Learning Tree Languages from Text

Henning Fernau

WSI-2001-19

Henning Fernau

Wilhelm-Schickard-Institut für Informatik
Universität Tübingen
Sand 13
D-72076 Tübingen
Germany

E-Mail: fernau@informatik.uni-tuebingen.de
Telefon: (07071) 29-77565
Telefax: (07071) 29-5061

ISSN 0946-3852

Learning tree languages from text

Henning Fernau

Wilhelm-Schickard-Institut für Informatik; Universität Tübingen
Sand 13; D-72076 Tübingen; Germany
email: fernau@informatik.uni-tuebingen.de

Abstract. We study the problem of learning regular tree languages from text. We show that the framework of function distinguishability as introduced in our ALT 2000 paper is generalizable from the case of string languages towards tree languages, hence providing 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 identified approximately.

1 Introduction

Grammatical inference mostly focussed on learning *string* languages, although there are many practical motivations for studying formally specified sets not being comprised of words, as well. For example, 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. Therefore, tree languages are quite important for linguistic studies. Moreover, derivation trees play an important role when studying the use of grammatical inference in connection with programming languages, see [3]. Furthermore, applications of tree languages and their inference to pattern recognition have been reported, see [12, Chapters 3.2 and 6.4]. Finally, trees can be interpreted equivalently as terms, so that the study of the inference of tree languages might also contribute to the learning of term algebras, a topic which seem to be touched only in [14, 16]. Hence, the study of the automatic inference of tree languages is well motivated. Besides [14, 16], the classes of regular tree languages presented in this paper are the first ones which are

characterized by well-defined restrictions on automata, an issue which is very important in grammatical inference, see [16, 15].

In some sense, the present paper can be seen as a continuation of Sakakibara's [22] on the inference of zero-reversible tree languages and of ours on the inference of distinguishable string languages [5, 8]: We will explore how to learn function distinguishable regular tree languages from text, a setting introduced by Gold [11].

In a nutshell, a function distinguishable regular tree language is given by a bottom-up deterministic tree automaton such that any occurring backward nondeterminism can be resolved by means of an "oracle" called distinguishing function. The zero-reversible languages considered by Sakakibara [22] are a special case where the oracle gives no information at all.

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 $y, x \in \mathbb{N}^*$, we write $y \leq x$ iff there is a $z \in \mathbb{N}^*$ with $x = y \cdot z$. "$y < x$" abbreviates: $y \leq x$ and $y \neq x$. As usual, $|x|$ denotes the length of the word 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 [21].

A *ranked alphabet* V is a finite set of symbols together with a finite relation called 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 $1 \leq j \leq i$. 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 $1 \leq i \leq n$. If $t(x) = A$, A is the *label* of x. Let $V^{\mathtt{t}}$ 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, $V^{\mathtt{t}}$ can be identified with the well-formed terms over V. 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 will be viewed as trees of size one and depth zero.

We are now going to define a catenation on trees. Let \$ be a new symbol, i.e., $\$ \notin V$, of rank 0. Let $V_{\$}^{\mathtt{t}}$ 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 $u \in V_{\$}^{\mathtt{t}}$ and $t \in (V^{\mathtt{t}} \cup V_{\$}^{\mathtt{t}})$, we define an operation $\#$ to replace the frontier node labelled with \$ of u by t according to

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

If $U \subseteq V_{\$}^{\mathtt{t}}$ and $T \subseteq (V^{\mathtt{t}} \cup V_{\$}^{\mathtt{t}})$, then $U\#T := \{u\#t \mid u \in U \wedge t \in T\}$. For $t \in V^{\mathtt{t}}$ 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 $T \subseteq V^{\mathtt{t}}$. Furthermore, for any $t \in V^{\mathtt{t}}$ and any tree language $T \subseteq V^{\mathtt{t}}$, the *quotient* of T and t is defined as:

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

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 $V^{\mathtt{t}}$ by letting

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

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 case of string automata, 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.

The notions of isomorphic automata and (state subset induced) subautomata can be easily carried over from the well-known case of string processing automata to tree automata. 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*.

We need four special constructions of tree automata in our treatment:

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: $Q = \mathrm{ST}(I_+) \setminus V_0$, $F = I_+$, $\delta_k(f, u_1, \ldots, u_k) = f(u_1, \ldots, u_k)$ whenever $u_1, \ldots, u_k \in Q \cup V_0$ and $f(u_1, \ldots, u_k) \in Q$. Obviously, $T(Bs(I_+)) = I_+$.

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.

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 $p_i = (q_i, q'_i)$.

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 [5] 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.

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 $V^{\mathfrak{t}} \to Q_\delta$. 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 $x, y \in V^{\mathfrak{t}}$ with $\bar{\delta}(x) = \bar{\delta}(y) = q$, we have $\delta(x) = \delta(y)$. (In other words, for $q \in Q$, $\delta(q) := \delta(x)$ for some x with $\bar{\delta}(x) = q$ is well-defined.)

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 $\delta(q_1) \neq \delta(q_2)$.

In some sense, A is "backward deterministic" with the aid of δ. 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.

Observe that trees with whose domain is contained in $\{1\}^*$ correspond to usual strings, and the notion of distinguishability obtained in this way is the same as the one introduced in [5]. The trivial tree automaton with just one state leads to a slight generalization of the zero-reversible tree automata studied by Sakakibara [22]; in Sakakibara's model, only one symbol σ_k for each arity $k \geq 1$ was permitted as label of the interior nodes.

As a further simple example of distinguishing function, consider the function *Ter* defined by $Ter(a) = \{a\}$ 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 [19]. As exhibited in [5] in the string case, several other similar distinguishing functions can be defined which also yield immediate analogues in the tree case. Analogues to k-terminal distinguishable languages [6] and to the k-reversible languages [2] can be defined by means of the notions of k-roots, k-forks and k-subtrees as defined in [14].

If we restrict ourselves to *Ter* applied to derivation trees of even linear grammars (with trivial labels for all interior nodes), we basically arrive at a variant of the terminal distinguishable even linear languages as discussed in [7] in more details. Speaking more generally, 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.

Remark 1 Any subautomaton of a δ-distinguishable tree automaton A is δ-distinguishable. □

Lemma 1 *Let $A = (Q, V, \bar{\delta}, F)$ be a δ-distinguishable tree automaton. Let $u \in V_{\$}^{\mathtt{t}}$ and $\{t_1, t_2\} \subseteq V^{\mathtt{t}}$. If $\{u\#t_1, u\#t_2\} \subseteq T(A)$ and if $\delta(t_1) = \delta(t_2)$, then $\bar{\delta}(t_1) = \bar{\delta}(t_2)$.*

Proof.

First observe that, since $\delta(t_1) = \delta(t_2)$, we have $\delta(u\#t_1) = \delta(u\#t_2)$ for all $u \in V_{\$}^{\mathtt{t}}$ and $t_1, t_2 \in V^{\mathtt{t}}$.

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 $u \in V_{\$}^{\mathtt{t}}$ 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 $s_1, \ldots, s_{k-1} \in V^{\mathtt{t}}$ and some $u' \in V_{\$}^{\mathtt{t}}$ having the node labelled \$ at level $h - 1$. The induction hypothesis yields: if A accepts both

$$u\#t_1 = u'\#f(s_1, \ldots, s_{i-1}, t_1, s_i, \ldots, s_{k-1})$$

and

$$u\#t_1 = u'\#f(s_1, \ldots, s_{i-1}, t_2, s_i, \ldots, s_{k-1}),$$

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,

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

so that condition 2b. for δ-distinguishable tree automata yields $q_1 = q_2$. □

Lemma 2 *Let $A = (Q, V, \bar{\delta}, F)$ be a δ-distinguishable tree automaton. Let $\{u_1, u_2\} \subseteq V_{\$}^{\mathtt{t}}$ and $\{t, t'\} \subseteq V^{\mathtt{t}}$. If $\{u_1\#t, u_2\#t\} \subseteq T(A)$ and if $\delta(t) = \delta(t')$, 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 1 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. □

Let $T \subseteq V^{\mathtt{t}}$ 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 3 (Characterization theorem) *Let $\delta : V^{\mathtt{t}} \to Q_\delta$ be a distinguishing function. Then, the following conditions are equivalent for a regular tree language $T \subseteq V^{\mathtt{t}}$:*

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

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

3. $\forall t_1, t_2 \in V^{\mathtt{t}}\ \forall u, v \in V_{\$}^{\mathtt{t}} : \ (\{u\#t_1, v\#t_2\} \subseteq T \wedge \delta(t_1) = \delta(t_2)) \Rightarrow (u\#t_1 \in T \iff v\#t_2 \in T)$.

4. $\forall t_1, t_2 \in V^{\mathtt{t}}\ \forall u, v \in U_T(t_1) : \ \delta(t_1) = \delta(t_2) \Rightarrow (u \in U_T(t_2) \iff v \in U_T(t_2))$.

5. *$A(T, \delta)$ is δ-distinguishable.*

6. $\forall t_1, t_2 \in V^{\mathtt{t}}\ \forall u \in V_{\$}^{\mathtt{t}} : \ (\{u_1\#t, u_2\#t\} \subseteq T \wedge \delta(t_1) = \delta(t_2)) \Rightarrow U_T(t_1) = U_T(t_2)$.

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

We are going to show 4. $\rightarrow$ 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 $1 \leq i < k$. Since $A(T, \delta)$ is stripped, there are a $\hat{u} \in V_{\$}^{\mathtt{t}}$ and $s_1, \ldots, s_{k-1} \in V^{\mathtt{t}}$ 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 $U_T(t_1) = U_T(t_2)$. $\square$

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

Proof. According to Remark 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 $\delta(q_1) = \delta(q_2)$,

$$\begin{aligned}&\{u \in V_{\$}^{\mathfrak{t}} \mid \exists t \in V^{\mathfrak{t}} \setminus V_0 : \bar{\delta}'(u\#q_1) \in F'\}\\ &\quad = \{u \in V_{\$}^{\mathfrak{t}} \mid \exists t \in V^{\mathfrak{t}} \setminus V_0 : \bar{\delta}'(u\#q_2) \in F'\}.\end{aligned}$$

implies that $q_1 = q_2$, since then, the mapping

$$q \mapsto (U_{T(A)}(t), \delta(q))$$

for some $t \in V^{\mathfrak{t}}$ with $\bar{\delta}'(t) = q$ will supply the required isomorphism.

Since A' is stripped, there are $t_1, t_2 \in V^{\mathfrak{t}}$ and $u \in V_{\$}^{\mathfrak{t}}$, $q_1 = \bar{\delta}'(t_1)$, $q_2 = \bar{\delta}'(t_2)$ 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 1 to show the result. □

4 Inferrability

The learning model we use is *identification in the limit from positive samples* as proposed by Gold [11], 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 $E(1), E(2), \ldots$, 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.

According to a theorem due to Angluin [1, Theorem 1], a language class $\mathcal{L}$ is inferrable if any language $L \in \mathcal{L}$ has a *characteristic sample*, i.e., a finite subset $\chi(L) \subseteq L$ such that L is a minimal language from $\mathcal{L}$ containing $\chi(L)$.

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

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

where $u(q) \in V_{\$}^{\mathtt{t}}$ and $t(q) \in V^{\mathtt{t}} \setminus V_0$ 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 (Characteristic sample) *For each A_δ and each $T \in \delta$-DT, $\chi(T,\delta)$ is a characteristic sample 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 characteristic sample, 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 1 yields

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

because

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

This completes the induction. □

5 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 $T \in \delta$-DT to be identified) and finds a minimal language $T' \in \delta$-DT which contains I_+. Of course, Theorem 5 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 [22].

Our merging state inference algorithm δ-`Ident` for δ-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. 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 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 $\delta(q_1) = \delta(q_2)$.

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 $1 \leq j \leq k$, either $p_j = p'_j \in V_0$ or $B(p_j, \pi_i) = B(p'_j, \pi_i)$.

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 $1 \leq \ell \leq k$ such that, for all $1 \leq j \leq k$ with $i \neq \ell$, 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 (δ-`Ident`)
Input: a nonempty positive sample $I_+ \subseteq V^{\mathtt{t}}$.
Output: $A(T, \delta)$, where T is a minimal δ-distinguishable tree language containing I_+.
*** Initialization
Let $A_0 = (Q = \mathrm{ST}(T) \setminus V_0, V, \bar{\delta}, F = I_+) = Bs(I_+)$.
Let π_0 be the trivial partition of Q.
Let LIST contain all unordered pairs $\{q, q'\}$ of final states of Q such that $q \neq q'$ and $\delta(q) = \delta(q')$.
Let $i := 0$.
*** Merging
While LIST$\neq \emptyset$ 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 $B_1 \neq B_2$, then begin
        Let π_{i+1} be π_i with B_1 and B_2 merged.

Resolve newly produced determinism conflicts and update LIST.

Resolve newly produced backward determinism conflicts and 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 [22] or by keeping track of the forward and backward transition functions of the automata A_i, as detailed in [5] in the case of string language inference. In either case, we obtain algorithms with polynomially bounded update times, see [22] for the definitions.

Moreover, it is possible to design so-called *incremental versions* of the algorithms, where basically 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 7 *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*[1] *$U_T(q_1) = U_T(q_2)$ and if $\delta(q_1) = \delta(q_2)$. 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

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

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

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,

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

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

Theorem 8 *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 $A_0 = Bs(I_+)$. 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 [2, Lemma 25] and [22, Lemma 13]. □

Theorem 9 *In the notations of the Theorem 8, $T(A_f)$ is a minimal δ-distinguishable language containing I_+.*

Proof. Theorem 8 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 7. This Lemma shows that

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

By Remark 1, $\pi^{-1}A_0$ is δ-distinguishable, because $A(T,\delta)$ is δ-distinguishable due to Theorem 3. Theorem 8 yields that π_f refines π, so that

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

Remark 2 Up to now, in accordance with the definition of a characteristic sample, 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.

Theorem 10 (Correctness of δ-`Ident`) *If $T \in \delta$-DT is enumerated as input to the algorithm δ-`Ident`, it converges to the δ-canonical automaton $A(T,\delta)$.*

Proof. At some point N of the enumeration process, the characteristic sample $\chi(T,\delta)$ will have been given to δ-`Ident`. By combining Theorems 5 and 9, for all $n \geq N$ and all automata A_n output by δ-`Ident`, we have $T(A_n) = T$. The argument of Theorem 9 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. $\square$

We finally remark that the performance of the general algorithm δ-`Ident` sketched above depends on the size of A_δ (since the characteristic sample $\chi(T,\delta)$ we defined above depends on this size) and is in this sense "scalable", since "larger" A_δ permit larger language families to be identified. More precisely, we can show:

Proposition 11 *If A_δ is a homomorphic image of A_γ, then δ-DT $\subseteq \gamma$-DT.* $\square$

Appendix: 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 [17].

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.

Remark 3 If $\mathcal{L}$ is closed under intersection, then $\bar{\mathcal{L}}L$ is uniquely defined.

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 *characteristic sample* 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 $T \subseteq V^{\mathtt{t}}$ *pseudo-δ-distinguishable* iff, for all $t_1, t_2 \in V^{\mathtt{t}}$ with $\delta(t_1) = \delta(t_2)$ and for all $u \in V^{\mathtt{t}}_{\$}$, we have $U_T(t_1) = U_T(t_2)$ whenever $\{u\#t_1, u\#t_2\} \subseteq T$. By our characterization theorem, $T \in \delta$-DT iff T is a pseudo-δ-distinguishable and regular tree language.

Immediately from the definition, we may conclude:

Proposition 12 *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 4 If $T \subseteq V^{\mathtt{t}}$ is a regular tree language and if $\delta : V^{\mathtt{t}} \to Q_\delta$ 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 $T \subseteq V^{\mathtt{t}}$ be some tree language. For any integer i, we will define $R_\delta(i,T)$ as follows:

1. $R_\delta(0,T) = T$ and
2. $R_\delta(i,T) = R_\delta(i-1,T) \cup \{\, u\#t_2 \mid u\#t_1, u'\#t_1, u'\#t_2 \in R_\delta(i-1,T) \wedge \delta(t_1) = \delta(t_2) \,\}$ 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 13 *For any tree language T and any distinguishing function δ, $R_\delta(T)$ is the smallest pseudo-δ-distinguishable language containing T.* □

Lemma 14 *Let $T \subseteq V^{\mathtt{t}}$ 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 $u \in V_{\$}^{\mathtt{t}}$ 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 13, $U_{R_\delta(T)}(t_1) = U_{R_\delta(T)}(t_2)$. □

Lemma 15 *Let $T \subseteq V^{\mathtt{t}}$ 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 $u_1, u_2 \in V_{\$}^{\mathfrak{t}}$ and $t, t' \in V^{\mathfrak{t}}$ 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 $u' \in V_{\$}^{\mathfrak{t}}$. 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 $U_{R_\delta(T)}(t_2) = U_{R_\delta(T)}(u_1\#t) = U_{R_\delta(T)}(t_1)$. □

By a reasoning completely analogous to [17], we may conclude:

Theorem 16 *For any distinguishing function δ, the class δ-DT has the u.b.a.p. with respect to the class of regular tree languages.* □

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 4.

Theorem 17 *For any distinguishing function δ, the class of regular languages is upper-best approximately identifiable in the limit from positive data by δ-DT.* □

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 3.

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.

6 Discussion and Prospects

For a variety of regular tree language classes, we showed in which way they can be inferred efficiently. 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. In the long version of this paper, we show that every regular tree language can be approximated in a well-defined manner by languages from δ-DT for any chosen A_δ, see [8, 17] for the string case.

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 [4, 10, 12, 13, 16, 18, 25, 26, 27]. 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 [24] for a short exposition. Especially, we will employ our results for devising learning algorithms for linear languages, which will be a complimentary approach to the ones detailed in [20, 23], as well as in the long version of [7]. 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 [9]. These connections will be investigated in the near future.

Finally, we will provide an publicly accessible implementation of our tree language inference algorithms.

Acknowledgments: We are grateful for stimulating discussions with our colleagues S. Kobayashi and J. M. Sempere.

References

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

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

[3] S. Crespi-Reghizzi, M. A. Melkanoff, and L. Lichten. The use of grammatical inference for designing programming languages. *Communications of the ACM*, 16:83–90, 1972.

[4] L. F. Fass. Learning context-free languages from their structured sentences. *SIGACT News*, 15(3):24–35, 1983.

[5] H. Fernau. Identification of function distinguishable languages. In H. Arimura, S. Jain, and A. Sharma, editors, *Proceedings of the 11th International Conference Algorithmic Learning Theory ALT 2000*, volume 1968 of *LNCS/LNAI*, pages 116–130. Springer, 2000.

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

[7] H. Fernau. Learning of terminal distinguishable languages. In *Proc. AMAI 2000*, 2000. Available through: `http://rutcor.rutgers.edu/~amai/AcceptedCont.htm`.

[8] H. Fernau. Approximative learning of regular languages. In P. Ružička, editor, *SOFSEM'01*, LNCS. Springer, 2001. To appear.

[9] H. Fernau and J. M. Sempere. Permutations and control sets for learning non-regular language families. In A. L. Oliveira, editor, *Grammatical Inference: Algorithms and Applications, 5th International Colloquium (ICGI 2000)*, volume 1891 of *LNCS/LNAI*, pages 75–88. Springer, 2000.

[10] C. C. Florêncio. Consistent identification in the limit of any of the classes k-valued is NP-hard. In *Proceedings of the Conference on Logical Aspects of Computational Linguistics (LACL 2001)*, volume 2099 of *LNCS/LNAI*, pages 125–138. Springer, 2001.

[11] E. M. Gold. Language identification in the limit. *Information and Control (now Information and Computation)*, 10:447–474, 1967.

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

[13] V. M. Jiménez and A. Marzal. Computation of the n best parse trees for weighted and stochastic context-free grammars. In F. J. Ferri et al., editors, *Advances in Pattern Recognition, Joint IAPR International Workshops SSPR+SPR'2000*, volume 1876 of *LNCS*, pages 183–192, 2000.

[14] T. Knuutila. How to invent characterizable methods for regular languages. In K. P. Jantke et al., editors, *4th Workshop on Algorithmic Learning Theory ALT'93*, volume 744 of *LNCS/LNAI*, pages 209–222, 1993.

[15] T. Knuutila. Inductive inference from positive data: from heuristic to characterizing methods. In L. Miclet and C. de la Higuera, editors, *Proceedings of the Third International Colloquium on Grammatical Inference (ICGI-96): Learning Syntax from Sentences*, volume 1147 of *LNCS/LNAI*, pages 22–47, Berlin, September25–27 1996. Springer.

[16] T. Knuutila and M. Steinby. The inference of tree languages from finite samples: an algebraic approach. *Theoretical Computer Science*, 129:337–367, 1994.

[17] S. Kobayashi and T. Yokomori. Learning approximately regular languages with reversible languages. *Theoretical Computer Science*, 174(1–2):251–257, 1997.

[18] D. López and I. Piñaga. Syntactic pattern recognition by error correcting analysis on tree automata. In F. J. Ferri et al., editors, *Advances in Pattern Recognition, Joint IAPR International Workshops SSPR+SPR'2000*, volume 1876 of *LNCS*, pages 133–142, 2000.

[19] V. Radhakrishnan and G. Nagaraja. Inference of regular grammars via skeletons. *IEEE Transactions on Systems, Man and Cybernetics*, 17(6):982–992, 1987.

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

[21] G. Rozenberg and A. Salomaa, editors. *Handbook of Formal Languages, Volume III*. Berlin: Springer, 1997.

[22] Y. Sakakibara. Efficient learning of context-free grammars from positive structural examples. *Information and Computation*, 97(1):23–60, March 1992.

[23] J. M. Sempere and G. Nagaraja. Learning a subclass of linear languages from positive structural information. In V. Honavar and G. Slutski, editors, *Proceedings of the Fourth International Colloquium on Grammatical Inference (ICGI-98)*, volume 1433 of *LNCS/LNAI*, pages 162–174, Berlin, July 1998. Springer.

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

[25] T. Yokomori. Inductive inference of context-free languages based on context-free expressions. *International Journal of Computer Mathematics*, 24:115–140, 1988.

[26] T. Yokomori. Polynomial-time learning of very simple grammars from positive data. In *Proc. 4th Annu. Workshop on Comput. Learning Theory*, pages 213–227, San Mateo, CA, 1991. Morgan Kaufmann.

[27] T. Yokomori. On learning systolic languages. In K. P. Jantke, S. Doshita, K. Furukawa, and T. Nishida, editors, *Proceedings of the 3rd Workshop on Algorithmic Learning Theory (ALT '92)*, volume 743 of *LNCS/LNAI*, pages 41–52. Springer, October 1992.