some point N of the enumeration process, the telltale set $\chi(T,\delta)$ will have been given to δ-`Ident`. By combining Theorems 5.1 and 6.5, for all $n \geq N$ and all automata A_n output by δ-`Ident`, we have $T(A_n) = T$. The argument of Theorem 6.5 shows that each A_n (with $n \geq N$) is isomorphic to a subautomaton of $A(T,\delta)$ generating $T = T(A(T,\delta))$. Since each A_n is stripped, it must be isomorphic to $A(T,\delta)$ for $n \geq N$ due to Lemma 4.7.   □

We finally remark that the performance of the general algorithm δ-`Ident` sketched above depends on the size of A_δ (since the telltale set $\chi(T, \delta)$ we defined above depends on this size). Hence, δ-`Ident` offers a tradeoff of the following form:

- the larger A_δ, the slower performs the algorithm δ-`Ident`, but the identifiable language class tends to be bigger;
- the smaller A_δ, the faster performs the algorithm δ-`Ident`, but the identifiable language class tends to be smaller.

The relationship between the size of A_δ and the size of the corresponding language class can be made precise as follows; the easy proof of the assertion is omitted.

Proposition 6.8. *If A_δ is a homomorphic image of A_γ, then δ-$DT \subseteq \gamma$-DT.*

7. Approximation

We are going to show that, for any class δ-DT, all regular tree languages may be approximated by some language from δ-DT in a certain sense. Firstly, we give the necessary general definitions due to Kobayashi and Yokomori [44].

Let $\mathcal{L}$ be a language class and L be a language, possibly outside $\mathcal{L}$. An *upper-best approximation* $\bar{\mathcal{L}}L$ *of L with respect to $\mathcal{L}$* is defined to be a language $L_* \in \mathcal{L}$ containing L such that for any $L' \in \mathcal{L}$ with $L \subseteq L'$, $L_* \subseteq L'$ holds. If such an L_* does not exist, $\bar{\mathcal{L}}L$ is undefined.

Let us consider a simple example:

Example 7.1. Let $\mathcal{L}$ be the class of co-finite languages over an alphabet Σ, *i.e.*, $\mathcal{L} = \{\Sigma^* \setminus F \mid F \subset \Sigma^*, |F| < \infty\}$. Then, the empty set has no upper-best approximation $\bar{\mathcal{L}}\emptyset$, since for every $L_* = \Sigma^* \setminus F_* \in \mathcal{L}$, there is a $w \in \Sigma^* \setminus F_*$ (since F_* is finite) such that $L' = \Sigma^* \setminus (F_* \cup \{w\} \in \mathcal{L}$ and $\emptyset \subseteq L'$, but $L_* \supsetneq L'$ holds.

Remark 7.2. If $\mathcal{L}$ is closed under intersection, then $\bar{\mathcal{L}}L$ is uniquely defined (or undefined).

Consider an inference machine I to which as input an arbitrary language L may be enumerated (possibly with repetitions) in an arbitrary order, *i.e.*, I receives an infinite input stream of words $E(1), E(2), \ldots$, where $E : \mathbb{N} \to L$ is an enumeration of L. We say that *I identifies an upper-best approximation of L in the limit (from positive data) by $\mathcal{L}$* if I reacts on an enumeration of L with an output device stream $D_i \in \mathcal{D}$ such that there is an $N(E)$ so that, for all $n \geq N(E)$, we have $D_n = D_{N(E)}$ and, moreover, the language defined by $D_{N(E)}$ equals $\bar{\mathcal{L}}L \in \mathcal{L}$.

Let $\mathcal{L}_1$ and $\mathcal{L}_2$ be two language classes. We say that $\mathcal{L}_1$ has the *upper-best approximation property (u.b.a.p.) with respect to $\mathcal{L}_2$* iff, for every $L \in \mathcal{L}_2$, $\bar{\mathcal{L}}_1 L$ is defined.

A language class $\mathcal{L}_1$ is called *upper-best approximately identifiable in the limit (from positive data) by $\mathcal{L}_2$* iff there exists an inference machine I which identifies an upper-best approximation of each $L \in \mathcal{L}_1$ in the limit (from positive data) by $\mathcal{L}_2$. Observe that this notion of identifiability coincides with Gold's classical notion of learning in the limit in the case when $\mathcal{L}_1 = \mathcal{L}_2$.

Consider a language class $\mathcal{L}$ and a language L from it. A finite subset $F \subseteq L$ is called a *telltale set* of L with respect to $\mathcal{L}$ iff, for any $L' \in \mathcal{L}$, $F \subseteq L'$ implies that $L \subseteq L'$.

Now, let us turn more specifically to the distinguishable languages. Fix some distinguishing function δ. We call a language $T \subseteq V^{\mathsf{t}}$ *pseudo-δ-distinguishable* iff, for all $t_1, t_2 \in V^{\mathsf{t}}$ with $\delta(t_1) = \delta(t_2)$ and for all $u \in V^{\mathsf{t}}_{\$}$, we have $U_T(t_1) = U_T(t_2)$ whenever $\{u\#t_1, u\#t_2\} \subseteq T$. By our characterization theorem, $T \in \delta\text{-DT}$ iff T is a pseudo-δ-distinguishable and regular tree language.

Immediately from the definition, we may conclude:

Proposition 7.3. *Let $T_1 \subseteq T_2 \subseteq \ldots$ be any ascending sequence of pseudo-δ-distinguishable languages. Then, $\bigcup_{i \geq 1} T_i$ is pseudo-δ-distinguishable.*

For brevity, we write $t_1 \equiv_{T,\delta} t_2$ iff $U_T(t_1) = U_T(t_2)$ and $\delta(t_1) = \delta(t_2)$.

Remark 7.4. If $T \subseteq V^{\mathsf{t}}$ is a regular tree language and if $\delta : V^{\mathsf{t}} \to Q_\delta$ is some distinguishing function, then the number of equivalence classes of $\equiv_{T,\delta}$ equals the number of states of $C(T)$ (plus one) times $|Q_\delta|$, and this is just the number of states of $A(T, \delta)$ (plus $|Q_\delta|$).

Let $T \subseteq V^{\mathsf{t}}$. For any integer i, we will recursively define $R_\delta(i, T)$ as follows:
  (1) $R_\delta(0, T) = T$ and
  (2) $R_\delta(i, T) = R_\delta(i-1, T)$
$$\cup \{\, u\#t_2 \mid u\#t_1, u'\#t_1, u'\#t_2 \in R_\delta(i-1, T) \wedge \delta(t_1) = \delta(t_2) \,\}$$
$$\text{for } i \geq 1.$$

Furthermore, set $R_\delta(T) = \bigcup_{i \geq 0} R_\delta(i, T)$.

Since R_δ turns out to be a hull operator, the following statement is obvious.

Proposition 7.5. *For any tree language T and any distinguishing function δ, $R_\delta(T)$ is the smallest pseudo-δ-distinguishable language containing T.*

Lemma 7.6. *Let $T \subseteq V^{\mathsf{t}}$ be any tree language. If t_1 and t_2 are subtrees of T, then $t_1 \equiv_{T,\delta} t_2$ implies that $U_{R_\delta(T)}(t_1) = U_{R_\delta(T)}(t_2)$.*

Proof. Let t_1 and t_2 be subtrees of T with $t_1 \equiv_{T,\delta} t_2$. By definition of $\equiv_{T,\delta}$, $U_T(t_1) = U_T(t_2) \neq \emptyset$. Hence, there is a tree $u \in V^{\mathsf{t}}_{\$}$ so that $\{u\#t_1, u\#t_2\} \subseteq T \subseteq R_\delta(T)$. Furthermore, by definition of $\equiv_{T,\delta}$, $\delta(t_1) = \delta(t_2)$. Since $R_\delta(T)$ is pseudo-δ-distinguishable due to Proposition 7.5, $U_{R_\delta(T)}(t_1) = U_{R_\delta(T)}(t_2)$. $\square$

Similarly to the string case [22], we can show:

Lemma 7.7. *Let $T \subseteq V^{\mathsf{t}}$ be any tree language and let δ be any distinguishing function. Then, for any subtree t_1 of $R_\delta(T)$, there exists a subtree t_2 of T with $U_{R_\delta(T)}(t_1) = U_{R_\delta(T)}(t_2)$.*

Proof. Since t_1 is a subtree of $R_\delta(T)$ iff t_1 is a subtree of $R_\delta(i, T)$ for some $i \geq 0$, it suffices to show the following claim by induction.

Let $i \geq 0$. Then, for any subtree t_1 of $R_\delta(i, T)$, there exists a subtree t_2 of T with $U_{R_\delta(T)}(t_1) = U_{R_\delta(T)}(t_2)$.

Trivially, the claim is true when $i = 0$, since $R_\delta(0, T) = T$. As an induction hypothesis, assume that the claim is shown for $i = \ell$. Hence, we have to consider some $t_1 \in \mathrm{ST}(R_\delta(\ell + 1, T))$ in the induction step. Consider some

$$u \# t_1 \in R_\delta(\ell + 1, T) \setminus R_\delta(\ell, T).$$

This means that there are trees $u_1, u_2 \in V_\$^{\mathsf{t}}$ and $t, t' \in V^{\mathsf{t}}$ with

$$\{u_1 \# t, u_2 \# t, u_2 \# t'\} \subseteq R_\delta(\ell, T), \quad \delta(t_1) = \delta(t_2) \quad \text{and} \quad u_1 \# t' = u \# t_1.$$

We encounter three possible situations:

(1) If t_1 is a subtree of t', then t_1 is a subtree of $u_2 \# t' \in R_\delta(\ell, T)$, and the claim follows by the induction hypothesis.

(2) If t_1 is not subtree of t' and if t' is not subtree of t_1, then t_1 is a subtree of u_1 and is, hence, a subtree of $u_1 \# t \in R_\delta(\ell, T)$, so that the induction hypothesis is again applicable.

(3) If t' is a subtree of t_1, then $t_1 = u' \# t'$ for some $u' \in V_\$^{\mathsf{t}}$. Since $R_\delta(T)$ is pseudo-δ-distinguishable and $\{u_2 \# t, u_2 \# t'\} \subseteq R_\delta(T)$ as well as $\delta(t) = \delta(t')$, $U_{R_\delta(T)}(t) = U_{R_\delta(T)}(t')$, which yields $U_{R_\delta(T)}(t_1) = U_{R_\delta(T)}(u' \# t') = U_{R_\delta(T)}(u' \# t)$. Since u' is a subtree of u_1, $u' \# t$ is a subtree of $u_1 \# t \in R_\delta(\ell, T)$. By our induction hypothesis, there is a subtree t_2 of T such that $U_{R_\delta(T)}(t_2) = U_{R_\delta(T)}(u_1 \# t) = U_{R_\delta(T)}(t_1)$.

$\square$

By a reasoning completely analogous to [44], we may conclude:

Theorem 7.8. *For any distinguishing function δ, the class δ-DT has the u.b.a.p. with respect to the class of regular tree languages.* $\square$

Observe that the number of states of $A_{R_\delta(T)}$ is closely related to the number of states of $A(T, \delta)$, see Remark 7.4.

Theorem 7.9. *For any distinguishing function δ, the class of regular languages is upper-best approximately identifiable in the limit from positive data by δ-DT.* $\square$

In addition to the last two theorems, we remark that an upper-best approximation of a regular tree language with respect to each class δ-DT is uniquely defined, since the classical product automaton construction shows that each of these classes is closed under intersection, see Remark 7.2.

Given some tree automaton A and some distinguishing function δ, an automaton accepting $\overline{\delta\text{-DT}}T(A)$ can be constructed as follows:

(1) Compute $C(T(A))$.

(2) Construct $A' = A(T(A), \delta)$.

(3) Merge "conflicting states" in A' as long as possible.

8. DISCUSSION AND PROSPECTS

For a variety of regular tree language classes, we showed in which way they can be efficiently inferred. To this end, we had to define new canonical automata specific to each of these classes. Each of these classes can be characterized in various ways. Moreover, we showed that every regular tree language can be approximated in a well-defined manner by languages from δ-DT for any chosen A_δ. In this context, the question what δ to choose seems to be quite interesting for practical purposes.

In the future, we will try to compare our work with other works on the inference of tree languages and of context-free languages, as they are contained, *e.g.*, in [5,27,30,33,43,47,48,61,64–66,71–73]. Moreover, it would be interesting to extend the work to other, more general classes of tree languages and the corresponding languages of yielded strings, see [70] for a short exposition.

Our results can be also employed for devising learning algorithms for linear languages, which will be a complimentary approach to the ones detailed in [55,62], as well as in [23]. In order to design string language learning algorithms based on tree learning, it seems to be best to prescribe, for each word length n, a skeleton tree S_n which defines how to parse strings of length n. There seem to be interesting connections to the idea of employing permutation (families) for learning as detailed in [26].

Of course, applications of our learning algorithms in the various domains where tree languages have been successfully applied to would be interesting. One of the nicest resources on the web is the "treebag page"

$$\texttt{www.informatik.uni-bremen.de/theorie/treebag/}$$

A corresponding book [15] is currently in print.

Together with L. Buisman, we have started investigating the possibility of applying tree learning algorithms to shock tree classification problems, see [39,63]. It would then probably become important to also incorporate "error-correcting features," since real-world data is always containing noise in some form, see [48, 49,51,67].

Finally, an implementation of our tree language inference algorithms, done by A. Radl and later worked on by L. Buisman, will be accessible through our new homepage at the University of Trier; for the string case, please consult

$$\texttt{www-fs.informatik.uni-tuebingen.de/\~{}fernau/GI.htm}$$

see [25].

References

[1] H. Ahonen, H. Mannila and E. Nikunen, Forming grammars for structured documents: an application of grammatical inference, in *Proceedings of the Second International Colloquium on Grammatical Inference (ICGI-94): Grammatical Inference and Applications*, edited by R.C. Carrasco and J. Oncina. *Lect. Notes Artif. Int.* **862** (1994) 153–167.

[2] D. Angluin, Inductive inference of formal languages from positive data. *Inform. Comput.* **45** (1980) 117–135.

[3] D. Angluin, Inference of reversible languages. *J. ACM* **29** (1982) 741–765.

[4] D. Angluin, Learning regular sets from queries and counterexamples. *Inform. Comput.* **75** (1987) 87–106.

[5] M. Bernard and C. de la Higuera, GIFT: grammatical inference for terms, in *Conférence d'Apprentissage, Palaiseau*, May 1999. English version: Late breaking paper of International Conference on Inductive Logic Programming; French journal version: Apprentissage de programmes logiques par inférence grammaticale. *Revue d'Intelligence Artificielle* **14** (2001) 375–396.

[6] J. Besombes and J.-Y. Marion, Identification of reversible dependency tree languages, in *Proc. 3rd Workshop on Learning Languages in Logic LLL'01*, edited by L. Popelínský and M. Nepil, 11–22. Technical report FIMU-RS-2001-08, FI MU Brno, Czech Republic, see `http://www.fi.muni.cz/ilpnet2/LLL2001/#proceedings`, September 2001.

[7] J. Besombes and J.-Y. Marion, Learning tree languages from positive examples and membership queries, in *Algorithmic Learning Theory, 15th International Conference, ALT*, edited by S. Ben-David, J. Case and A. Maruoka. *Lect. Notes Comput. Sci.* **3244** (2004) 440–453.

[8] H. Boström, Theory-guided induction of logic programs by inference of regular languages. in *Proc. of the 13th International Conference on Machine Learning*, Morgan Kaufmann, (1996) 46–53.

[9] R.C. Carrasco, J. Oncina and J. Calera-Rubio, Stochastic inference of regular tree languages. *Mach. Learn.* **44** (2001) 185–197.

[10] R. C. Carrasco and J. R. Rico-Juan, A similarity between probabilistic tree languages: Application to XML document families. *Pattern Recogn.* **36** (2003) 2197–2199.

[11] M. Ceresna and G. Gottlob, Query based learning of XPath fragments, in *Proceedings of Dagstuhl Seminar on Machine Learning for the Semantic Web (05071)*, Dagstuhl, Germany, (2005).

[12] S. Crespi-Reghizzi, M.A. Melkanoff and L. Lichten, The use of grammatical inference for designing programming languages. *Comm. ACM* **16** (1972) 83–90.

[13] E. Cypher, D.C. Halbert, D. Kurlander, H. Liebermanand D. Maulsby, B.A. Myers and A. Turransky, Eds. *Watch What I Do: Programming by Demonstration*. MIT Press, 1993. Available online at `http://www.acypher.com/wwid/WWIDToC.html`

[14] A. Dix and A. Patrick, Query by browsing. in *Proceedings of IDS'94: The 2nd International Workshop on User Interfaces to Databases*, edited by P. Sawyer, Springer (1994) 236–248. HTML version:
`http://www.comp.lancs.ac.uk/computing/users/dixa/papers/QbB-IDS94/`

[15] F. Drewes, *Grammatical Picture Generation – A Tree-Based Approach*. Texts in Theoretical Computer Science. An EATCS Series. Springer, (2006). More information can be found on the web page `http://www.cs.umu.se/~drewes/picgen/index.html`

[16] F. Drewes and J. Högberg, Learning a regular tree language from a teacher. In *Proc. Developments in Language Theory DLT 2003*, edited by Z. Ésik and Z. Fülöp. *Lect. Notes Comput. Sci.* **2710** (2003) 279–291.

[17] F. Drewes and J. Högberg, Query learning of regular tree languages: How to avoid dead states. *Theor. Comput. Syst.* (2006). To appear.

[18] L.F. Fass, Learning context-free languages from their structured sentences. *SIGACT News*, **15** (1983) 24–35.

[19] H. Fernau, *k*-gram extensions of terminal distinguishable languages. In *International Conference on Pattern Recognition (ICPR 2000). IEEE/IAPR* **2** (2000) 125–128.

[20] H. Fernau, Learning XML grammars. in *Machine Learning and Data Mining in Pattern Recognition MLDM'01*, edited by P. Perner. *Lect. Notes Artif. Int.* **2123** (2001) 73–87.

[21] H. Fernau, Learning tree languages from text. in *Computational Learning Theory COLT 2002*, edited by J. Kivinen and R.H. Sloan. *Lect. Notes Artif. Int.* **2375** (2002) 153–168.

[22] H. Fernau, Identification of function distinguishable languages. *Theoret. Comput. Sci.* **290** (2003) 1679–1711.

[23] H. Fernau, Identifying terminal distinguishable languages. *Ann. Math. Artif. Intell.* **40** (2004) 263–281.

[24] H. Fernau and C. de la Higuera, Grammar induction: An invitation to formal language theorists. *GRAMMARS* **7** (2004) 45–55.

[25] H. Fernau and A. Radl, Algorithms for learning function distinguishable regular languages. in *Structural, Syntactic, and Statistical Pattern Recognition SSPR and SPR 2002*, edited by T. Caelli, A. Amin, R.P.W. Duin, M. Kamel and D. de Ridder. *Lect. Notes Comput. Sci.* **2396** (2002) 64–72.

[26] H. Fernau and J.M. Sempere, Permutations and control sets for learning non-regular language families. in *Grammatical Inference: Algorithms and Applications, 5th International Colloquium (ICGI 2000)*, edited by A.L. Oliveira. *Lect. Notes Artif. Int.*, **1891** (2000) 75–88.

[27] C.C. Florêncio, Consistent identification in the limit of any of the classes of *k*-valued is NP-hard. in *Logical Aspects of Computational Linguistics LACL'01*, edited by P. de Groote, G. Morrill and C. Retoré. *Lect. Notes Artif. Int.* **2099** (2001) 125–138.

[28] H. Fukuda and K. Kamata, Inference of tree automata from sample set of trees. *Int. J. Comput. Inform. Sci.* **13** (1984) 177–196.

[29] P. García, Learning *k*-testable tree sets from positive data. Technical Report DSIC/II/46/1993, Departamento de Sistemas Informáticos y Computación, Universidad Politécnica de Valencia, `http://www.dsic.upv.es/users/tlcc/tlcc.html` (1993).

[30] P. García and J. Oncina, *Inference of recognizable tree sets.* Technical Report DSIC-II/47/93, Departamento de Sistemas Informáticos y Computación (1993).

[31] M. Garofalakis, A. Gionis, R. Rastogi, S. Seshadri and K. Shim, XTRACT: learning document type descriptors from XML document collections. *Data Min. Knowl. Discov.* **7** (2003) 23–56.

[32] E.M. Gold, Language identification in the limit. *Inform. Comput.* **10** (1967) 447–474.

[33] R.C. Gonzalez and M.G. Thomason, *Syntactic Pattern Recognition; An Introduction.* Addison-Wesley (1978).

[34] J. Gregor, Data-driven inductive inference of finite-state automata. *Int. J. Pattern Recogn.* **8** (1994) 305–322.

[35] C. de la Higuera, A bibliographical study of grammatical inference. *Pattern Recogn.* **38** (2005) 1332–1348.

[36] C. de la Higuera, Current trends in grammatical inference. in *Advances in Pattern Recognition, Joint IAPR International Workshops SSPR+SPR'2000*, edited by F.J. Ferri *et al.* *Lect. Notes Comput. Sci.* **1876** (2000) 28–31.

[37] E. Jeong and C.-H. Hsu, Induction of integrated view for XML data with heterogenous DTDs. In *Proceedings of the 10th International Conference on Information and Knowledge Management CIKM*, ACM, (2001) 151–158.

[38] M. Kanazawa, Learnable Classes of Categorial Grammars. *Ph.D.*, CSLI (1998).

[39] B.B. Kimia, A.R. Tannenbaum and S.W. Zucker, Shapes, shocks, and deformations, I. *Int. J. Comput. Vision* **15** (1995) 189–224.

[40] B. Knobe and K. Knobe, A method for inferring context-free grammars. *Inform. Comput.* **31** (1976) 129–146.

[41] T. Knuutila, Inference of *k*-testable tree languages, in *Proc. IAPR Workshop on Structural and Syntactical Pattern Recognition.* World Scientific (1992) 109–120.

[42] T. Knuutila, How to invent characterizable methods for regular languages. in *4th Workshop on Algorithmic Learning Theory ALT'93*, edited by K.P. Jantke *et al. Lect. Notes Artif. Int.* **744** (1993) 209–222.

[43] T. Knuutila and M. Steinby, The inference of tree languages from finite samples: an algebraic approach. *Theoret. Comput. Sci.* **129** (1994) 337–367.

[44] S. Kobayashi and T. Yokomori, Learning approximately regular languages with reversible languages. *Theoret. Comput. Sci.* **174** (1997) 251–257.

[45] D. Kozen, On the Myhill-Nerode theorem for trees. *EATCS Bull.* **47** (1992) 170–173.

[46] H. Lieberman, editor. *Your Wish is My Command: Programming by Example.* Morgan Kaufmann (2001).

[47] D. López and S. España, Error correcting tree language inference. *Pattern Recogn. Lett.* **23** (2002) 1–12.

[48] D. López and I. Piñaga, Syntactic pattern recognition by error correcting analysis on tree automata, in *Advances in Pattern Recognition, Joint IAPR International Workshops SSPR+SPR'2000*, edited by F.J. Ferri *et al. Lect. Notes Comput. Sci.* **1876** (2000) 133–142.

[49] D. López, J.M. Sempere and P. García, Error correcting analysis for tree languages. *Int. J. Pattern Recogn.* **14** (2000) 357–368.

[50] D. López, J.M. Sempere and P. García, Inference of reversible tree languages. *IEEE T. Syst. Man Cy.* **34** (2004) 1658–1665.

[51] H.R. Lu and K.S. Fu, Error-correcting tree automata for syntactic pattern recognition. *IEEE T. Comput.* **27** (1978) 1040–1053.

[52] S. Matsumoto and T. Shoudai, Learning of ordered tree languages with height-bounded variables using queries. in *Algorithmic Learning Theory, 15th International Conference, ALT*, edited by S. Ben-David, J. Case and A. Maruoka. *Lect. Notes Comput. Sci.* **3244** (2004) 425–439.

[53] F. Neven, Automata, logic, and XML. In *Computer Science Logic; 16th International Workshop, CSL 2002*, edited by J. Bradfield. *Lect. Notes Comput. Sci.* **2471** (2002) 2–26.

[54] V. Radhakrishnan and G. Nagaraja, Inference of regular grammars via skeletons. *IEEE T. Syst. Man Cy.* **17** (1987) 982–992.

[55] V. Radhakrishnan and G. Nagaraja, Inference of even linear grammars and its application to picture description languages. *Pattern Recogn.* **21** (1988) 55–62.

[56] J.R. Rico-Juan, J. Calera-Rubio and R.C. Carrasco, Probabilistic *k*-testable tree languages. in *Grammatical Inference: Algorithms and Applications, 5th International Colloquium (ICGI 2000)*, edited by A.L. Oliveira. *Lect. Notes Artif. Int.* **1891** (2000) 221–228.

[57] J.R. Rico-Juan, J. Calera-Rubio and R.C. Carrasco, Stochastic *k*-testable tree languages and applications, in *Grammatical Inference: Algorithms and Applications, 6th International Colloquium, ICGI 2002*, edited by P. Adriaans, H. Fernau and M. van Zaanen. *Lect. Notes Artif. Int.* **2484** (2002) 199–212.

[58] G. Rozenberg and A. Salomaa, eds. *Handbook of Formal Languages, Volume III.* Springer, Berlin (1997).

[59] Y. Sakakibara, Learning context-free grammars from structural data in polynomial time. *Theoret. Comput. Sci.* **76** (1990) 223–242.

[60] Y. Sakakibara, Efficient learning of context-free grammars from positive structural examples. *Inform. Comput.* **97** (1992) 23–60.

[61] Y. Sakakibara and H. Muramatsu, Learning context-free grammars from partially structured examples, in *Grammatical Inference: Algorithms and Applications, 5th International Colloquium (ICGI 2000)*, edited by A.L. Oliveira. *Lect. Notes Artif. Int.* **1891** (2000) 229–240.

[62] J.M. Sempere and G. Nagaraja, Learning a subclass of linear languages from positive structural information. in *Proceedings of the Fourth International Colloquium on Grammatical Inference (ICGI-98)*, edited by V. Honavar and G. Slutski. *Lect. Notes Artif. Int.* **1433** (1998) 162–174.

[63] K. Siddiqi, A. Shakoufandeh, S. Dickinson and S. Zucker, Shock graphs and shape matching. *Int. J. Comput. Vision* **30** (1999) 1–24.

[64] B. Starkie, Developing Spoken Dialog Systems using Grammatical Inference. *Ph.D. Thesis*, The University of Newcastle (AUS) (2005).

[65] B. Starkie and H. Fernau, The Boisdale algorithm — an induction method for a subclass of unification grammar from positive data, in *Grammatical Inference: Algorithms and Applications; 7th International Colloquium ICGI*, edited by G. Paliouras and Y. Sakakibara. *Lect. Notes Artif. Int.* **3264** (2004) 235–247.

[66] Y. Takada and T.Y. Nishida, A note on grammatical inference of slender context-free languages, in *Proceedings of the Third International Colloquium on Grammatical Inference (ICGI-96): Learning Syntax from Sentences*, edited by L. Miclet and C. de la Higuera. *Lect. Notes Artif. Int.* **1147** (1996) 117–125.

[67] E. Tanaka and K.S. Fu, Error-correcting parsers for formal languages. *IEEE T. Comput.* **27** (1978) 605–616.

[68] J.L. Verdú-Mas, M.L. Forcada, R.C. Carrasco and J. Calera-Rubio, Tree k-grammar models for natural language modelling and parsing, in *Structural, Syntactic, and Statistical Pattern Recognition SSPR and SPR 2002*, edited by T. Caelli, A. Amin, R.P.W. Duin, M. Kamel and D. de Ridder. *Lect. Notes Comput. Sci.* **2396** (2002) 56–63.

[69] J.L. Verdú-Mas, R.C. Carrasco and J. Calera-Rubio, Parsing with probabilistic strictly locally testable tree languages. *IEEE T. Pattern Anal.*, **27** (2005) 1040–1050.

[70] H. Volger, Grammars with generalized contextfree rules and their tree automata. in *Proceedings of CLIN '99; Selected Papers* (1999) 223–233, see
`http://www-uilots.let.uu.nl/publications/clin1999/papers.html`

[71] T. Yokomori, Inductive inference of context-free languages based on context-free expressions. *Int. J. Comput. Math.* **24** (1988) 115–140.

[72] T. Yokomori, On learning systolic languages. in *Proceedings of the 3rd Workshop on Algorithmic Learning Theory (ALT '92)*, edited by K.P. Jantke, S. Doshita, K. Furukawa and T. Nishida. *Lect. Notes Artif. Int.* **743** (1992) 41–52.

[73] T. Yokomori, Polynomial-time identification of very simple grammars from positive data. *Theoret. Comput. Sci.* **298** (2003) 179–206.