Acta Cybernetica **19** (2009) 249–274.

MAT Learners for Recognizable Tree Languages and Tree Series

Frank Drewes*

Abstract

We review a family of closely related query learning algorithms for unweighted and weighted tree automata, all of which are based on adaptations of the minimal adequate teacher (MAT) model by Angluin. Rather than presenting new results, the goal is to discuss these algorithms in sufficient detail to make their similarities and differences transparent to the reader interested in grammatical inference of tree automata.

Keywords: algorithmic learning, grammatical inference, tree automaton, tree language, tree series

1 Introduction

This article discusses a family of algorithms for grammatical inference of unweighted and weighted tree automata. Traditionally, the area of grammatical inference studies the problem of learning a formal (string) language L by automatically inferring an explicit automata-theoretic or grammatical description A of L from examples or some other type of information about L. In other words, the aim is to come up with a *learner*, an algorithm that exploits a source S of information about L in order to construct A. Different so-called learning models are obtained by specifying (a) which source S of information the learner is provided with, (b) how the learner gets access to this information, and (c) what the exact criterion of success is.

The three most well-established categories of learning models in grammatical inference are Gold's *learning from examples with identification in the limit* [23], Valiants *probably approximately correct (PAC) learning* [39], and Angluin's *query learning* [4].

Here, we focus on query learning. This model, which is also called active learning, gives the learner access to a *teacher*, an oracle able to answer certain types of queries. Suppose that L is a regular string language and the goal is to construct a corresponding finite-state automaton A. The most well-studied type of teacher is the so-called minimal adequate teacher (MAT) [3]. The MAT will answer two different sorts of queries regarding L. The first is the *membership query*, in which

*Umeå University, 906 87 Umeå, Sweden, E-mail: drewes@cs.umu.se

the learner passes the teacher a string u, and the teacher checks whether $u \in L$. In the second type of query, the *equivalence query*, the learner passes the teacher a proposed automaton A', and the teacher checks whether A' correctly describes L. If so, A' is accepted and the learning process terminates. Otherwise, the teacher returns a counterexample to the learner, i.e., an element of the symmetric difference of L and the language described by A'.

A learning model closely related to MAT learning is learning from representative samples and membership queries [2]. Here, the learner has access to a weaker teacher who will only answer membership queries. To compensate for the lack of equivalence queries, the learner is initially provided with a *representative sample*, a set of strings in L, such that every transition of A is used at least once when processing the strings in the sample.

Here, we want to consider algorithms for learning unweighted and weighted tree automata rather than ordinary finite-state automata. Why would such extensions be of interest? Apart from theoretical curiosity and the fact that tree languages play an important role in many application areas, motivation is provided by the fact that almost all results regarding the inference of context-free languages are negative. However, recognizable (or regular) tree languages may be seen as context-free languages whose strings are enriched with explicit structural information. Thus, positive results for grammatical inference of recognizable tree languages make it possible to learn context-free languages if the learner is provided with the additional structural information (cf. [32]).

If we want to use the learning models described above, they have to be adapted. This can be done in a straightforward way. In membership queries, trees rather than strings must be passed to the teacher, and in equivalence queries, tree automata of the type considered must be checked by the teacher. Similarly, a representative sample is now a set of trees. Moreover, in the weighted case, membership queries must be replaced with coefficient queries (i.e., the teacher returns the coefficient of the tree passed, with respect to the sought tree series), and the counterexample returned as an answer to an equivalence query must be a tree for which the proposed automaton computes a coefficient that differs from the one it should compute.

The appropriateness of the MAT model is not undisputed. Obviously, the assumption of having access to an oracle able to answer equivalence queries is strong and may be considered unrealistic. Moreover, it has been argued in [6] that membership queries are oversimplified and should be replaced by a type of query yielding a more informative result, e.g., so-called correction queries. To a certain extent, this criticism is certainly justified. In particular, future research should continue to explore reasonable alternative settings. However, in the author's opinion, this does not diminish the value of the algorithms reviewed in the next two sections. In general, one should keep in mind that the learning models considered are idealizations that – as always in Theoretical Computer Science – trade realism for mathematical elegance and simplicity. Having read this paper, the reader who has never seen these algorithms before will hopefully acknowledge that they are based on beautiful formal reasonings. In particular, they make elegant use of Myhill-Nerode-like characterizations of the tree languages and series to be learned.

Since grammatical inference is an inherently difficult goal, there seem to be only two ways to achieve positive results whose correctness can formally be proved. One either has to simplify the goal, e.g., by placing severe restrictions on the concepts to be learned, or give the learner access to a rather powerful source of information, such as a MAT. Clearly, both approaches have their advantages and disadvantages. This paper focuses on the second, because we are interested in the grammatical inference of unrestricted recognizable tree languages and tree series. For this task, there do not yet seem to exist many algorithms other than the ones discussed here. Moreover, these algorithms are all very closely related to each other, which makes them interesting (in the authors opinion), because it indicates that they are based on "robust" ideas worth being explored.

As mentioned above, the MAT model is a formal idealization. Therefore, one cannot expect that learning algorithms based on a formal setting such as the MAT model can directly be applied to learning tasks in, say, natural language processing. However, it may be an interesting goal to pursue in future research to identify practical scenarios in which the teacher can be simulated by, e.g., statistical methods. Of course, such an approach would no more be guaranteed to yield an affirmatively correct answer, but it may perform sufficiently well in practice – and hopefully much better than an ad-hoc approach. In fact, it may then be a theoretically interesting and practically well-motivated question under which assumptions imperfect teachers give rise to reasonably good results, e.g., in a PAC-like setting.

From what has been said above, it should be clear that this paper is not a general survey of the large field of grammatical inference. In fact, it does not even attempt to cover the subarea of grammatical inference of tree languages and tree series. Readers who wish to obtain a general overview of grammatical inference are referred to the various existing survey papers [1, 13, 21, 28, 34]. Readers interested in inference of tree languages, using other methods and models than the ones discussed here, may also wish to have a look at [33, 27, 20, 29].

In the next section, learners for recognizable tree languages based on (variations of) the MAT model are discussed. In Section 3, we discuss generalizations of these algorithms, that learn recognizable tree series. The paper concludes with some final remarks in Section 4.

2 Learners for Recognizable Tree Languages

As mentioned above, grammatical inference is the task to construct an automaton or a grammar describing a language L, given certain information about L. For the moment being, let us consider the string case. Suppose that we are interested in learning a class $\mathcal{L}(\mathcal{A})$ of string languages, where $\mathcal{A}$ is a class of automata, and $\mathcal{L}(\mathcal{A}) = \{L(A) \mid A \in \mathcal{A}\}$ is the class of languages generated by $\mathcal{A}$. The task of the learner is to construct, for a given language $L \in \mathcal{L}(\mathcal{A})$, an automaton $A \in \mathcal{A}$ with $L(A) = L$. For this, the learner needs to have access to information regarding L. Here, we mainly want to study the case where this information is provided by a MAT [3]. This oracle that will (correctly) answer two different sorts of queries:

Membership query Given a string $u \in \Sigma^*$ (provided by the learner), the membership query $\mathsf{member}(u)$ will be answered by returning 1 if $u \in L$, and 0 if $u \notin L$. (Thus, member computes the characteristic function of L; see below.)

Equivalence query Given an automaton $A \in \mathcal{A}$ (provided by the learner), the equivalence query $\mathsf{eqQuery}(A)$ will be answered by returning the special token $\perp$ if $L(A) = L$. Otherwise, a counterexample $u \in L(A) \triangle L$ is returned, where the operator $\triangle$ yields the symmetric difference of sets.

The learner L_* proposed in [3] learns the class of regular languages from a MAT in polynomial time, where $\mathcal{A}$ is the set of total deterministic finite-state automata. It makes use of the Myhill-Nerode theorem for regular languages to construct the canonical finite-state automaton recognizing L.[1] To achieve this goal, the learner maintains a so-called observation table, which can be seen as an adapted version of the *state characterization matrix* introduced by Gold [24] for identifying regular languages from positive and negative examples in the limit. In the following, we discuss extensions and variations of L_* that learn tree automata.

Let us first recall a few basic definitions and facts. A ranked alphabet Σ is a finite set of ranked symbols (f, k), where f is a symbol and $k \in \mathbb{N}$, its rank, is a non-negative integer. We let $\Sigma_{(k)} = \{(f, l) \in \Sigma \mid l = k\}$. In the following, a ranked symbol (f, k) will simply be denoted by f, or by $f^{(k)}$ if it is necessary to specify its rank. The set T_Σ of trees over Σ is the smallest set of formal expressions such that $f[t_1, \ldots, t_k] \in \mathrm{T}_\Sigma$, for every $f^{(k)} \in \Sigma$ ($k \in \mathbb{N}$) and all $t_1, \ldots, t_k \in \mathrm{T}_\Sigma$. Here, the brackets and commas are special symbols not in Σ. For $k = 0$, the tree $f[]$ may simply be denoted by f. For a set T of trees, we let $\Sigma(T)$ denote the set of all trees of the form $f[t_1, \ldots, t_k]$, where $f^{(k)} \in \Sigma$ and $t_1, \ldots, t_k \in T$. The set of all subtrees of a tree $t = f[t_1, \ldots, t_k]$ is given by $\mathrm{subtrees}(t) = \{t\} \cup \bigcup_{i=1}^{k} \mathrm{subtrees}(t_i)$. A *tree language* is a set $L \subseteq \mathrm{T}_\Sigma$. The characteristic function of L is denoted by χ_L. Thus, for $t \in \mathrm{T}_\Sigma$, $\chi_L(t) = 1$ if $t \in L$, and $\chi_L(t) = 0$, otherwise.

Definition 2.1. *A deterministic bottom-up finite tree automaton (fta) is a tuple $A = (\Sigma, Q, \delta, F)$ consisting of a ranked alphabet Σ, a ranked alphabet Q of states such that $Q = Q_{(0)}$, a transition table δ, and a set $F \subseteq Q$ of final states. The transition table is a partial function $\delta\colon \Sigma(Q) \to Q$. This extends to trees in the canonical way, yielding a partial function $\delta\colon \mathrm{T}_\Sigma \to Q$. A tree $t \in \mathrm{T}_\Sigma$ is accepted by A if $\delta(t) \in F$. The language recognized by A consists of all trees accepted by A, i.e., $L(A) = \{t \in \mathrm{T}_\Sigma \mid \delta(t) \in F\}$, and is called a recognizable (or regular) tree language.*

As usual, an fta is said to be total if the transition table δ is a total function. We note that δ can also be regarded as a set of transitions $f[q_1, \ldots, q_k] \to q$, where $\delta(f[q_1, \ldots, q_k]) = q$. In other words, a transition is a pair in $\Sigma(Q) \times Q$. Since we consider only the deterministic case, transitions have pairwise distinct left-hand sides $f[q_1, \ldots, q_k]$. However, unless the fta is total, not all left-hand sides need to be present.

[1] It may be interesting to note that the class of regular languages is not learnable in polynomial time from membership or equivalence queries alone [5]. This provides some justification for calling the oracle above a *minimal adequate teacher*.

The first extension of L_* to so-called skeletal tree languages[2] was given by Sakakibara [32]. Let us have a look at this learner, which we may call $\mathrm{L}_*^{\mathrm{tfta}}$. It constructs the canonical total fta recognizing the target language L. In the presentation below, we drop the restriction to skeletal tree languages, since it is not important for the correctness of $\mathrm{L}_*^{\mathrm{tfta}}$. In fact, this slight generalization has the advantage that L_* may be seen as a special case of $\mathrm{L}_*^{\mathrm{tfta}}$, by identifying a string $a_1 \cdots a_n$ with the monadic tree $a_n[\cdots a_1[\epsilon]\cdots]$. (The string case can, of course, even be simulated using skeletal trees, but this seems to require the use of a representation that maps strings to trees in a non-surjective way, for example, by representing $u = a_1 \cdots a_n$ as $\mathrm{tree}(u) = *[\cdots *\,[a_1, a_2], \cdots a_n]$. As a consequence, if A is a deterministic finite-state string automaton, an fta recognizing $\{\mathrm{tree}(u) \mid u \in L(A)\}$ will in general contain more states than A.)

As indicated in the introduction, the idea behind L_* and all its descendants is to construct an automaton by exploiting the Myhill-Nerode congruence of the target language. Let $\Box^{(0)} \notin \Sigma$ be a special symbol, and let C_Σ be the set of all trees in $\mathrm{T}_{\Sigma\cup\{\Box\}}$ with exactly one occurrence of $\Box$, called *contexts over* Σ. The concatenation $c \cdot t$ of $c \in \mathrm{C}_\Sigma$ with $t \in \mathrm{T}_\Sigma \cup \mathrm{C}_\Sigma$ is the tree obtained from c by replacing $\Box$ with t. Now, the Myhill-Nerode congruence $\equiv_L$ on T_Σ is given by

$$t \equiv_L t' \text{ if and only if } \chi_L(c \cdot t) = \chi_L(c \cdot t') \text{ for all } c \in \mathrm{C}_\Sigma.$$

It is well known that $\equiv_L$ is of finite index (i.e., its congruence classes are finite in number) if and only if L is recognizable. The canonical (total) fta A_L^{t} recognizing L can be obtained as usual, by taking the congruence classes $[t]_{\equiv_L}$, $t \in \mathrm{T}_\Sigma$, as states and defining $\delta(f[[t_1]_{\equiv_L}, \ldots, [t_k]_{\equiv_L}]) = [f[t_1, \ldots, t_k]]_{\equiv_L}$. By the congruence property, the choice of the representatives $t_1, \ldots, t_k$ does not matter. A state $[t]_{\equiv_L}$ is final if $t \in L$.

Now, let us define an equivalence relation $\sim_C$ on T_Σ by replacing C_Σ in the definition of $\equiv_L$ with a finite set of contexts. For $C \subseteq \mathrm{C}_\Sigma$, let $t \sim_C t'$ if and only if, for all $c \in C$, $\chi_L(c \cdot t) = \chi_L(c \cdot t')$. By definition, $\equiv_L = \sim_{\mathrm{C}_\Sigma}$. Moreover, if $\equiv_L$ is of finite index, there is a finite set C of contexts such that $\equiv_L = \sim_C$. The learners based on L_* (and, in fact, several other learners as well), discover such a set C and construct the target automaton from it. Note that, for arbitrary $C \subseteq \mathrm{C}_\Sigma$, $\sim_C$ is not necessarily a congruence.

Following the same idea as L_*, the learner $\mathrm{L}_*^{\mathrm{tfta}}$ uses membership and equivalence queries to discover trees representing different congruence classes, together with suitable separating contexts. The data structure used for this is the previously mentioned observation table. Its rows are indexed by the trees in $\Sigma(S)$, for a finite set $S \subseteq \mathrm{T}_\Sigma$, and its columns are indexed by contexts from a finite set $C \subseteq \mathrm{C}_\Sigma$ containing $\Box$. The cell in row t and column c contains the value $\chi_L(c \cdot t)$, which the learner obtains by asking a membership query. For $t \in \Sigma(S)$, if the observation table Ω in question is clear from the context, we let $\langle t \rangle$ denote the C-indexed vector given by the row of t in Ω. For a set $T \subseteq \Sigma(S)$, we let $\langle T \rangle = \{\langle t \rangle \mid t \in T\}$.

[2] A tree language L is *skeletal* if $L \subseteq \mathrm{T}_\Sigma$ for a ranked alphabet Σ with $|\Sigma_{(k)}| \leq 1$ for all $k \geq 1$.

We require that S be *subtree-closed*, meaning that $s_1, \ldots, s_k \in S$ for every tree $f[s_1, \ldots, s_k] \in S$. In other words, $S \subseteq \Sigma(S)$, which means that Ω even contains rows for the trees $s \in S$. Note that, for $t, t' \in \Sigma(S)$, $\langle t \rangle \neq \langle t' \rangle$ implies $t \not\equiv_L t'$, because $\sim_C \supseteq \equiv_L$. Moreover, as observed above, there exists an observation table for which the converse holds as well. The aim of the learner is to build such an observation table.

During its run, the learner $\mathrm{L}_*^{\mathrm{tfta}}$ repeatedly uses the tentative observation table Ω it has built in order to construct a total fta A_Ω consistent with the observations in Ω. This fta is passed to the teacher, and if it is not approved, then the counterexample received is used to enlarge Ω. To be able to construct A_Ω from Ω, the following two properties are needed.

1. Ω is *closed*, meaning that $\langle t \rangle \in \langle S \rangle$, for every $t \in \Sigma(S)$.
2. Ω is *consistent*. To define this property, let $\Sigma_\square(S) = \mathrm{C}_\Sigma \cap \Sigma(S \cup \{\square\})$. The observation table Ω is consistent if $\langle c \cdot s \rangle = \langle c \cdot s' \rangle$, for all $c \in \Sigma_\square(S)$ and all $s, s' \in S$ with $\langle s \rangle = \langle s' \rangle$. Note that $\langle c \cdot s \rangle \neq \langle c \cdot s' \rangle$ would mean that there is a $d \in C$ such that $\chi_L((d \cdot c) \cdot s) \neq \chi_L((d \cdot c) \cdot s')$, i.e., $d \cdot c$ would be a context witnessing that $s \not\equiv_L s'$, despite the fact that $\langle s \rangle = \langle s' \rangle$. Moreover, the addition of $d \cdot c$ to C would make the rows of s and s' different, thus resolving the inconsistency.

If Ω is both closed and consistent, A_Ω can be defined by a construction similar to the construction of the canonical fta from $\equiv_L$. The set of states is $\langle S \rangle$, a state $\langle s \rangle$ being final if $s \in L$. For every tree $t = f[s_1, \ldots, s_k] \in \Sigma(S)$, we let $\delta(f[\langle s_1 \rangle, \ldots, \langle s_k \rangle]) = \langle t \rangle$. Note that, by the closedness of Ω, $\langle t \rangle$ belongs to $\langle S \rangle$. Consistency is needed to ensure that $\delta(f[\langle s_1 \rangle, \ldots, \langle s_k \rangle])$ is uniquely determined. Moreover, using subtree-closedness, one can easily verify the following lemma by structural induction on t.

Lemma 2.1. *If Ω is a closed and consistent observation table, then $\delta(t) = \langle t \rangle$ for all $t \in \Sigma(S)$. In particular, for $t \in \Sigma(S)$, we have $t \in L(A_\Omega)$ if and only if $t \in L$.*

The learner $\mathrm{L}_*^{\mathrm{tfta}}$ starts with the observation table given by $S = \emptyset$ and $C = \{\square\}$. In its main loop, it first makes sure that Ω is closed and consistent. This is done by a straightforward procedure complete that adds appropriate trees and contexts to S and C, resp., until Ω is closed and consistent. Then, $\mathrm{L}_*^{\mathrm{tfta}}$ constructs A_Ω and passes it to the teacher in an equivalence query. If the teacher accepts it, learning has been successful. Otherwise, subtrees(t) is added to S and the next iteration starts. Whenever elements are added to S or C, the required membership queries are asked to fill the new cells (t, c) of the table with the membership information $\chi_L(c \cdot t)$.

Below follows the pseudo code of the learner. In this pseudo code, we denote an observation table by the components S and C:

```
procedure L*^tfta where Ω = (S, C)
  Ω := (∅, {□})
  loop
    complete(Ω);
    construct A_Ω;
    t := eqQuery(A_Ω);          (ask equivalence query)
    if t = ⊥ then return A_Ω
    else S := S ∪ subtrees(t)

procedure complete(S, C)
  loop
    if ∃c ∈ Σ_□(S), s, s' ∈ S: ⟨s⟩ = ⟨s'⟩ ∧ ⟨c·s⟩ ≠ ⟨c·s'⟩ then    (table inconsistent)
      choose d ∈ C with member(d·c·s) ≠ member(d·c·s');
      C := C ∪ {d·c}                                          (add witness to C)
    else if ∃t ∈ Σ(S) such that ⟨t⟩ ∉ ⟨S⟩ then                (table not closed)
      S := S ∪ {t}
    else return;
```

Clearly, as long as Ω is not closed and consistent, each iteration of complete enlarges $\langle S\rangle$. In particular, complete terminates, because the index of L is finite. Now, consider the main procedure of $\mathrm{L}_*^{\mathrm{tfta}}$, and let Ω' be the new observation table Ω' obtained by adding subtrees(t) to S (where t is a counterexample). If Ω' would still be closed and consistent, on the one hand, it could easily be shown that $A_\Omega = A_{\Omega'}$. On the other hand, Lemma 2.1 would apply to $A_{\Omega'}$, stating that t is not a counterexample for $A_{\Omega'}$, contradicting the fact that it is a counterexample for A_Ω. Thus, Ω' cannot be closed and consistent. By the reasoning above, this means that the following call of complete enlarges $\langle S\rangle$. We conclude that $\mathrm{L}_*^{\mathrm{tfta}}$ terminates after at most n executions of the main loop, where n is the index of L.

Theorem 2.1 ([32]). *Let $A_L^{\mathrm{t}} = (\Sigma, Q, \delta, F)$. The learner $\mathrm{L}_*^{\mathrm{tfta}}$ returns an fta isomorphic to A_L^{t}, and runs in time polynomial in m^r and $|\delta|$, where m is the maximum size of counterexamples returned by the teacher, r is the maximum rank of symbols in Σ, and $|\delta|$ is the number of transitions.*

Note that the number $|Q|$ of states of A_L^{t} (i.e., the index n of L) does not occur in the preceding statement, because the totality of the fta implies that $|\delta| \geq |Q|$. Let us have a look at an example.

Example 2.1. Let $\Sigma = \{f^{(2)}, g^{(1)}, a^{(0)}\}$, and consider the tree language L consisting of all trees in T_Σ that do not contain two nodes such that one is a child of the other and both are labelled with the same symbol.

The learner $\mathrm{L}_*^{\mathrm{tfta}}$ starts with the table $(\emptyset, \{\Box\})$, which is not closed, because $\langle S\rangle = \emptyset$ does not contain $\langle a\rangle$, but $a \in \Sigma(S)$. Thus, complete adds a to S. The resulting observation table is the first one shown in Figure 1. Here, the trees in S are those above the single horizontal line, and the trees in $\Sigma(S) \setminus S$ are those shown below it. The table is obviously closed and consistent, because all trees in $\Sigma(S)$ have equal rows. The transitions of the resulting automaton A_Ω are shown

to the left of the rows they result from. Since the state $\langle a \rangle$ is accepting (because $a \in L$, which is signified by the fact that $\langle a \rangle$ equals 1 at $\Box$), we have $L(A_\Omega) = \mathrm{T}_\Sigma$. Hence, the teacher may give the counterexample $t = g[g[a]]$. The table resulting from the addition of subtrees(t) to S is inconsistent, since the two trees shown in boldface letters have equal rows, whereas the trees they are subtrees of do not. After the addition of $g[\Box]$ to C, the table is closed and consistent. The resulting fta is passed to the teacher in another equivalence query, and the teacher returns a counterexample. Again, the table needs to be made consistent using complete. As the reader may check, the fta A_Ω obtained from the resulting table is isomorphic to A_L^{t}.

Let us say that a tree t is *live* (with respect to a recognizable tree language $L \subseteq \mathrm{T}_\Sigma$) if it occurs as a subtree of at least one tree in L. Otherwise, t is *dead*. As a direct consequence of this definition, the set of dead trees forms a congruence class of $\equiv_L$ (or is empty). The state of A_L^{t} corresponding to this congruence class is said to be the *dead state of* A_L^{t} (if it exists). The canonical *partial* fta recognizing L, denoted by A_L^{p}, is constructed in the same way as A_L^{t}, but taking as its state set the set $\{[t]_{\equiv_L} \mid t \in \mathrm{T}_\Sigma \text{ is live}\}$, and restricting the transition function accordingly. In other words, A_L^{p} is obtained from A_L^{t} by deleting its dead state, if it exists, and is equal to A_L^{t}, otherwise. If a computation of A_L^{t} reaches the dead state on one of the subtrees of the input tree, then this input tree cannot be accepted. Hence, we obviously have $L(A_L^{\mathrm{p}}) = L(A_L^{\mathrm{t}}) = L$. We shall now consider a learner that constructs A_L^{p} instead of A_L^{t}.

The learner $\mathrm{L}_*^{\mathrm{tfta}}$ has the advantage that it asks at most n equivalence queries, where n is the index of L. Its major disadvantages are that (a) S potentially contains a lot of redundant information, since all subtrees of all counterexamples received end up in S, and (b) the observation table contains $|\Sigma(S)|$ rows to make A_Ω total. Together, (a) and (b) are responsible for the appearance of m^r in Theorem 2.1. Moreover, A_L^{t} always contains at least n^r transitions, whereas the number of transitions of A_L^{p} may be much smaller. The learner $\mathrm{L}_*^{\mathrm{fta}}$ developed in [18] avoids these disadvantages at the price of potentially asking a considerably larger number of equivalence queries.

Even $\mathrm{L}_*^{\mathrm{fta}}$ uses an observation table. However, rather than indexing the rows by the trees in $\Sigma(S)$, they are now indexed by trees in a set T such that $S \subseteq T \subseteq \Sigma(S)$. Thus, this set T takes the role of $\Sigma(S)$, but will typically not contain all trees in $\Sigma(S)$. As before, columns are indexed by contexts from a finite set $C \subseteq \mathrm{C}_\Sigma$.

Since $S \subseteq T \subseteq \Sigma(S)$, both T and S are subtree-closed. In addition to this, $\mathrm{L}_*^{\mathrm{fta}}$ maintains the invariant that, for every tree $t \in T$, there is exactly one tree $s \in S$ such that $\langle s \rangle = \langle t \rangle$. This means that closedness and consistency do not need to be checked explicitly, because S never contains redundant information. As a consequence, $A_\Omega = (Q, \Sigma, \delta, F)$ can be defined as before, the only difference being that it is total only if it happens to be the case that $T = \Sigma(S)$. As the trees in S have pairwise distinct rows, the correspondences between S and Q and between T and δ (viewing δ as a set of transitions) are bijections. In particular, each transition is represented by a unique tree in T.

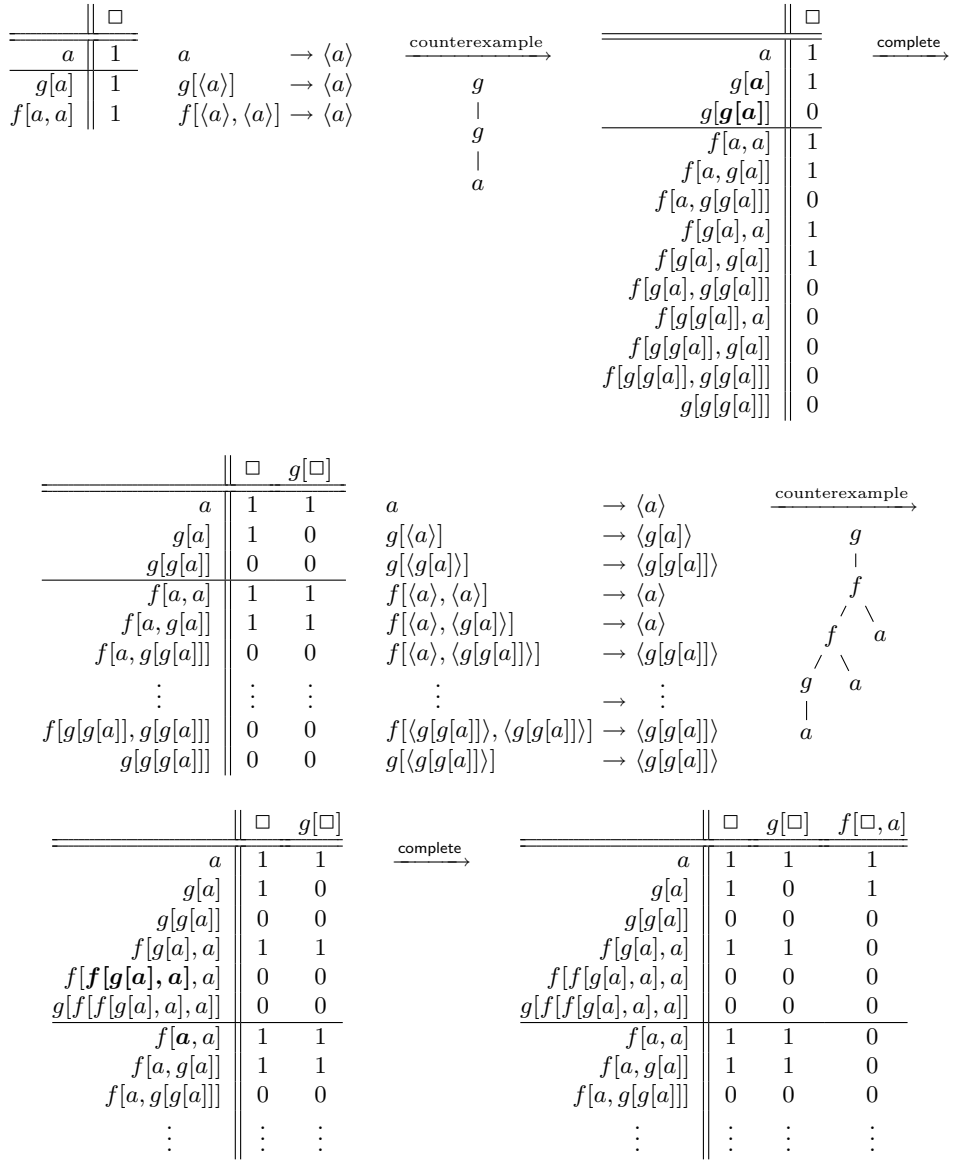


Figure 1: A run of $\mathrm{L}_*^{\mathrm{tfta}}$, showing (partial) observation tables, inconsistencies (in boldface letters), transitions resulting from the rows of consistent tables (except for the final table), and counterexamples that the teacher may choose to return.

Similar to $\mathrm{L}_*^{\mathrm{tfta}}$, $\mathrm{L}_*^{\mathrm{fta}}$ starts with the observation table given by $S = \emptyset$ (and, thus, also $T = \emptyset$), and $C = \{\Box\}$. It repeatedly constructs A_Ω and asks an equivalence query. As long as a counterexample t is received, Ω is extended by a tree (and possibly a context) extracted from t, and the process continues:

```
procedure L*^fta where Ω = (S, T, C)
  Ω := (∅, ∅, {□})
  loop
    construct A_Ω;
    t := eqQuery(A_Ω);          (ask equivalence query)
    if t = ⊥ then return A_Ω
    else Ω := extend(Ω, t)
```

The heart of $\mathrm{L}_*^{\mathrm{fta}}$ is the procedure extend, which examines a counterexample in a bottom-up manner to find out where things go wrong, rather than adding all subtrees of t to S. The technique used for this was introduced by Shapiro [35] and is known as contradiction backtracking. The pseudo code looks like this:

```
procedure extend(Ω, t) where Ω = (S, T, C)
  loop
    decompose t into t = c · t' where t' = f[s_1, ..., s_k] ∈ Σ(S) \ S;
    if t' ∈ T then
      let s be the unique tree in S with ⟨s⟩ = ⟨t'⟩
      if member(c · s) = member(t) then t := c · s        (case 1a)
      else return close(S, T, C ∪ {c})                   (case 1b)
    else return close(S, T ∪ {t'}, C)                    (case 2)
```

Here, the decomposition of t into $c \cdot t'$ can be done by a simple algorithm that checks in a bottom-up manner which subtrees of t belong to S, and returns the first tree t' encountered which is not in S (but which, therefore, must necessarily be in $\Sigma(S)$). The procedure close is a simplified version of the procedure complete of $\mathrm{L}_*^{\mathrm{tfta}}$, corresponding to the second case in the latter. It checks the trees $t \in T$ one by one, and adds t to S if S does not yet contain a tree s with $\langle s \rangle = \langle t \rangle$. Let δ be the transition function of A_Ω. If $t' \in T$, then $\delta(c \cdot s) = \delta(c \cdot t') = \delta(t)$, because $\delta(s) = \langle s \rangle = \langle t' \rangle = \delta(t')$. In other words, A_Ω returns the same answer if run on t and $c \cdot s$. Together with the condition $\mathsf{member}(c \cdot s) = \mathsf{member}(t)$, this means that $c \cdot s$ is also a counterexample, in case 1a. In case 1b, we have found a context c that separates the trees s and t' that have been equivalent according to Ω. Finally, in case 2, we have found a missing transition.

The use of contradiction backtracking in extend makes sure that the trees in S represent pairwise distinct states, those in T represent pairwise distinct transitions, and the total number of contexts added does not exceed the number of states. Moreover, it guarantees that no dead tree is ever added to T. Indeed, only case 2 results in the addition of a tree t' to T. Since the transition represented by t' is not in T, we know that A_Ω rejects $t = c \cdot t'$. Hence, t must be a positive counterexample, which shows that t' is live.[3] These properties make $\mathrm{L}_*^{\mathrm{fta}}$ quite efficient.

[3] This fact, showing that c is a so-called sign of life for t', will turn out to be of some importance in Section 3.

Theorem 2.2 ([18]). *The learner* $\mathrm{L}_*^{\mathrm{fta}}$ *returns an fta* (Σ, Q, δ, F) *isomorphic to* A_L^{p}*, and runs in time* $O(r \cdot |Q| \cdot |\delta| \cdot (|Q| + m))$*, where* m *is the maximum size of counterexamples returned by the teacher,* r *is the maximum rank of symbols in* Σ*, and* $|\delta|$ *is the number of transitions.*

The algorithm requires $|Q| + |\delta| + 1$ equivalence and $m + |Q| \cdot (|\delta| + 1)$ membership queries. As mentioned above, the number of equivalence queries asked is the major disadvantage of $\mathrm{L}_*^{\mathrm{fta}}$ in comparison with $\mathrm{L}_*^{\mathrm{tfta}}$. In practice, the number of equivalence queries used by $\mathrm{L}_*^{\mathrm{fta}}$ can often be reduced by re-using counterexamples [17]; see also the following example.

Example 2.2. Let $\Sigma = \{f^{(2)}, g^{(1)}, a^{(0)}\}$ be as in Example 2.1, and consider the tree language L consisting of all trees of the form $c \cdot f[t, a]$, where $c \in \mathrm{C}_{\{g\}}$ and $t \in \mathrm{T}_{\{g,a\}}$. Thus, the trees in L consist of a chain of gs at the top, followed by a single f, whose first subtree is a chain of gs (ending in an a), whereas the second is a single a.

In the first step, the teacher will be given the empty automaton, which accepts the empty language. Suppose the teacher returns the left-most tree in Figure 2 as a counterexample. Searching for a subtree in $\Sigma(S) \setminus S$ in a bottom-up manner, we immediately encounter one of the leaves a and observe that it represents a missing transition (case 2). Therefore, a is added to T (and close adds it to S, because S does not yet contain any tree whose row is 0). Following $\mathrm{L}_*^{\mathrm{fta}}$ strictly, we would now build the new automaton A_Ω and ask the teacher a new equivalence query. However, since the current tree is still a counterexample (it is not accepted by the new automaton either), we can as well continue using the current tree (see [17]). We now find the subtree $g[a]$, which represents again a new transition, but not a new state. In the next iteration (again re-using the counterexample), we find that $g[a]$ is in T and can be replaced with a without invalidating the counterexample (case 1a). Thus, we continue with the third tree in Figure 2, and find that $f[a, a]$ represents a new transition and state. Finally, we also find that $g[f[a, a]]$ represents a transition. When this has happened, the automaton correctly accepts the tree, so that we have to ask a new equivalence query.

Suppose the teacher chooses the leftmost tree in the second row of Figure 2. We find that $g[a]$ cannot be replaced with a once more, because $f[a, a] \in L$ (case 1b). Consequently, $f[a, \Box]$ is a context that distinguishes between a and $g[a]$.

Finally, when processing the last counterexample, we first discover that $g[g[a]]$ represents a transition, and then that $f[g[a], a]$ represents another one. Now, an equivalence query reveals that the resulting automaton is the correct one.

Recently, Besombes and Marion [7] have proposed the learner $\mathrm{L}_*^{\mathrm{rep}}$ (called Altex in [7]), that avoids the use of equivalence queries. Instead, it exploits a set of positive examples in which all the transitions of the sought automaton are required to be represented (see also [2]). Intuitively, there is a close relation between the two learners, because $\mathrm{L}_*^{\mathrm{fta}}$ uses equivalence queries precisely in order to discover such representatives. It may be interesting to try to find out whether there is a deeper formal relationship.

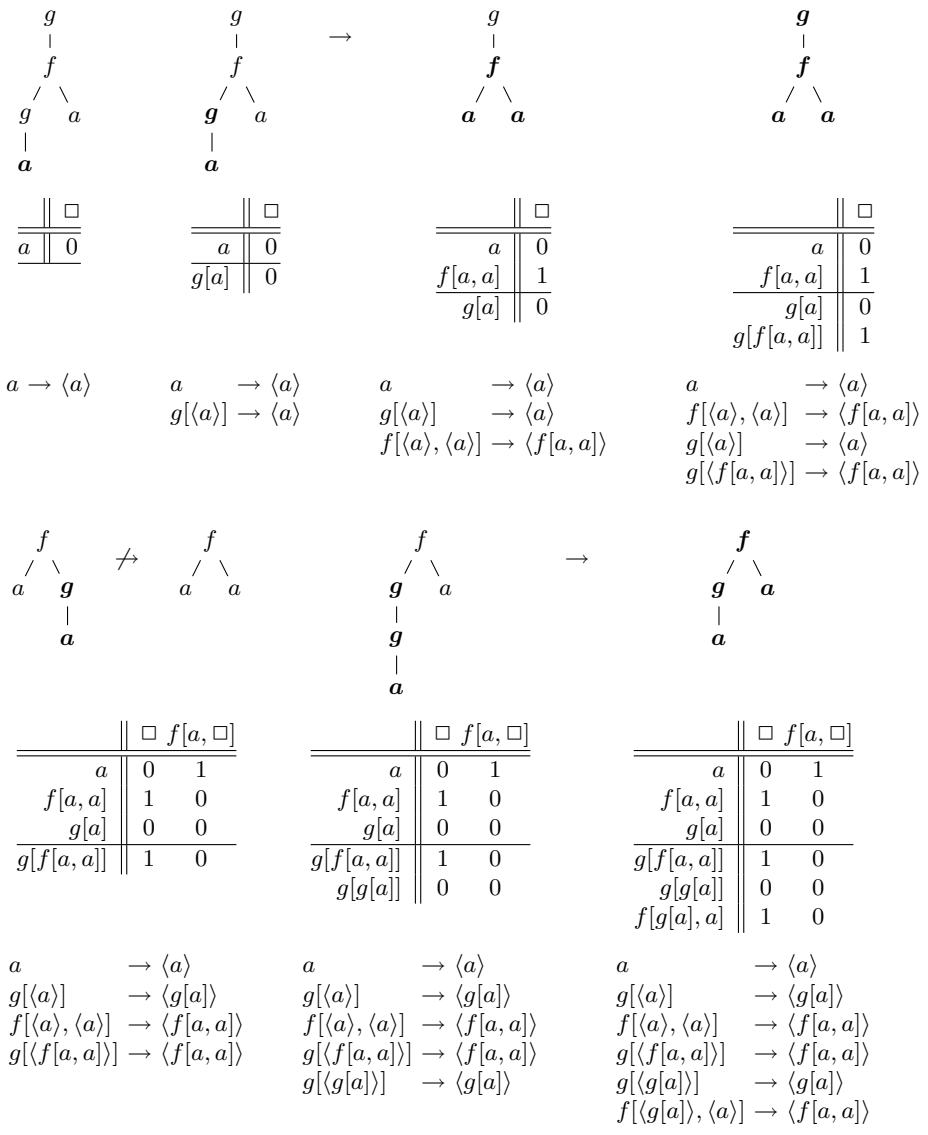


Figure 2: A run of $\mathrm{L}_*^{\mathrm{fta}}$, showing the trees inspected, the resulting observation tables, and the transitions. Steps according to case 1a (preserving the property of being a counterexample) are indicated by '$\to$', whereas $\not\to$ indicates steps according to case 1b (yielding a separating context).

Let us have a coarse look at $\mathrm{L}_*^{\mathrm{rep}}$. A set $R \subseteq \mathrm{T}_\Sigma$ is a *representative sample for* L if subtrees(R) contains, for every live tree $t = f[t_1, \ldots, t_k]$, a tree $t' = f[t'_1, \ldots, t'_k]$ such that $t'_1 \equiv_L t_1, \ldots, t'_k \equiv_L t_k$. In other words, the transition $f[[t_1]_{\equiv_L}, \ldots, [t_k]_{\equiv_L}] \to [t]_{\equiv_L}$ of the canonical fta is represented by a subtree of at least one of the trees in R. Now, learning starts with the observation table given by $T = \text{subtrees}(R)$ and $C = \{c \in \mathrm{C}_\Sigma \mid \exists t \in \mathrm{T}_\Sigma \colon c \cdot t \in R\}$. The set T is never going to change, and there is no distinguished subset S of trees representing states.

Somewhat similar to the situation in $\mathrm{L}_*^{\mathrm{tfta}}$, and in contrast to $\mathrm{L}_*^{\mathrm{fta}}$, Ω may be inconsistent, which now means that there are trees $t = f[t_1, \ldots, t_k]$ and $t' = f[t'_1, \ldots, t'_k]$ in T such that $\langle t_i \rangle = \langle t'_i \rangle$ for $i = 1, \ldots, k$, but $\langle t \rangle \neq \langle t' \rangle$. It can be shown that, in this case, there is an inconsistency with $t_i \equiv_L t'_i$ for all but one $i \in \{1, \ldots, k\}$. With this in mind, the situation becomes entirely similar to $\mathrm{L}_*^{\mathrm{tfta}}$: if j is the unique index with $t_j \not\equiv_L t'_j$, and $d \in C$ is a context separating t from t' (which exists because $\langle t \rangle \neq \langle t' \rangle$), then the context $d \cdot c$ with $c = f[t_1, \ldots, t_{j-1}, \Box, t_{j+1}, \ldots, t_k]$ separates t_j from t'_j.

The learner can now choose such a separating context d for every inconsistent pair of trees t and t' as above, and ask a membership query for each of the trees $d \cdot f[t_1, \ldots, t_{j-1}, t'_j, t_{j+1}, \ldots, t_k]$ ($j \in \{1, \ldots, k\}$), until the answer differs from the table entry for t in column d, to find c. In this way, a context $d \cdot c$ that separates t_j from t'_j is obtained.[4] Having found such a context, $\mathrm{L}_*^{\mathrm{rep}}$ adds it to C and checks again whether the observation table is consistent. Since the index of L is finite, the process must eventually terminate, yielding a consistent table. This table gives rise to an fta A_Ω in a similar manner as before. For a consistent table, using the fact that every transition is represented in $T = \text{subtrees}(R)$, it can be shown by induction on the size of minimal separating contexts that, for $t, t' \in T$, if $\langle t \rangle = \langle t' \rangle$, then $t \equiv_L t'$. From this, it follows easily that A_Ω is isomorphic to A_L^{p}.[5]

Theorem 2.3 ([7])**.** *The learner* $\mathrm{L}_*^{\mathrm{rep}}$ *returns an fta* (Σ, Q, δ, F) *isomorphic to* A_L^{p} *in time polynomial in* $\sum_{t \in R} |t|$ *(where* $|t|$ *denotes the size of* t*).*

Let us have a look at an example.

Example 2.3. Let $\Sigma = \{f^{(2)}, a^{(0)}, b^{(0)}\}$ and $L = \mathrm{T}_\Sigma \setminus (\mathrm{T}_{\{f,a\}} \cup \{b\})$, i.e., L contains all trees over Σ of size greater than one that contain at least one b. The canonical fta contains states q_a, q_b, q_{f}, where q_{f} is final. Its transition table is

$$\delta(t) = \begin{cases} q_a & \text{if } t \in \{a, f[q_a, q_a]\} \\ q_b & \text{if } t = b \\ q_{\mathrm{f}} & \text{otherwise.} \end{cases}$$

The set R of trees shown in Figure 3 is a representative sample. Building the

[4] Alternatively, following the description in [7], the learner could simply pick any inconsistent pair t, t' as above and a separating context d, and add all contexts $d \cdot f[t_1, \ldots, t_{j-1}, \Box, t_{j+1}, \ldots, t_k]$ to C, because it will eventually also encounter the right one and include it. However, it seems clear that this may have a negative impact on the efficiency.

[5] The proof of this fact given in [7, Lemma 5] does not seem to be convincing, but it is easily corrected by the inductive argument mentioned, showing that $\langle t \rangle = \langle t' \rangle$ implies $t \equiv_L t'$.

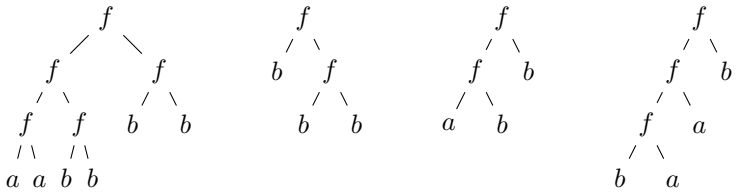


Figure 3: A representative sample R

corresponding initial observation table, we see that $\sim_\Omega$ divides $T = \text{subtrees}(R)$ into two equivalence classes, namely $T \setminus L = \{a, b, f[a, a]\}$ and $T \cap L$. The reason is that, among the contexts in C, only $\Box$ separates any trees at all, because every $c \in C \setminus \{\Box\}$ (i.e., every context obtained from a tree in R by replacing a proper subtree with $\Box$) contains a b, which means that $\chi_L(c \cdot t) = 1$ for all $t \in \mathrm{T}_\Sigma$.

Thus, we should be able to find a pair of trees in T revealing an inconsistency. Indeed, there are three, obtained by combining $f[a, b], f[b, a], f[b, b] \in L$ with $f[a, a] \notin L$. This gives rise to the context $f[a, \Box]$ separating a from b. Of course, $f[\Box, a]$ would do as well, but it may be interesting to note that neither $f[\Box, b]$ nor $f[b, \Box]$ does (see also footnote 4). As the reader may wish to verify, the table Ω' enlarged by this context is consistent, and $A_{\Omega'}$ is isomorphic to A^{p}_L.

3 Learning Tree Series

It is now a natural step to wonder whether learning of recognizable tree series is possible as well. The number of papers addressing this problem is still rather small. One may roughly divide them into two categories. The first deals with the special case of stochastic tree automata, weighted tree automata (wta) with weights in $[0, 1]$ that compute a probability distribution on T_Σ. This case is of particular interest because stochastic languages play an important role in, e.g., natural language processing. To learn stochastic tree languages, it is probably most natural to consider a learning-from-text-like setting: positive examples are drawn according to a probability distribution D, and the goal is to learn D in the limit by, e.g., constructing an appropriate wta. A learner of this kind has recently been presented by Denis and Habrard [16].

The second category of learners does not assume that the sought wta is a stochastic tree automaton. There seem to be only two results of this kind, both using the MAT model and the general algorithmic idea explained in the previous section. Let us first give some basic definitions. Readers who wish to read a more decent introduction to weighted tree automata are referred to the excellent survey by Fülöp and Vogler [22].

Let $\mathbf{S} = (\mathbb{S}, +, \cdot, 0, 1)$ be a (commutative) semiring, i.e., a set $\mathbb{S}$ together with binary addition and multiplication operations $+$ and $\cdot$ and distinct elements $0, 1 \in \mathbb{S}$ such that $(\mathbb{S}, +, 0)$ and $(\mathbb{S}, \cdot, 1)$ are commutative monoids, multiplication distributes over addition, and 0 is absorbing with respect to multiplication. From now on, we

simply denote $\mathbf{S}$ by $\mathbb{S}$. A *tree series* is a mapping $\psi\colon \mathrm{T}_\Sigma \to \mathbb{S}$. Given such a tree series, we call the set $\mathrm{supp}(\psi) = \{t \in \mathrm{T}_\Sigma \mid \psi(t) \neq 0\}$ the *support of* ψ.

Below, for a finite index set I, we let $\mathbb{S}^I$ denote the set of all vectors over $\mathbb{S}$ indexed by I. As usual, the ith component of $v \in \mathbb{S}^I$ is denoted by v_i, for $i \in I$. The inner product of $u, v \in \mathbb{S}^I$ is $u \cdot v = \sum_{i \in I} u_i \cdot v_i$.

Definition 3.1. *Let $\mathbb{S}$ be a semiring. A* weighted tree automaton *(wta) over $\mathbb{S}$ is a tuple $A = (\Sigma, Q, \mu, \lambda)$ consisting of a ranked alphabet Σ, a ranked alphabet Q of states such that $Q = Q_{(0)}$, a transition weight table $\mu \in \mathbb{S}^{\Sigma(Q) \times Q}$, and a root weight mapping $\lambda \in \mathbb{S}^Q$. Thus, μ assigns a weight μ_τ to every transition $\tau \in \Sigma(Q) \times Q$. A is (bottom-up)* deterministic *(a dwta) if, for every $l \in \Sigma(Q)$, there is at most one $q \in Q$ such that $\mu_{l \to q} \neq 0$.*

For $t = f[t_1, \ldots, t_k] \in \mathrm{T}_\Sigma$, we define $\widehat{\mu}(t) \in \mathbb{S}^Q$ by setting

$$\widehat{\mu}(t)_q = \sum_{q_1, \ldots, q_k \in Q} \mu_{f[q_1, \ldots, q_k] \to q} \cdot \prod_{i=1,\ldots,k} \widehat{\mu}(t_i)_{q_i},$$

for all $q \in Q$.

The tree series recognized by A is given by $\psi_A(t) = \lambda \cdot \widehat{\mu}(t)$, and is called a recognizable tree series.

In the following, we want to consider the problem of learning a wta in the MAT model, first for dwta over a semifield, and then for nondeterministic wta over a field. Clearly, for this to be possible, the teacher has to be given appropriate capabilities. Thus, if $\mathcal{A}$ is the class of wta to be learned, and ψ is the target series, membership queries become *coefficient queries*: given a tree $t \in T_\Sigma$, the procedure $\mathsf{coef}(t)$ will return $\psi(t)$. Similarly, equivalence queries have to be extended: the input is a wta $A \in \mathcal{A}$, and $\mathsf{eqQuery}(A)$ will either return $\perp$, indicating that $\psi_A = \psi$, or a counterexample, a tree $t \in \mathrm{T}_\Sigma$ such that $\psi_A(t) \neq \psi(t)$.

As mentioned, we are first going to have a look at the deterministic case. For readers who are not yet familiar with wta, a small example (which will be continued later) follows.

Example 3.1. We consider the semifield $\mathbb{S} = (\mathbb{Z} \cup \{\infty\}, \min, +, \infty, 0)$. To avoid confusion, the reader should keep in mind that $+$ plays the role of multiplication in this example, with ∞ being the absorbing element, and 0 being the neutral element.

As in Example 2.2, let $\Sigma = \{f^{(2)}, g^{(1)}, a^{(0)}\}$. For a tree t of the form $c \cdot f[t', a]$, where $c \in \mathrm{C}_{\{g\}}$ and $t' \in \mathrm{T}_{\{g,a\}}$, let $\psi(t) = 2m + n$, where m is the number of occurrences of g in c, and n is the size of t'. For all other trees $t \in \mathrm{T}_\Sigma$, let $\psi(t) = \infty$. Thus, the support of ψ is the tree language in Example 2.2.

A dwta over $\mathbb{S}$ recognizing ψ can be constructed by using states q_1, q_2, q_3. Except for the addition of weights, the automaton is the same as the one in Example 2.2. It will be in state q_1 when it has just read an a, in state q_2 when it has read a number of gs above an a, and in state q_3 when it has read a tree in $\mathrm{supp}(\psi)$. For the specification of concrete dwta, it is convenient to write μ as a set of rules of the form $l \xrightarrow{w} q$, where $l \in \Sigma(Q)$ and q is the unique element of Q such that $w = \mu_{l \to q}$

is non-zero (which, in the present case, means that $w \neq \infty$). Using this notation, A contains the following rules:

$$a \xrightarrow{0} q_1, \qquad g[q_1] \xrightarrow{1} q_2, \qquad g[q_2] \xrightarrow{1} q_2,$$
$$f[q_1, q_1] \xrightarrow{1} q_3, \quad f[q_2, q_1] \xrightarrow{1} q_3, \quad g[q_3] \xrightarrow{2} q_3.$$

Furthermore, $\lambda_{q_1} = \lambda_{q_2} = \infty$ and $\lambda_{q_3} = 0$.

Now, let us have a look at the MAT learner $\mathrm{L}_*^{\mathrm{dwta}}$ for dwta over a (commutative) semifield $\mathbb{S}$ by Maletti [30]. It extends $\mathrm{L}_*^{\mathrm{fta}}$ to the weighted case and generalizes an earlier version proposed by Drewes and Vogler [19], which was restricted to the class of "all-accepting" dwta.

The learner $\mathrm{L}_*^{\mathrm{dwta}}$ makes use of the Myhill-Nerode theorem for deterministically recognizable tree series over commutative semifields [8]. Thus, from now on, every $a \in \mathbb{S} \setminus \{0\}$ is assumed to have a multiplicative inverse. As in $\mathrm{L}_*^{\mathrm{fta}}$, observation tables are given by sets $S, T \subseteq \mathrm{T}_\Sigma$ and $C \subseteq \mathrm{C}_\Sigma$. The entry in row t and column c is now the coefficient $\psi(c \cdot t)$. The fact that $\mathrm{L}_*^{\mathrm{fta}}$, in T, only collects live trees becomes now crucial for the correctness of the learner. In the context of tree series, a tree $t \in \mathrm{T}_\Sigma$ is live if there exists a *sign of life for* t, a context $c \in \mathrm{C}_\Sigma$ such that $\psi(c \cdot t) \neq 0$. The case of tree series poses a difficulty not present in the language case: if $\widehat{\mu}(t)_q \neq 0$ but $\lambda_q = 0$, then the value of $\widehat{\mu}(t)_q$ is hidden in the sense that a coefficient query on t will yield $\psi(t) = 0$. To determine the right coefficients during the construction of A_Ω, we thus have to make sure that C contains a sign of life, for every $t \in T$. In the algorithm `extend`, this is easily guaranteed by changing case 2 in such a way that c is added to C (see footnote 3 on p. 258).

Thus, the crucial invariant maintained by $\mathrm{L}_*^{\mathrm{dwta}}$ is that, as in $\mathrm{L}_*^{\mathrm{fta}}$, the observation table $\Omega = (S, T, C)$ satisfies $S \subseteq T \subseteq \Sigma(S)$. In addition, C now contains a sign of life for every tree in T. For $t, t' \in T$, what used to be the equality of $\langle t \rangle$ and $\langle t' \rangle$ in the unweighted setting, is now replaced by the requirement that one row be a multiple of the other. More precisely, let $\langle t \rangle \approx \langle t' \rangle$ if and only if there exists an $a \in \mathbb{S}$ such that $\langle t \rangle = a \cdot \langle t' \rangle$ (where $a \cdot \langle t' \rangle$ denotes the scalar multiplication of the row $\langle t' \rangle$ by a). Note that, due to the existence of signs of life, a is non-zero and is uniquely determined for every pair of trees in T (if it exists). Similar to $\mathrm{L}_*^{\mathrm{fta}}$, for every tree $t \in T$, S will always contain exactly one tree s such that $\langle s \rangle \approx \langle t \rangle$. Given $t \in T$, we will denote this particular tree $s \in S$ by $\mathrm{rep}_\Omega(t)$.

Now, we can assign a weight $\psi_\Omega(t)$ to every tree $t \in T$: $\psi_\Omega(t)$ is the unique factor $a \in \mathbb{S}$ such that $\langle t \rangle = a \cdot \langle \mathrm{rep}_\Omega(t) \rangle$. In particular, $\psi_\Omega(s) = 1$ for every $s \in S$.[6]

Using these definitions, an observation table $\Omega = (S, T, C)$ gives rise to the dwta $A_\Omega = (\Sigma, Q, \mu, \lambda)$, where

- $Q = \langle S \rangle$,
- for every transition $\tau = (f[\langle s_1 \rangle, \ldots, \langle s_k \rangle] \to \langle t \rangle)$, where $t = f[s_1, \ldots, s_k] \in T$, we let $\mu_\tau = \psi_\Omega(t)$,

[6] This definition of $\psi_\Omega(t)$ differs from the one given in [30], but fulfills the same purpose. This illustrates the fact that there may be several minimal wta recognizing ψ, which differ in their transition weights (and in λ).

- all remaining transition weights μ_τ are 0, and
- $\lambda_{\langle s \rangle} = \psi(s)$ for all $s \in S$.

In the same way as $\mathrm{L}_*^{\mathrm{fta}}$, $\mathrm{L}_*^{\mathrm{dwta}}$ now starts with the observation table $\Omega = (\emptyset, \emptyset, \{\Box\})$. It repeatedly constructs A_Ω, asks an equivalence query, and passes the counterexample received (if any) to the procedure extend. In other words, the main procedure of $\mathrm{L}_*^{\mathrm{dwta}}$ looks exactly like that of $\mathrm{L}_*^{\mathrm{fta}}$ (although it does now work with dwta rather than fta, of course). Even extend looks quite the same as before, the major difference being that we now add the context c as a sign of life in case 2:

```
procedure extend(Ω, t) where Ω = (S, T, C)
  loop
    decompose t into t = c · t' where t' = f[s_1, ..., s_k] ∈ Σ(S) \ S;
    let s = rep_Ω(t');
    if t' ∈ T then
      if coef(t) = ψ_Ω(t') · coef(c · s) then t := c · s          (case 1a)
      else return close(S, T, C ∪ {c})                            (case 1b)
    else return close(S, T ∪ {t'}, C ∪ {c})                       (case 2)
```

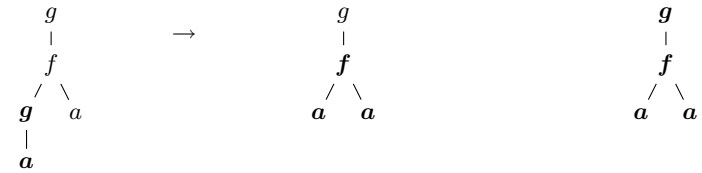
The following result, similar to Theorem 2.2, holds under the assumption that all relevant operations on $\mathbb{S}$ (addition, multiplication, and taking inverses) can be computed in constant time. Compared to Theorem 2.2, an additional factor $|Q|$ results from the fact that rows are not bit strings anymore, and thus cannot be stored as single integers.

Theorem 3.1 ([30])**.** *The learner* $\mathrm{L}_*^{\mathrm{dwta}}$ *returns a minimal dwta* $A = (\Sigma, Q, \mu, \lambda)$ *recognizing* ψ *in time* $O(r \cdot |Q|^2 \cdot |\delta| \cdot (|Q| + m))$*, where* m *is the maximum size of counterexamples returned by the teacher,* r *is the maximum rank of symbols in* Σ*, and* $|\delta|$ *is the number of transitions* $\tau \in \Sigma(Q) \times Q$ *such that* $\mu_\tau \neq 0$*.*

Let us have a look at an example.

Example 3.2. Consider the tree series ψ in Example 3.1, where, again, $\mathbb{S} = (\mathbb{Z} \cup \{\infty\}, \min, +, \infty, 0)$. We now apply $\mathrm{L}_*^{\mathrm{dwta}}$ in order to construct, by means of learning, a dwta over $\mathbb{S}$ recognizing ψ. The counterexamples used as well as the states and transitions discovered are the same as in Example 2.2. In particular, counterexamples are re-used if possible. Furthermore, the context c in case 2 of extend is not added to the table if the table already contains a sign of life for t'. To save space in Figure 4, the very first step, in which a is found to be a new state and transition, is omitted. Otherwise, the figure is very similar to Figure 2. Indeed, the resulting wta recognizes ψ, as the reader may easily check, although the transition weights differ from those used in Example 3.1.

It seems to be clear that the learner $\mathrm{L}_*^{\mathrm{rep}}$ discussed in the previous section carries over to deterministic wta over $\mathbb{S}$ in quite exactly the same way as $\mathrm{L}_*^{\mathrm{fta}}$. Thus, the resulting learner would use coefficient queries and a representative sample, the latter being a subset of $\mathrm{supp}(\psi)$ covering every transition of a minimal dwta recognizing



	$\square$	$g[f[g[\square],a]]$
a	∞	4
$g[a]$	∞	5

$$a \xrightarrow{0} \langle a\rangle \qquad g[\langle a\rangle] \xrightarrow{1} \langle a\rangle \qquad \lambda_{\langle a\rangle} = \infty$$

	$\square$	$g[f[g[\square],a]]$
a	∞	4
$f[a,a]$	1	∞
$g[a]$	∞	5

$$a \xrightarrow{0} \langle a\rangle \qquad f[\langle a\rangle,\langle a\rangle] \xrightarrow{0} \langle f[a,a]\rangle \qquad g[\langle a\rangle] \xrightarrow{1} \langle a\rangle$$
$$\lambda_{\langle a\rangle} = \infty,\ \lambda_{\langle f[a,a]\rangle} = 1$$

	$\square$	$g[f[g[\square],a]]$
a	∞	4
$f[a,a]$	1	∞
$g[a]$	∞	5
$g[f[a,a]]$	3	∞

$$a \xrightarrow{0} \langle a\rangle \qquad f[\langle a\rangle,\langle a\rangle] \xrightarrow{0} \langle f[a,a]\rangle \qquad g[\langle a\rangle] \xrightarrow{1} \langle a\rangle \qquad g[\langle f[a,a]\rangle] \xrightarrow{2} \langle f[a,a]\rangle$$
$$\lambda_{\langle a\rangle} = \infty,\ \lambda_{\langle f[a,a]\rangle} = 1$$

	$\square$	$g[f[g[\square],a]]$	$f[a,\square]$
a	∞	4	1
$f[a,a]$	1	∞	∞
$g[a]$	∞	5	∞
$g[f[a,a]]$	3	∞	∞

$$a \xrightarrow{0} \langle a\rangle \qquad f[\langle a\rangle,\langle a\rangle] \xrightarrow{0} \langle f[a,a]\rangle \qquad g[\langle a\rangle] \xrightarrow{0} \langle g[a]\rangle \qquad g[\langle f[a,a]\rangle] \xrightarrow{2} \langle f[a,a]\rangle$$
$$\lambda_{\langle a\rangle} = \lambda_{\langle g[a]\rangle} = \infty,\ \lambda_{\langle f[a,a]\rangle} = 1$$

	$\square$	$g[f[g[\square],a]]$	$f[a,\square]$
a	∞	4	1
$f[a,a]$	1	∞	∞
$g[a]$	∞	5	∞
$g[f[a,a]]$	3	∞	∞
$g[g[a]]$	∞	6	∞
$f[g[a],a]]$	2	∞	∞

$$a \xrightarrow{0} \langle a\rangle \qquad g[\langle a\rangle] \xrightarrow{0} \langle g[a]\rangle \qquad g[\langle f[a,a]\rangle] \xrightarrow{2} \langle f[a,a]\rangle \qquad g[\langle g[a]\rangle] \xrightarrow{1} \langle g[a]\rangle$$
$$f[\langle a\rangle,\langle a\rangle] \xrightarrow{0} \langle f[a,a]\rangle \qquad f[\langle g[a]\rangle,\langle a\rangle] \xrightarrow{1} \langle f[a,a]\rangle$$
$$\lambda_{\langle a\rangle} = \lambda_{\langle g[a]\rangle} = \infty,\ \lambda_{\langle f[a,a]\rangle} = 1$$

Figure 4: A run of $\mathrm{L}_*^{\mathrm{dwta}}$, similar to the run of $\mathrm{L}_*^{\mathrm{fta}}$ in Figure 2.

ψ. Note that, even though there may be various minimal dwta recognizing ψ, their representative sets coincide, because two minimal dwta recognizing the same tree series over $\mathbb{S}$ differ only in the weights of their transitions (and in their root weights).

We now turn to the second learner for recognizable tree series, proposed by Habrard and Oncina [26]. In contrast to the one explained above (and, in fact, also in contrast to all other extensions of L_* known to the author), this learner works for nondeterministic wta. This becomes possible by making the stronger assumption that $\mathbb{S}$, the semiring considered, is a field. Thus, from now on, $\mathbb{S}$ is even assumed to have additive inverses. From the point of view of MAT learning, the important consequence of this assumption is that we, again, can make use of a Myhill-Nerode theorem; see [22, Theorem 3.31].

Below, since we are now dealing with nondeterministic wta $A = (\Sigma, Q, \mu, \lambda)$, it is occasionally convenient to specify μ as a function $\mu\colon \Sigma(Q) \to \mathbb{S}^Q$. The connection between the two views is, of course, that $\mu_{l\to q} = \mu(l)_q$ for all $l \in \Sigma(Q)$ and $q \in Q$.

Before turning to the discussion of the learner, let us have a look at an example of a nondeterministic wta.

Example 3.3. Let $\Sigma = \{f^{(2)}, g^{(1)}, a^{(0)}\}$ and $\mathbb{S} = \mathbb{Q}$, where addition and multiplication are as usual. For a tree $t \in \mathrm{T}_\Sigma$, let $\psi(t) = m + n$, where n is the number of nodes labelled f in t, and m is the number of nodes labelled f in t that do not have a child node labelled f. In other words, we count fs, and those which do not have another f as a direct descendant are counted twice. A minimal wta $A = (\Sigma, Q, \mu, \lambda)$ recognizing ψ has three states q_1, q_2, q_3. The intuition behind them is as follows. At the root of a (sub-)tree t, state q_1 carries the weight $w = 0$ if the root of t is labelled f, and $w = 1$ otherwise. At the same time, q_2 carries the weight $1 - w$. State q_3 always carries the weight $\psi(t)$. Consequently, denoting $v \in \mathbb{S}^Q$ as $(v_{q_1}, v_{q_2}, v_{q_3})$, the specification of μ reads as follows:

$$
\begin{array}{lll}
\mu(a) = (1,0,0) & & \\
\mu(g[q_1]) = (1,0,0) & \mu(f[q_1,q_1]) = (0,1,2) & \\
\mu(g[q_2]) = (1,0,0) & \mu(f[q,q']) = (0,1,1) & \text{if } q_2 \in \{q,q'\} \subseteq \{q_1,q_2\} \\
\mu(g[q_3]) = (0,0,1) & \mu(f[q,q']) = (0,0,1) & \text{if } \{q,q'\} \in \{\{q_1,q_3\},\{q_2,q_3\}\}.
\end{array}
$$

The root weights are given by $\lambda = (0, 0, 1)$. Figure 5 illustrates a computation.

Now, suppose that ψ is a recognizable tree series over a field $\mathbb{S}$. For the moment, let Ω denote the infinite observation table obtained by taking all of T_Σ as T (indexing the rows) and all of C_Σ as C (indexing the columns). Then the rank of Ω, viewed as a matrix, is finite. Moreover, it is not difficult to show that, for every set $S \subseteq \mathrm{T}_\Sigma$, if there exists a tree $t \in \mathrm{T}_\Sigma$ such that $\langle t\rangle$ is linearly independent of $\langle S\rangle$, then a tree with this property can even be found in $\Sigma(S)$. Therefore, there is a finite subtree-closed set[7] $S \subseteq \mathrm{T}_\Sigma$ such that, for all $s \in S$, $\langle s\rangle$ is linearly independent of $\langle S\rangle \setminus \{\langle s\rangle\}$, and every row in $\langle \mathrm{T}_\Sigma\rangle$ is a linear combination of rows in $\langle S\rangle$.

[7] Recall that subtree-closedness of S means that S even contains all subtrees of trees in S.

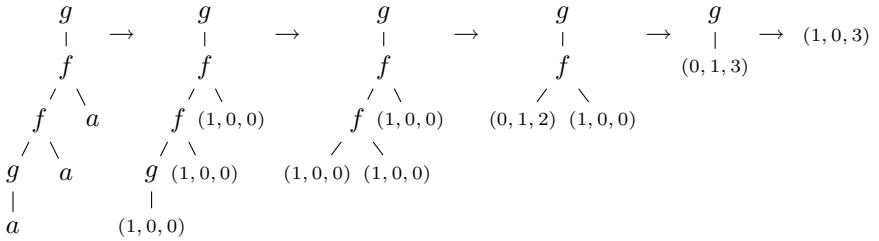


Figure 5: A computation of the wta in Example 3.3.

Assume that we have discovered such a set S, and let $Q = \langle S \rangle$. For every tree $t \in \mathrm{T}_\Sigma$, let $\widetilde{\mu}(t) \in \mathbb{S}^Q$ be the unique vector such that $\langle t \rangle = \sum_{s \in S} \widetilde{\mu}(t)_{\langle s \rangle} \cdot \langle s \rangle$. In other words, $\widetilde{\mu}(t)$ is the vector of coefficients of $\langle t \rangle$, if expressed as a linear combination of rows in $\langle S \rangle$. Then, for a tree t and a context c, we have

$$\psi(c \cdot t) = \Omega(t, c) = \sum_{s \in S} \widetilde{\mu}(t)_{\langle s \rangle} \cdot \Omega(s, c) = \sum_{s \in S} \widetilde{\mu}(t)_{\langle s \rangle} \cdot \psi(c \cdot s).$$

In particular, choosing $c = \Box$ and setting $\lambda_{\langle s \rangle} = \psi(s)$, we get $\psi(t) = \lambda \cdot \widetilde{\mu}(t)$. Since this is just the definition of $\psi_A(t)$, it remains to show how to discover S, together with a weight table μ such that $\widehat{\mu} = \widetilde{\mu}$.

This is done as follows, again using an observation table. As in $\mathrm{L}_*^{\mathrm{tfta}}$, rows are indexed by the trees in $\Sigma(S)$, i.e., $\Sigma(S)$ plays the role of T. Each time new contexts have been added to C, the learner makes sure that the table is closed, which now means that $\langle t \rangle$ is a linear combination of $\langle S \rangle$, for every tree $t \in \Sigma(S)$. Closedness can be achieved by an straightforward iterative procedure close that preserves subtree-closedness. Given that Ω is closed, a corresponding wta $A_\Omega = (\Sigma, Q, \mu, \lambda)$ with $Q = \langle S \rangle$ can be obtained along the lines of the preceding discussion: for every tree $l = f[\langle s_1 \rangle, \ldots, \langle s_k \rangle] \in \Sigma(Q)$, we let $\mu(l)$ be the unique vector such that $\langle t \rangle = \sum_{s \in S} \mu(l)_{\langle s \rangle} \cdot \langle s \rangle$. Furthermore, $\lambda_{\langle s \rangle} = \psi(s)$ for all $s \in S$.

Now, here is the pseudo-code of the main routine of the learner:

```
procedure L*^wta
  Ω = (S, C) := (∅, ∅)
  loop
    construct A_Ω;
    t := eqQuery(A_Ω);      (ask equivalence query)
    if t = ⊥ then return A_Ω;
    else
      C := C ∪ {c ∈ C_Σ | ∃t' ∈ T_Σ: c · t' = t};
      S := close(S);
```

Thus, when a counterexample is received, C is enlarged by all contexts obtained from this counterexample. Since it can be shown that this increases the rank of Ω, termination is guaranteed. (In fact, the learner in [26] is slightly more optimized than the version described here. Before asking a new equivalence query, it is checked

whether there is a tree $t \in T$ such that $\psi_{A_\Omega}(t) \neq \psi(t)$. In other words, there is a context $c \in C$ such that $c \cdot t$ is a counterexample. In this case, the learner can obviously proceed by using $c \cdot t$ as a counterexample, thus avoiding the need to ask an equivalence query.)

Every counterexample increases $|S|$, which never gets larger than the number of states of a minimal wta recognizing ψ. Furthermore, every counterexample t leads to the inclusion of at most $|t|$ new contexts in C. As all the basic steps in the algorithm can be performed in polynomial time in the size of Ω (i.e, in $|\Sigma(S)|+|C|$), we get the following theorem.

Theorem 3.2 ([26])**.** *For every recognizable tree series over* $\mathbb{S}$, $\mathrm{L}_*^{\mathrm{wta}}$ *learns a minimal wta* A *recognizing* ψ *in polynomial time with respect to the size of* A *and the size of the largest counterexample returned by the teacher.*

Again, let us have a look at an example.

Example 3.4. We apply $\mathrm{L}_*^{\mathrm{wta}}$ to the tree series in Example 3.3. The initial wta, without any states, assigns the weight 0 to all trees. The teacher may respond with the counterexample $f[a, a]$, which leads to the first observation table in Figure 6. In the figure, only as many contexts of C are shown as needed. For example, $f[a, \Box]$ is left out in the first table. Furthermore, for the sake of clarity, the part below the horizontal line in each table lists all of T, rather than only $T \setminus S$.

The teacher may now give the counterexample $t = f[f[f[a, a], a], f[a, a]]$, because $\psi_{A_\Omega}(t) = 27/4$ rather than 6. Of the contexts obtained from t, we need only $f[f[f[\Box, a], a], f[a, a]]$ to distinguish between three states; see the second table in Figure 6. A_Ω now recognizes ψ, even though the "intuition" of the learner differs from the one used to construct the (equivalent) wta in Example 3.3. More precisely, let $\mathrm{root}_f(t)$ be the predicate which is true if and only if the root symbol of t is f. Then, if $\mu(t) = (v_1, v_2, v_3)$, we have

$$v_1 = 1 - \psi(t)/2, \qquad v_2 = \begin{cases} 1 & \text{if } \mathrm{root}_f(t) \\ 0 & \text{otherwise,} \end{cases} \qquad v_3 = \begin{cases} \psi(t)/2 - 1 & \text{if } \mathrm{root}_f(t) \\ \psi(t)/2 & \text{otherwise.} \end{cases}$$

Indeed, given the choice of λ, this means that A_Ω recognizes ψ.

4 Final Remarks

We have considered a family of grammatical inference algorithms for tree languages and tree series that can be regarded as more or less direct descendants of the learner L_* proposed by Angluin in [3]. An approach that has not been discussed here is the one presented in [6, 38] for string and tree languages, respectively (see also [37, 36]). This approach uses so-called correction queries instead of membership queries. Given a recognizable tree language $L \subseteq \mathrm{T}_\Sigma$ to be learned, a correction query $\mathsf{correct}(t)$ (where $t \in \mathrm{T}_\Sigma$) is answered by returning the smallest context $c \in \mathrm{C}_\Sigma$ such that $c \cdot t \in L$. Here, contexts are ordered according to a Knuth-Bendix order. A special token is returned if no c with $c \cdot t \in L$ exists, i.e., in case t is

	$\Box$	$f[\Box, a]$	
$s_1 = a$	0	2	$\lambda_{\langle s_1 \rangle} = 0$
$s_2 = f[a, a]$	2	3	$\lambda_{\langle s_2 \rangle} = 2$
a	0	2	$\mu(a) = (1, 0)$
$g[s_1]$	0	2	$\mu(g[\langle s_1 \rangle]) = (1, 0)$
$g[s_2]$	2	4	$\mu(g[\langle s_2 \rangle]) = (\frac{1}{2}, 1)$
$f[s_1, s_1]$	2	3	$\mu(f[\langle s_1 \rangle, \langle s_1 \rangle]) = (0, 1)$
$f[s_1, s_2]$	3	4	$\mu(f[\langle s_1 \rangle, \langle s_2 \rangle]) = (-\frac{1}{4}, \frac{3}{2})$
$f[s_2, s_1]$	3	4	$\mu(f[\langle s_2 \rangle, \langle s_1 \rangle]) = (-\frac{1}{4}, \frac{3}{2})$
$f[s_2, s_2]$	5	6	$\mu(f[\langle s_2 \rangle, \langle s_2 \rangle]) = (-\frac{3}{4}, \frac{5}{2})$

	$\Box$	$f[\Box, a]$	$f[f[f[\Box, a], a], f[a, a]]$	
$s_1 = a$	0	2	6	$\lambda_{\langle s_1 \rangle} = 0$
$s_2 = f[a, a]$	2	3	7	$\lambda_{\langle s_2 \rangle} = 2$
$s_3 = g[f[a, a]]$	2	4	8	$\lambda_{\langle s_3 \rangle} = 2$
a	0	2	6	$\mu(a) = (1, 0, 0)$
$g[s_1]$	0	2	6	$\mu(g[\langle s_1 \rangle]) = (1, 0, 0)$
$g[s_2]$	2	4	8	$\mu(g[\langle s_2 \rangle]) = (0, 0, 1)$
$g[s_3]$	2	4	8	$\mu(g[\langle s_3 \rangle]) = (0, 0, 1)$
$f[s_1, s_1]$	2	3	7	$\mu(f[\langle s_1 \rangle, \langle s_1 \rangle]) = (0, 1, 0)$
$f[s_1, s_2]$	3	4	8	$\mu(f[\langle s_1 \rangle, \langle s_2 \rangle]) = (-\frac{1}{2}, 1, \frac{1}{2})$
$f[s_1, s_3]$	4	5	9	$\mu(f[\langle s_1 \rangle, \langle s_3 \rangle]) = (-1, 1, 1)$
$f[s_2, s_1]$	3	4	8	$\mu(f[\langle s_2 \rangle, \langle s_1 \rangle]) = (-\frac{1}{2}, 1, \frac{1}{2})$
$f[s_2, s_2]$	5	6	10	$\mu(f[\langle s_2 \rangle, \langle s_2 \rangle]) = (-\frac{3}{2}, 1, \frac{3}{2})$
$f[s_2, s_3]$	5	6	10	$\mu(f[\langle s_2 \rangle, \langle s_3 \rangle]) = (-\frac{3}{2}, 1, \frac{3}{2})$
$f[s_3, s_1]$	4	5	9	$\mu(f[\langle s_3 \rangle, \langle s_1 \rangle]) = (-1, 1, 1)$
$f[s_3, s_2]$	5	6	10	$\mu(f[\langle s_3 \rangle, \langle s_2 \rangle]) = (-\frac{3}{2}, 1, \frac{3}{2})$
$f[s_3, s_3]$	6	7	11	$\mu(f[\langle s_3 \rangle, \langle s_3 \rangle]) = (-2, 1, 2)$

Figure 6: Applying $\mathrm{L}_*^{\mathrm{wta}}$ to the tree series ψ in Example 3.3

dead. It seems to be an interesting question whether this approach carries over the weighted setting in a reasonable way.

In the case of weighted tree automata, the learners mentioned in Section 3 seem to be the only ones known so far. In contrast, a variety of learning approaches for stochastic string languages and recognizable string series have been proposed in the

literature (see, e.g., [31, 9, 12, 10, 11, 14, 15, 25]). It could be interesting to see whether these approaches extend to stochastic tree languages or tree series as well.

Another question that may be worth studying is whether grammatical inference of tree series is easier if some aspects of the target series are already known, such as the support or the yield of the support.

Acknowledgment

I thank the anonymous referees for their critical comments on the overall contents and focus of this paper as well as for their help regarding details. Hopefully, I have been able to use these comments to the reader's advantage. Furthermore, I want to thank Brink van der Merwe for pointing out errors in the computations in Example 3.4.

References

[1] Adriaans, P. and van Zaanen, M. Computational grammar induction for linguists. *Formal Grammars*, 7:57–68, 2004. Electronic open access journal, see `http://grammars.grlmc.com/special.asp`.

[2] Angluin, D. A note on the number of queries needed to identify regular languages. *Information and Computation*, 51:76–87, 1981.

[3] Angluin, D. Learning regular sets from queries and counterexamples. *Information and Computation*, 75:87–106, 1987.

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

[5] Angluin, D. Negative results for equivalence queries. *Machine Learning*, 5:121–150, 1990.

[6] Becerra-Bonache, L., Dediu, A.H., and Tîrnăucă, C. Learning DFA from correction and equivalence queries. In Sakakibara, Y., Kobayashi, S., Sato, K., Nishino, T., and Tomita, E., editors, *8th Intl. Coll. Grammatical Inference: Algorithms and Applications (ICGI'06)*, volume 4201 of Lecture Notes in Computer Science, pages 281–292. Springer, 2006.

[7] Besombes, J. and Marion, J.-Y. Learning tree languages from positive examples and membership queries. *Theoretical Computer Science*, 382:183–197, 2007.

[8] Borchardt, B. The Myhill-Nerode theorem for recognizable tree series. In Ésik, Z. and Fülöp, Z., editors, *Proceedings of the 7th International Conference on Developments in Language Theory (DLT'03)*, volume 2710 of Lecture Notes in Computer Science, pages 146–158. Springer, 2003.

[9] Carrasco, R.C., Forcada, M.L., and Santamaría, L. Inferring stochastic regular grammars with recurrent neural networks. In Higuera, C. de la and Laurent, M., editors, *Proc. 3rd International Colloquium on Grammatical Inference (ICGI'96)*, volume 1147 of Lecture Notes in Artificial Intelligence, pages 274–281. Springer, 1996.

[10] Carrasco, R.C. and Oncina, J. Learning deterministic regular grammars from stochastic samples in polynomial time. *RAIRO Theoretical Informatics and Applications*, 33:1–20, 1999.

[11] Casacuberta, F. and de la Higuera, C. Computational complexity of problems on probabilistic grammars and transducers. In Oliviera, A.L., editor, *Proc. 5th International Colloquium on Grammatical Inference (ICGI'00)*, volume 1891 of Lecture Notes in Artificial Intelligence, pages 15–24. Springer, 2000.

[12] de la Higuera, C. Learning stochastic finite automata from experts. In Honavar, V. and Slutzki, G., editors, *Proc. 4th International Colloquium on Grammatical Inference (ICGI'98)*, volume 1433 of Lecture Notes in Artificial Intelligence, pages 79–89. Springer, 1998.

[13] de la Higuera, C. A bibliographical study of grammatical inference. *Pattern Recognition*, 38:1332–1348, 2005.

[14] Denis, F. and Esposito, Y. Learning classes of probabilistic automata. In Shawe-Taylor, J. and Singer, Y., editors, *Proc. 17th Annual Conference on Learning Theory (COLT'04)*, volume 3120 of Lecture Notes in Artificial Intelligence, pages 124–139. Springer, 2004.

[15] Denis, F., Esposito, Y., and Habrard, H. Learning rational stochastic languages. In Lugosi, G. and Simon, H.U., editors, *Proc. 19th Annual Conference on Learning Theory (COLT'06)*, volume 4005 of Lecture Notes in Artificial Intelligence, pages 274–288. Springer, 2006.

[16] Denis, F. and Habrard, H. Learning rational stochastic tree languages. In Hutter, M., Servedio, R.A., and Takimoto, E., editors, *Proc. 18th International Conference on Algorithmic Learning Theory (ALT'07)*, volume 4754 of Lecture Notes in Computer Science, pages 242–256. Springer, 2007.

[17] Drewes, F. and Högberg, J. Extensions of a MAT learner for regular tree languages. In Minock, M.J., Eklund, P., and Lindgren, H., editors, *Proc. 23rd Annual Workshop of the Swedish Artificial Intelligence Society (SAIS'06)*, pages 35–44, 2006.

[18] Drewes, F. and Högberg, J. Query learning of regular tree languages: How to avoid dead states. *Theory of Computing Systems*, 40:163–185, 2007.

[19] Drewes, F. and Vogler, H. Learning deterministically recognizable tree series. *Journal of Automata, Languages and Combinatorics*, 12:333–354, 2007.

[20] Fernau, H. Learning tree languages from text. In Kivinen, J. and Sloan, R. H., editors, *Proceedings of the 15th Annual Conference on Computational Learning Theory (COLT'02)*, volume 2375 of Lecture Notes in Artificial Intelligence, pages 153–168. Springer, 2002.

[21] Fernau, H. and de la Higuera, C. Grammar induction: An invitation for formal language theorists. *Formal Grammars*, 7:45–55, 2004. Electronic open access journal, see `http://grammars.grlmc.com/special.asp`.

[22] Fülöp, Z. and Vogler, H. Weighted tree automata and tree transducers. In Kuich, Werner, Droste, Manfred, and Vogler, H., editors, *Handbook of Weighted Automata*. Springer, 2009.

[23] Gold, E.M. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

[24] Gold, E.M. Complexity of automaton identification from given data. *Information and Control*, 37:302–320, 1978.

[25] Habrard, H., Denis, F., and Esposito, Y. Using pseudo-stochastic rational languages in probabilistic grammatical inference. In Sakakibara, Y., Kobayashi, S., Sato, K., Nishino, T., and Tomita, E., editors, *Proc. 8th International Colloquium on Grammatical Inference (ICGI'06)*, volume 4201 of Lecture Notes in Artificial Intelligence, pages 112–124. Springer, 2006.

[26] Habrard, H. and Oncina, J. Learning multiplicity tree automata. In Sakakibara, Y., Kobayashi, S., Sato, K., Nishino, T., and Tomita, E., editors, *Proc. 8th International Colloquium on Grammatical Inference (ICGI'06)*, volume 4201 of Lecture Notes in Artificial Intelligence, pages 268–280. Springer, 2006.

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

[28] Lee, L. Learning of context-free languages: A survey of the literature. Report TR-12-96, Harvard Univ., Center for Research in Computing Technology, 1996.

[29] López, D., Sempere, J.M., and Garcìa, P. Inference of reversible tree languages. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 34:1658–1665, 2004.

[30] Maletti, A. Learning deterministically recognizable tree series – revisited. In Bozapalidis, S. and Rahonis, G., editors, *Proc. 2nd International Conference on Algebraic Informatics (CAI'07)*, volume 4728 of Lecture Notes in Computer Science, pages 218–235. Springer, 2007.

[31] Ron, D., Singer, Y., and Tishby, N. On the learnability and usage of acyclic probabilistic finite automata. In *Proc. 8th Annual Conference on Learning Theory (COLT'95)*, pages 31–40. ACM Press, 1995.

[32] Sakakibara, Y. Learning context-free grammars from structural data in polynomial time. *Theoretical Computer Science*, 76:223–242, 1990.

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

[34] Sakakibara, Y. Recent advances of grammatical inference. *Theoretical Computer Science*, 185(1):15–45, 1997.

[35] Shapiro, E.Y. *Algorithmic Program Debugging.* MIT Press, Cambridge, MA, USA, 1983.

[36] Tîrnăucă, C. A note on the relationship between different types of correction queries. In Clark, A., Coste, F., and Miclet, L., editors, *Proc. 9th International Colloquium on Grammatical Inference (ICGI'08)*, volume 5278 of Lecture Notes in Artificial Intelligence, pages 213–223. Springer, 2008.

[37] Tîrnăucă, C. and Kobayashi, S. A characterization of the language classes learnable with correction queries. In Cai, J.-Y., Cooper, S.B., and Zhu, H., editors, *Proc. 4th Annual Conference on Theory and Applications of Models of Computation (TAMC'07)*, volume 4484 of Lecture Notes in Computer Science, pages 398–407. Springer, 2007.

[38] Tîrnăucă, C.I. and Tîrnăucă, C. Learning regular tree languages from correction and equivalence queries. *Journal of Automata, Languages and Combinatorics*, 12:501–524, 2007.

[39] Valiant, L.G. A theory of the learnable. *Communications of the ACM*, 27:1134–1142, 1984.

Received 11th July 2008