Learning Deterministically Recognizable Tree Series — Revisited

Andreas Maletti

Institute of Theoretical Computer Science, Faculty of Computer Science
Technische Universität Dresden

maletti@tcs.inf.tu-dresden.de

Abstract. We generalize a learning algorithm originally devised for deterministic all-accepting weighted tree automata (wta) to the setting of arbitrary deterministic wta. The learning is exact, supervised, and uses an adapted minimal adequate teacher; a learning model introduced by Angluin. Our algorithm learns a minimal deterministic wta that recognizes the taught tree series and runs in polynomial time in the size of that wta and the size of the provided counterexamples. Compared to the original algorithm, we show how to handle non-final states in the learning process; this problem was posed as an open problem in [Drewes, Vogler: *Learning Deterministically Recognizable Tree Series*, J. Autom. Lang. Combin. 2007].

1 Introduction

We devise a supervised learning algorithm for deterministically recognizable tree series. Learning algorithms for formal languages have a long and studied history (see the seminal and survey papers [16, 1, 3, 4]). The seminal paper [16] reports first results on identification in the limit; in particular it shows that every recursively enumerable language can be learned from a teacher. Here we study series; quantitative versions of languages. In particular, a tree series associates to each tree of T_Σ (the set of all well-formed expressions over the ranked alphabet Σ) a coefficient. Thus, it is nothing else than a mapping $\psi\colon T_\Sigma \to A$ for some suitable set A. It depends on A whether the coefficient represents, e.g., a probability, a count, a string, etc. For the moment, we assume that $(A, +, \cdot, 0, 1)$ is a field. A tree language $L \subseteq T_\Sigma$ can then be identified with the tree series that maps the elements of L to 1 and remaining elements of T_Σ to 0.

Angluin [2] proposed *query learning*, a model of interactive learning. In this learning model, the learner can question a teacher (or oracle). The teacher will answer predetermined types of questions. For example, the

minimally adequate teacher [2, 12] for a tree series $\psi\colon T_\Sigma \to A$ answers only two types of questions about ψ: *coefficient* and *equivalence queries*. A coefficient query asks for the coefficient of a certain tree t in the tree series ψ. The teacher truthfully supplies $\psi(t)$. Second, the learner can query the teacher whether his learned tree series φ coincides with ψ. The teacher either returns the special token $\perp$ to signal equality (i.e., $\varphi = \psi$) or he supplies a counterexample. Such a counterexample is a tree t on which φ and ψ disagree (i.e., $\varphi(t) \neq \psi(t)$).

Certainly, we need to be able to finitely represent the learned tree series. To this end, we use an automaton model called *(bottom-up) weighted tree automaton* (for short: wta; see [8] and the references therein). These devices are classical bottom-up tree automata [14, 15] with transition weights. The weights are elements of A and are combined using the operations $+$ and $\cdot$ of the field (see Definition 3). In [17], a learning algorithm based on the introduced minimally adequate teacher is presented for wta over fields. Here we will restrict ourselves to deterministic wta [8] and their recognized series, which are called deterministically recognizable. Since no general determinization procedure for wta over fields is known, this task is not encompassed by the result of [17].

For deterministic wta over fields (actually, semifields), the learning algorithm [12] was proposed. It is based on a restricted Myhill-Nerode theorem [12] for the series recognized by deterministic *all-accepting* (i.e., all states are final) wta (for short: aa-wta). Consequently, this algorithm learns the minimal deterministic aa-wta that recognizes ψ (which is unique up to renaming of states) provided that any deterministic aa-wta recognizing ψ exists. We extend this algorithm to arbitrary deterministic wta and solve the open problem of [12].

Let us discuss the main differences. First, an aa-wta M makes no distinction between final and non-final states because all of its states are final. In essence, the internal working of M is completely exposed to the outside. It yields that the recognized series ψ is *subtree-closed* [12]. This property demands that with every tree t such that $\psi(t) \neq 0$, also all of its subtrees are mapped (under ψ) to some nonzero weight. With this property, the weight of the last transition that is used to accept $t = \sigma(t_1, \ldots, t_k)$ can simply be computed as $\psi(t) \cdot \prod_{i=1}^{k} \psi(t_i)^{-1}$ (see Definition 11). Consequently, the minimal deterministic aa-wta recognizing ψ is unique (up to renaming of states).

On the contrary, there exists no unique minimal deterministic wta recognizing ψ because the weights on transitions that lead to non-final states can be varied (occassionally called *pushing* [13]). In summary, we need

to (i) distinguish final and non-final states and (ii) use a more complicated mechanism to compute the transition weights (because some $\psi(t_i)$ might be 0 in the above expression). The basis for our generalized learning will be the general MYHILL-NERODE theorem [5], which provides a characterization of the deterministically recognizable tree series by means of finite-index congruences of the initial term algebra (T_Σ, Σ). We then follow the approach of [5] and introduce a helping tree series (see Definition 10). The exact changes to the learning algorithm are discussed in the main body. Our new algorithm runs in time $\mathcal{O}(sm^2nr)$ where s is the size of the largest counterexample supplied by the teacher, m and n are the number of transitions and the number of states of the returned automaton, respectively, and r is the maximal rank of the input symbols.

Including this Introduction, the paper comprises 6 sections. The second section recalls basic notions and notations. In the next section, we recall wta and the MYHILL-NERODE theorem [5]. In Sect. 4, we present the main contribution of this paper, which is the generalized learning algorithm. Moreover, we prove its correctness and continue in Sect. 5 with an elaborated example run of the algorithm. In the last section, we discuss the runtime complexity of our new algorithm and compare it to the learning algorithm of [12].

2 Preliminaries

We write $\mathbb{N}$ to represent the nonnegative integers. Further, we write $[l, u]$ for $\{n \in \mathbb{N} \mid l \leqslant n \leqslant u\}$. Any nonempty and finite set Σ is an *alphabet*. A *ranked alphabet* is a partition $(\Sigma_k)_{k\in\mathbb{N}}$ of an alphabet Σ. For every ranked alphabet $\Sigma = (\Sigma_k)_{k\in\mathbb{N}}$, the set of *$\Sigma$-trees*, denoted by T_Σ, is inductively defined to be the smallest set T such that for every $\sigma \in \Sigma_k$ and $t_1, \ldots, t_k \in T$ also $\sigma(t_1, \ldots, t_k) \in T$. We write α instead of $\alpha()$ if $\alpha \in \Sigma_0$. Given a set $T \subseteq T_\Sigma$, the set $\{\sigma(t_1, \ldots, t_k) \mid \sigma \in \Sigma_k, t_1, \ldots, t_k \in T\}$ is denoted by $\Sigma(T)$. The size of a tree $t \in T_\Sigma$, denoted by $\mathrm{size}(t)$, is the number of occurrences of symbols of Σ in t. Let $\square \notin \Sigma$ be a distinguished nullary symbol. Let $\Sigma'_k = \Sigma_k$ for every $k > 0$ and $\Sigma'_0 = \Sigma_0 \cup \{\square\}$. A *$\Sigma$-context* c is a tree of $T_{\Sigma'}$ such that $\square$ occurs exactly once in c. The set of all Σ-contexts is denoted by C_Σ, and we write $c[t]$ for the tree that is obtained by replacing in $c \in C_\Sigma$ the occurrence of $\square$ with $t \in T_\Sigma$.

Let $\equiv$ be an equivalence on a set S. We write $[s]_\equiv$ for the equivalence class of $s \in S$ and $(S/\equiv)$ for $\{[s]_\equiv \mid s \in S\}$. We drop the subscript from $[s]_\equiv$ if $\equiv$ is clear. Finally, if $S = T_\Sigma$, then $\equiv$ is a *congruence* if for

every $\sigma \in \Sigma_k$ and $t_1, \ldots, t_k, u_1, \ldots, u_k \in T_\Sigma$ such that $t_i \equiv u_i$ for every $i \in [1,k]$ also $\sigma(t_1, \ldots, t_k) \equiv \sigma(u_1, \ldots, u_k)$.

A *(commutative) semiring* is an algebraic structure $(A, +, \cdot, 0, 1)$ comprising two commutative monoids $(A, +, 0)$ and $(A, \cdot, 1)$ such that $\cdot$ distributes over $+$ and 0 is absorbing for $\cdot$. A semiring $(A, +, \cdot, 0, 1)$ is a *semifield* if for every $a \in A \setminus \{0\}$ there exists an $a^{-1} \in A$ such that $a \cdot a^{-1} = 1$. A *tree series* ψ is a mapping $\psi\colon T_\Sigma \to A$; the set of all such mappings is denoted by $\mathcal{A}\langle\langle T_\Sigma \rangle\rangle$. Given $t \in T_\Sigma$, we denote $\psi(t)$ also by (ψ, t). The HADAMARD product of two tree series $\psi, \varphi \in \mathcal{A}\langle\langle T_\Sigma \rangle\rangle$ is denoted by $\psi \cdot \varphi$ and given by $(\psi \cdot \varphi, t) = (\psi, t) \cdot (\varphi, t)$ for every $t \in T_\Sigma$. Finally, a series $\psi \in \mathcal{A}\langle\langle T_\Sigma \rangle\rangle$ is *subtree-closed* if for every $t \in T_\Sigma$ with $(\psi, t) \neq 0$ also $(\psi, u) \neq 0$ for every subtree u of t.

3 Weighted Tree Automaton

In this section, we recall from [8, 5] the central notions of this contribution: deterministic weighted tree automata (wta) and deterministically recognizable tree series. For the rest of the paper, let $\mathcal{A} = (A, +, \cdot, 0, 1)$ be a commutative semifield; in examples we will use the field $\mathbb{R} = (\mathbb{R}, +, \cdot, 0, 1)$ of real numbers. In Sect. 4 we show how to learn a deterministic wta from a teacher using the characterization given by the MYHILL-NERODE theorem [5].

Definition 1 (see [8, Definitions 3.1 and 3.3]). *A* weighted tree automaton *M is a tuple $(Q, \Sigma, \mathcal{A}, F, \mu)$ with*

- *a finite set Q of* states*;*
- *a ranked alphabet Σ of* input symbols*;*
- *a set $F \subseteq Q$ of* final states*; and*
- *a* tree representation *$(\mu_k)_{k \in \mathbb{N}}$ such that $\mu_k\colon \Sigma_k \to A^{Q^k \times Q}$.*

We call M (bottom-up) deterministic *if for every symbol $\sigma \in \Sigma_k$ and $w \in Q^k$ there exists at most one $q \in Q$ such that $\mu_k(\sigma)_{w,q} \neq 0$.*

Note that a wta model with final weights (i.e., with $F\colon Q \to A$ instead of $F \subseteq Q$) is considered in [6]. However, for every deterministic "final weight" wta an equivalent deterministic wta can be constructed [7, Lemma 6.1.4]. Instead of $\mu_0(\alpha)_{\varepsilon,q}$ with $\alpha \in \Sigma_0$ and ε the empty word, we commonly write $\mu_0(\alpha)_q$.

Let us present our running-example wta. It is supposed to assign a probability to (simplified) syntax trees of simple English sentences. If the tree is ill-formed, then the assigned probability shall be 0. This shall

signal that it is rejected. Moreover, the probability shall diminish with the length of the input sentence.

Example 2. Let $\Sigma = (\Sigma_k)_{k\in\mathbb{N}}$ with $\bigcup_{k\in\mathbb{N}\setminus\{0,2\}} \Sigma_k = \emptyset$ and $\Sigma_2 = \{\sigma\}$ and $\Sigma_0 = \{\mathrm{Alice}, \mathrm{Bob}, \mathrm{loves}, \mathrm{hates}, \mathrm{ugly}, \mathrm{nice}, \mathrm{mean}, \mathrm{tall}\}$. Moreover, let $(Q, \Sigma, \mathbb{R}, F, \mu)$ be the deterministic wta with $\{\mathrm{NN}, \mathrm{VB}, \mathrm{ADJ}, \mathrm{NP}, \mathrm{VP}, \mathrm{S}\}$ as set Q of states and $F = \{\mathrm{S}\}$ and the nonzero tree representation entries

$$\begin{aligned}
0.5 &= \mu_0(\mathrm{Alice})_{\mathrm{NN}} = \mu_0(\mathrm{Bob})_{\mathrm{NN}} = \mu_0(\mathrm{loves})_{\mathrm{VB}} = \mu_0(\mathrm{hates})_{\mathrm{VB}} \\
0.25 &= \mu_0(\mathrm{ugly})_{\mathrm{ADJ}} = \mu_0(\mathrm{nice})_{\mathrm{ADJ}} = \mu_0(\mathrm{mean})_{\mathrm{ADJ}} = \mu_0(\mathrm{tall})_{\mathrm{ADJ}} \\
0.5 &= \mu_2(\sigma)_{\mathrm{NN\,VP,S}} = \mu_2(\sigma)_{\mathrm{NP\,VP,S}} = \mu_2(\sigma)_{\mathrm{VB\,NN,VP}} = \mu_2(\sigma)_{\mathrm{VB\,NP,VP}} \\
0.5 &= \mu_2(\sigma)_{\mathrm{ADJ\,NN,NP}} = \mu_2(\sigma)_{\mathrm{ADJ\,NP,NP}} \ .
\end{aligned}$$ ∎

In the sequel, we will sometimes abbreviate the nullary symbols used in Example 2 to just their initial letter. Let us continue with the semantics of wta.

Definition 3 (see [8, Definition 3.3]). *Let $M = (Q, \Sigma, \mathcal{A}, F, \mu)$ be a wta. The mapping $h_\mu \colon T_\Sigma \to A^Q$ is given by*

$$h_\mu(\sigma(t_1, \ldots, t_k))_q = \sum_{q_1\cdots q_k \in Q^k} \mu_k(\sigma)_{q_1\cdots q_k, q} \cdot h_\mu(t_1)_{q_1} \cdot \ldots \cdot h_\mu(t_k)_{q_k}$$

for every $\sigma \in \Sigma_k$, $q \in Q$, and $t_1, \ldots, t_k \in T_\Sigma$. The tree series that is recognized by *M, denoted by $S(M)$, is defined for every $t \in T_\Sigma$ by $(S(M), t) = \sum_{q\in F} h_\mu(t)_q$.*

We note that deterministic wta do not essentially use the additive operation. A tree series $\psi \in \mathcal{A}\langle\langle T_\Sigma \rangle\rangle$ is *deterministically recognizable* if there exists a deterministic wta M such that $S(M) = \psi$. Let us illustrate the definition of the semantics on a small example.

Example 4. Recall the deterministic wta M of Example 2. Then

$$\begin{aligned}
(S(M), \sigma(\mathrm{Alice}, \sigma(\mathrm{loves}, \mathrm{Bob}))) &= 3.125 \cdot 10^{-2} \\
(S(M), \sigma(\sigma(\mathrm{mean}, \mathrm{Bob}), \sigma(\mathrm{hates}, \sigma(\mathrm{ugly}, \mathrm{Alice})))) &= 4.8828125 \cdot 10^{-4} \\
(S(M), \sigma(\sigma(\mathrm{Alice}, \mathrm{loves}), \mathrm{Bob})) &= 0 \ .
\end{aligned}$$

Let us illustrate the computation of the last coefficient. To this end, let $t = \sigma(\sigma(\mathrm{A}, \mathrm{l}), \mathrm{B})$. Since S is the only final state of M, we obtain that $(S(M), t) = h_\mu(t)_{\mathrm{S}}$. We continue with

$$h_\mu(\sigma(\sigma(\mathrm{A}, \mathrm{l}), \mathrm{B}))_{\mathrm{S}}$$

$$= \sum_{q_1 q_2 \in Q^2} \mu_2(\sigma)_{q_1 q_2,\mathrm{S}} \cdot h_\mu(\sigma(\mathrm{A},\mathrm{l}))_{q_1} \cdot h_\mu(\mathrm{B})_{q_2}$$
$$= \sum_{q_1 \in Q} 0.5 \cdot h_\mu(\sigma(\mathrm{A},\mathrm{l}))_{q_1} \cdot \mu_0(\mathrm{B})_{\mathrm{VP}} = 0 \ .$$

We showed two parse trees for the sentence "Alice loves Bob". One of them is ill-formed and the other is assigned a positive probability. Thus, the sentence would not be considered ill-formed because a parse tree with nonzero weight exists. ∎

Let us conclude this section with the MYHILL-NERODE theorem [5] for deterministically recognizable tree series. Let $\psi \in \mathcal{A}\langle\langle T_\Sigma \rangle\rangle$. The MYHILL-NERODE congruence relation $\equiv_\psi \subseteq T_\Sigma \times T_\Sigma$ is given by $t \equiv_\psi u$ if and only if there exists a coefficient $a \in A \setminus \{0\}$ such that $(\psi, c[t]) = a \cdot (\psi, c[u])$ for every $c \in C_\Sigma$. Finally, by L_ψ we denote $\{t \in T_\Sigma \mid \forall c \in C_\Sigma \colon (\psi, c[t]) = 0\}$.

Theorem 5 (see [5, Theorem 2]). *A tree series $\psi \in \mathcal{A}\langle\langle T_\Sigma \rangle\rangle$ is deterministically recognizable if and only if $\equiv_\psi$ has finite index. Moreover, every minimal deterministic wta recognizing ψ has* $\mathrm{card}((T_\Sigma \setminus L_\psi)/\equiv_\psi)$ *states.*

4 Learning Algorithm

Next, we show how to learn a minimal deterministic wta for a given deterministically recognizable tree series ψ with the help of a teacher. To this end, we now fix a tree series $\psi \in \mathcal{A}\langle\langle T_\Sigma \rangle\rangle$. Let us clarify the role of the teacher. He is able to answer two types of questions:

1. *Coefficient queries:* Given $t \in T_\Sigma$, the teacher supplies (ψ, t).
2. *Equivalence queries:* Given a wta M, he answers whether $S(M) = \psi$. If so, he returns the special token $\perp$. Otherwise he returns a counterexample; i.e., some tree $t \in T_\Sigma$ such that $(S(M), t) \neq (\psi, t)$.

This straightforward adaptation of the *minimally adequate teacher* [2] was proposed in [12] and is based on the adaptation for tree languages [9, 11]. Equivalence queries might be considered unrealistic in a fully automatic setting and might there be replaced by tests that check a predetermined number of trees in applications. We will, however, not investigate the ramifications of this approximation. At this point, we will only note that equivalence of deterministic wta is decidable [6]. So in the particular case, that the teacher uses a deterministic wta to represent ψ, both types of queries can automatically be answered.

The following development is heavily inspired by the learning algorithm devised in [12]; in its turn an extension of the learning algorithm of [11] to deterministic *all-accepting* [12] wta. It was argued in [12] that the all-accepting property is no major restriction because any deterministically recognizable tree series ψ can be presented as the HADAMARD product of a series ψ' recognized by a deterministic all-accepting wta and a series ψ'' recognized by a deterministic BOOLEAN (i.e., only weights 0 and 1) wta (for the latter class, learning algorithms are known [9, 11]).

Let us discuss the problems of this approach. First, the decomposition is not unique; in general, we need to guess coefficients in ψ' (namely the ones where ψ is 0). The guessed coefficients affect the size of the minimal deterministic wta recognizing ψ'. Second, we learn minimial deterministic wta M' and M'' recognizing ψ' and ψ'', respectively, however, the HADAMARD product of M' and M'' is not necessarily a minimal deterministic wta recognizing $\psi = \psi' \cdot \psi''$. Third, we run two very similar algorithms and then perform a HADAMARD product construction; this is most likely not the most efficient solution.

The first problem (the completion of ψ to a subtree-closed ψ') can indeed be easily solved, provided that a representation of ψ by a deterministic wta is available (on the other hand, provided that a representation as deterministic wta is available, we could also just minimize the available representation). If no such representation is available, then the problem is far more complicated, and we will now show that very simple completions can even lead to deterministically non-recognizable tree series.

Example 6. Recall the wta M from Example 2. Clearly, $S(M)$ is not yet subtree-closed, so we complete it to $\psi' \in \mathbb{R}\langle\langle T_\Sigma \rangle\rangle$ (cf. Definition 10) by

$$(\psi', t) = \begin{cases} (S(M), t) & \text{if } (S(M), t) \neq 0 \\ 1 & \text{otherwise} \end{cases}$$

for every $t \in T_\Sigma$. We consider trees of the form $\sigma(\mathrm{m}, \sigma(\mathrm{m}, \ldots \sigma(\mathrm{m}, \mathrm{B}) \ldots))$. For every $n \in \mathbb{N}$, let t_n be the such obtained tree with n occurrences of m. Clearly, $(\psi', t_n) = 1$ for every $n \in \mathbb{N}$. Thus, for every $i, j \in \mathbb{N}$, we have $t_i \equiv_{\psi'} t_j$ if and only if $(\psi', c[t_i]) = (\psi', c[t_j])$ for every $c \in C_\Sigma$ because $(\psi', t_i) = (\psi', t_j)$. Now, we consider the context $c = \sigma(\mathrm{A}, \sigma(\mathrm{l}, \Box))$. An easy computation shows that $(\psi', c[t_n]) = 0.5^5 \cdot (0.5 \cdot 0.25)^n$ for every $n \in \mathbb{N}$. Consequently, $t_i \not\equiv_{\psi'} t_j$ whenever $i \neq j$. Consequently, $\equiv_{\psi'}$ has infinite index and thus ψ' is not deterministically recognizable by Theorem 5. ∎

Our main contribution is a slightly modified learning algorithm that is not restricted to deterministic *all-accepting* wta. To this end, we first

define a restriction of the MYHILL-NERODE congruence [5]. Henceforth, we will drop the index ψ from $\equiv_\psi$ and L_ψ.

Definition 7 (cf. [5, Sect. 5]). *Let $C \subseteq C_\Sigma$. The relation $\equiv_C$ contains all $(t, u) \in T_\Sigma \times T_\Sigma$ for which there exists an $a \in A \setminus \{0\}$ such that for every context $c \in C$ the equality $(\psi, c[t]) = a \cdot (\psi, c[u])$ holds.*

Clearly, $\equiv_C$ is an equivalence for every $C \subseteq C_\Sigma$. Moreover, the relation $\equiv_{C_\Sigma}$ coincides with the MYHILL-NERODE congruence [5] and for every $t, u \in T_\Sigma$ and $c \in C_\Sigma$ it holds that $t \equiv_{\{c\}} u$ if and only if $(\psi, c[t]) \neq 0$ precisely when $(\psi, c[u]) \neq 0$ (cf. Condition (MN2) in [5]). In particular, for $c = \Box$, we have that $t \equiv_{\{\Box\}} u$ if and only if both (ψ, t) and (ψ, u) are nonzero or both zero. Consequently, the context $\Box$ will allow us to distinguish final and non-final states. Finally, let

$$L_C = \{t \in T_\Sigma \mid \forall c \in C\colon (\psi, c[t]) = 0\}$$

for every $C \subseteq C_\Sigma$. Note that $L = L_{C_\Sigma}$.

An important observation is that there exists a finite set C of contexts such that $\equiv_C$ and $\equiv$ coincide, if $\equiv$ has finite index. Moreover, we note that for every $C \subseteq C_\Sigma$ the index of $\equiv_C$ is at most as large as the index of $\equiv$. Consequently, if $\equiv$ has finite index, then also $\equiv_C$ has finite index. Our learning strategy is to learn a set C of contexts such that $\equiv_C$ and $\equiv$ coincide. Next, we present our main data structure.

Definition 8 (cf. [12, Definition 4.3]). *We call a triple (E, T, C) an* observation table *if*

1. *E and T are finite subsets of T_Σ such that $E \subseteq T \subseteq \Sigma(E)$;*
2. *C is a subset of C_Σ with $\Box \in C$ and* $\mathrm{card}(C) \leq \mathrm{card}(E) + \mathrm{card}(T) + 1$*;*
3. *$T \cap L_C = \emptyset$; and*
4. $\mathrm{card}(E) = \mathrm{card}(E/{\equiv_C})$.

If, additionally, $\mathrm{card}(E) = \mathrm{card}(T/{\equiv_C})$*, then we call* (E, T, C) complete.

The only major difference to [12] is found in Condition 2. First, the presence of the context $\Box$ in C basically enables us to distinguish final and non-final states. There is no need for $\Box$ in [12] because all states will be final. Second, we changed the size restriction on C from $\mathrm{card}(C) \leq \mathrm{card}(E)$ (as in [12]) to $\mathrm{card}(C) \leq \mathrm{card}(E) + \mathrm{card}(T) + 1$. In [12], for every $e, e' \in E$, the coefficient a of Definition 7 (required to show that $e \equiv_C e'$) can always be determined with the help of the context $\Box$. Clearly, $\mathrm{card}(E)$ contexts are then sufficient to separate the elements of E. In our more general

setting, we cannot always determine the coefficient a of Definition 7 with the help of the context $\Box$. Rather, the contexts of C shall not only separate the elements of E, but shall also serve as explicit evidence that no tree in T (and thus also in E) is in L_C. This evidence is needed to determine the right coefficient in Definition 7 and is, consequently, used in Definition 10 to fix the right weight.

The third condition encodes the avoidance of trees t such that no supertree of t can be accepted (dead states; see [12]). This condition is only checked for those contexts that we accumulated in C.

Proposition 9. *Let $L \neq \emptyset$, and let $C \subseteq C_\Sigma$ be such that $\equiv_C$ coincides with $\equiv$. Then $L_C = L$.*

Proof. The direction $L \subseteq L_C$ is trivial. We prove the remaining direction by contradiction. Let $t \in L_C \setminus L$. Thus, $(\psi, c[t]) = 0$ for every $c \in C$, and clearly, $t \equiv_C u$ for every $u \in L$. However, there exists a context $c \in C_\Sigma \setminus C$ such that $(\psi, c[t]) \neq 0$ because $t \notin L$. This yields that $t \not\equiv u$ for every $u \in L$. Thus, we can derive the contradiction that $\equiv_C$ and $\equiv$ do not coincide because $L \neq \emptyset$. $\Box$

The condition $L \neq \emptyset$ is necessary in the above statement because the partition induced by $\equiv$ (and thus also $\equiv_C$) does not distinguish between an equivalence class containing only one tree, which happens to be in L, and an equivalence class containing only one tree, which is not in L.

The fourth and completeness condition in Definition 8 are equivalent to: $e \not\equiv_C e'$ for every two distinct $e, e' \in E$, and for every $t \in T$ there exists an $e \in E$ such that $t \equiv_C e$, respectively. Clearly, such an element e is uniquely determined by the former condition. In the sequel, given a complete observation table $\mathcal{T} = (E, T, C)$ and $t \in T$ we write $\mathcal{T}(t)$ for the unique $e \in E$ such that $e \equiv_C t$. Clearly, $\mathcal{T}(e) = e$ for every $e \in E$ (see [12, Lemma 4.4]). Next we show how to construct a deterministic wta given a complete observation table. To achieve this, we modify the construction [5] of a deterministic wta from the MYHILL-NERODE congruence $\equiv$.

Definition 10 (cf. [5, Lemma 8 and Page 9]). *Let $\mathcal{T} = (E, T, C)$ be a complete observation table. Let $\psi(\mathcal{T})\colon T_\Sigma \to A \setminus \{0\}$ be such that for every $t \in T_\Sigma$*

$$(\psi(\mathcal{T}), t) = \begin{cases} (\psi, t) & \text{if } (\psi, t) \neq 0 \\ (\psi, c[t]) \cdot (\psi, c[\mathcal{T}(t)])^{-1} & \text{if } (\psi, t) = 0 \text{ and } t \in T \text{ and} \\ & \quad (\psi, c[t]) \neq 0 \text{ for some } c \in C \\ 1 & \text{otherwise.} \end{cases}$$

Here, we only consider a baseline implementation; an efficient implementation could avoid many queries to the teacher [10] and store the required information in an extended observation table. Note that, for example, a suitable context for the second case in Definition 10 is observed by our algorithm (see Algorithms 1 and 3) when t is added to the observation table; it could thus be stored for efficient retrieval.

Some notes on the well-definedness of $\psi(\mathcal{T})$ are necessary. First, the condition $(\psi, t) \neq 0$ can be checked easily by a coefficient query. Second, $t \in T$ implies $t \notin L_C$ by the third condition of Definition 8. Thus, there trivially exists a context $c \in C$ such that $(\psi, c[t]) \neq 0$. It follows that $(\psi, c[\mathcal{T}(t)]) \neq 0$ because $t \equiv_C \mathcal{T}(t)$ and hence $t \equiv_{\{c\}} \mathcal{T}(t)$. Consequently, the inverse is well-defined. It remains to show that the result is independent of the selection of the context c. To this end, let $c' \in C$ be another context such that $(\psi, c'[t]) \neq 0$. Following the above argumentation, $(\psi, c'[\mathcal{T}(t)]) \neq 0$. Since $t \equiv_C \mathcal{T}(t)$, there exists a coefficient $a \in A \setminus \{0\}$ such that $(\psi, c''[t]) = a \cdot (\psi, c''[\mathcal{T}(t)])$ for every $c'' \in C$. It follows that

$$(\psi, c[t]) \cdot (\psi, c[\mathcal{T}(t)])^{-1} = a = (\psi, c'[t]) \cdot (\psi, c'[\mathcal{T}(t)])^{-1} \ .$$

Definition 11 (cf. [5, Definition 4]). *Let $\mathcal{T} = (E, T, C)$ be a complete observation table. We construct the wta $\mathcal{M}(\mathcal{T}) = (E, \Sigma, \mathcal{A}, F, \mu)$ such that*

- $F = \{e \in E \mid (\psi, e) \neq 0\}$;
- *for every $\sigma \in \Sigma_k$ and $e_1, \ldots, e_k \in E$ such that $\sigma(e_1, \ldots, e_k) \in T$*

$$\mu_k(\sigma)_{e_1 \cdots e_k, \mathcal{T}(\sigma(e_1, \ldots, e_k))} = (\psi(\mathcal{T}), \sigma(e_1, \ldots, e_k)) \cdot \prod_{i=1}^{k} (\psi(\mathcal{T}), e_i)^{-1}$$

- *and all remaining entries in μ are 0.*

Let us immediately observe some properties of the constructed wta. Clearly, $\mathcal{M}(\mathcal{T})$ is deterministic. Moreover, $S(\mathcal{M}(\mathcal{T}))$ coincides with ψ on all trees of T.

Lemma 12 (cf. [12, Lemma 4.5]). *Let $\mathcal{T} = (E, T, C)$ be a complete observation table. Then $(S(\mathcal{M}(\mathcal{T})), t) = (\psi, t)$ for every $t \in T$.*

Proof. Suppose that $\mathcal{M}(\mathcal{T}) = (E, \Sigma, \mathcal{A}, F, \mu)$. We first prove that

$$h_\mu(t)_{\mathcal{T}(t)} = (\psi(\mathcal{T}), t) \tag{1}$$

for every $t \in T$. Let $t = \sigma(t_1, \ldots, t_k)$ for some $\sigma \in \Sigma_k$ and $t_1, \ldots, t_k \in E$. By the induction hypothesis, we have $h_\mu(t_i)_{\mathcal{T}(t_i)} = (\psi(\mathcal{T}), t_i)$ for every

Algorithm 1 Learn a minimal deterministic wta recognizing ψ

```
    T ← (∅, ∅, {□})                          {initial observation table}
 2: loop
        M ← M(T)                              {construct new wta}
 4:     t ← EQUAL?(M)                         {ask equivalence query}
        if t = ⊥ then
 6:         return M                          {return the approved wta}
        else
 8:         T ← EXTEND(T, t)                  {extend the observation table}
```

$i \in [1,k]$. Clearly, $h_\mu(t_i)_e = 0$ for all states $e \in E$ with $e \neq \mathcal{T}(t_i)$ because $\mathcal{M}(\mathcal{T})$ is deterministic (see [8, Lemma 3.6]). Then

$$
\begin{aligned}
& h_\mu(\sigma(t_1,\ldots,t_k))_{\mathcal{T}(\sigma(t_1,\ldots,t_k))} \\
&= \sum_{e_1,\ldots,e_k \in E} \mu_k(\sigma)_{e_1\cdots e_k,\mathcal{T}(\sigma(t_1,\ldots,t_k))} \cdot \prod_{i=1}^{k} h_\mu(t_i)_{e_i} \\
&= \mu_k(\sigma)_{\mathcal{T}(t_1)\cdots\mathcal{T}(t_k),\mathcal{T}(\sigma(t_1,\ldots,t_k))} \cdot \prod_{i=1}^{k} (\psi(\mathcal{T}), t_i) \\
&= \mu_k(\sigma)_{t_1\cdots t_k,\mathcal{T}(\sigma(t_1,\ldots,t_k))} \cdot \prod_{i=1}^{k} (\psi(\mathcal{T}), t_i) \\
&= (\psi(\mathcal{T}), \sigma(t_1,\ldots,t_k)) \cdot \prod_{i=1}^{k} (\psi(\mathcal{T}), t_i)^{-1} \cdot \prod_{i=1}^{k} (\psi(\mathcal{T}), t_i) \\
&= (\psi(\mathcal{T}), \sigma(t_1,\ldots,t_k))
\end{aligned}
$$

where the second equality is by the induction hypothesis; the third is due to the fact that $t_1,\ldots,t_k \in E$; and the fourth is by the definition of μ (see Definition 11).

Thus, $h_\mu(t)_{\mathcal{T}(t)} \neq 0$. We now return to the main statement and complete the proof by distinguishing two cases: $(\psi,t) = 0$ and $(\psi,t) \neq 0$. In the former case, $(\psi,\mathcal{T}(t)) = 0$ because $\mathcal{T}(t) \equiv_C t$ and thus $\mathcal{T}(t) \equiv_{\{\square\}} t$ (since $\square \in C$). Consequently, $\mathcal{T}(t) \notin F$ and $(S(\mathcal{M}(\mathcal{T})),t) = 0$. In the latter case, an analoguous reasoning leads to $(\psi,\mathcal{T}(t)) \neq 0$ and $\mathcal{T}(t) \in F$. Consequently, $(S(\mathcal{M}(\mathcal{T})),t) = h_\mu(t)_{\mathcal{T}(t)} = (\psi(\mathcal{T}),t) = (\psi,t)$. □

In Algorithm 1 we show the principal structure of the learner. The bulk of work is done in EXTEND, which is shown in Algorithm 3. We start with the initial empty observation table $(\emptyset,\emptyset,\{\square\})$ and iteratively query the teacher for counterexamples and update our complete observation table

Algorithm 2 The COMPLETE function

Require: an observation table (E, T, C)

Ensure: return a complete observation table (E', T, C) such that $E \subseteq E'$

```
    for all t ∈ T do
2:      if t ≢_C e for every e ∈ E then
            E ← E ∪ {t}
4: return (E, T, C)
```

with the returned counterexample. Once the teacher approves our wta, we simply return it. Clearly, the returned wta recognizes ψ because the teacher certifies this. In Sect. 5 we show an example application of the learning algorithm to learn the series recognized by the wta of Example 2. We say that an algorithm works *correctly* if whenever the pre-conditions (Require) are met at the beginning of the algorithm, then the algorithm terminates and the post-conditions (Ensure) hold at the point of return.

Theorem 13 (see [12, Theorem 5.4]). *Let us suppose that* EXTEND *works correctly and ψ is deterministically recognizable. Then Algorithm 1 terminates and returns a minimal deterministic wta recognizing ψ.*

Proof. Let ψ be deterministically recognizable. Then $\equiv$ has finite index by Theorem 5. Let $l = \mathrm{card}(T_\Sigma/\equiv)$. We already remarked that, for every $C \subseteq C_\Sigma$, the index of $\equiv_C$ is at most l. This yields that for every observation table (E, T, C) we have $\mathrm{card}(E) \leqslant l$ because

$$\mathrm{card}(E) = \mathrm{card}(E/\equiv_C) \leqslant \mathrm{card}(T_\Sigma/\equiv_C) \leqslant \mathrm{card}(T_\Sigma/\equiv) = l \ .$$

It is easily checked that EXTEND is always called with a complete observation table and a counterexample as parameters. Since $\mathrm{card}(E)$ and $\mathrm{card}(T)$ are bounded, there can only be finitely many calls to EXTEND. Thus, Algorithm 1 terminates. Moreover, the returned wta, say $\mathcal{M}(\mathcal{T})$, is approved by the teacher, so we have $S(\mathcal{M}(\mathcal{T})) = \psi$. By the construction of $\mathcal{M}(\mathcal{T})$, we know that $\mathcal{M}(\mathcal{T})$ has at most l states. Consequently, $\mathcal{M}(\mathcal{T})$ is a minimal deterministic wta recognizing ψ by Theorem 5. □

Next, we describe the functionality of COMPLETE, which is shown in Algorithm 2. This function takes an observation table (E, T, C) and returns a complete observation table (E', T, C) with $E \subseteq E'$. We simply check for every $t \in T$ whether there exists an $e \in E$ such that $t \equiv_C e$. If this is not the case, then we add t to E. It is clear that COMPLETE works correctly.

Algorithm 3 The EXTEND function

Require: a complete observation table $\mathcal{T} = (E, T, C)$ and a counterexample $t \in T_\Sigma$

Ensure: return a complete observation table $\mathcal{T}' = (E', T', C')$ such that $E \subseteq E'$ and $T \subseteq T'$ and one inclusion is strict

Decompose t into $t = c[u]$ where $c \in C_\Sigma$ and $u \in \Sigma(E) \setminus E$
2: **if** $u \in T$ and $u \equiv_{C \cup \{c\}} \mathcal{T}(u)$ **then**
return EXTEND$(\mathcal{T}, c[\mathcal{T}(u)])$ {normalize and continue}
4: **else**
return COMPLETE$(E, T \cup \{u\}, C \cup \{c\})$ {add u and c}

Finally, let us discuss the EXTEND function, which is shown in Algorithm 3. We search for a minimal subtree that is still a counterexample using a technique called *contradiction backtracking* [18]. Let $\mathcal{T} = (E, T, C)$ be a complete observation table, $\mathcal{M}(\mathcal{T}) = (E, \Sigma, \mathcal{A}, F, \mu)$ be the constructed wta, and $t \in T_\Sigma$ be a counterexample; i.e., a tree t such that $(S(\mathcal{M}(\mathcal{T})), t) \neq (\psi, t)$. We first decompose t into a context $c \in C_\Sigma$ and a tree u that is not in E but whose direct subtrees are all in E. In some sense, this is a minimal offending subtree because the wta works correctly on all trees of T by Lemma 12. Moreover, such a subtree must exist because t is a counterexample.

Now we distinguish two cases. If u was already seen (i.e., $u \in T$), then $u \equiv_C \mathcal{T}(u)$. By Lemma 12, the wta $\mathcal{M}(\mathcal{T})$ works correctly on u. Thus the error is made when processing the context c. We test whether c separates u and $\mathcal{T}(u)$. Provided that $u \equiv_{C \cup \{c\}} \mathcal{T}(u)$, then u and $\mathcal{T}(u)$ behave equally in all contexts of $C \cup \{c\}$ and we continue our search for the counterexample with $c[\mathcal{T}(u)]$.

In all other cases, either u and $\mathcal{T}(u)$ should be separated or u was not seen before (i.e., is not already present in T). In the latter case, $h_\mu(u)_e = 0$, and consequently, also $h_\mu(c[u])_e = 0$ for every $e \in E$ (see [8, Lemma 3.7]). Hence $(S(\mathcal{M}(\mathcal{T})), c[u]) = 0$ and $(\psi, c[u]) \neq 0$. Thus we claim that in $\mathcal{T}' = (E, T \cup \{u\}, C \cup \{c\})$ is an observation table and return the completion of $\mathcal{T}'$. If $u \notin T$, then $(\psi, c[u]) \neq 0$ and thus $u \notin L_{C \cup \{c\}}$. Moreover, $\equiv_{C \cup \{c\}} \subseteq \equiv_C$ and either we add u to T (if $u \notin T$) or we add u to E (if $u \in T$ but $u \not\equiv_{C \cup \{c\}} \mathcal{T}(u)$). Thus the post-condition of the algorithm and the size restriction on the set of contexts are met.

The next lemma will rely on two straightforward lemmata; their proofs offer little insight and can thus be skipped on first reading.

Lemma 14 (see [5, Theorem 1]). *Let $M = (Q, \Sigma, \mathcal{A}, F, \mu)$ be a deterministic wta, and let $t, u \in T_\Sigma$ be such that $h_\mu(t)_p \neq 0$ and $h_\mu(u)_p \neq 0$*

for some state $p \in Q$. Then for every context $c \in C_\Sigma$ and state $q \in Q$

$$h_\mu(c[t])_q \cdot h_\mu(t)_p^{-1} = h_\mu(c[u])_q \cdot h_\mu(u)_p^{-1} \ .$$

Proof. We prove the statement by induction on the context c. Let $c = \Box$. Then $h_\mu(c[t])_q = h_\mu(t)_q$ and $h_\mu(c[u])_q = h_\mu(u)_q$. We now distinguish two cases: (i) $q = p$ and (ii) $q \neq p$. In the former case, we immediately obtain

$$h_\mu(t)_p \cdot h_\mu(t)_p^{-1} = 1 = h_\mu(u)_p \cdot h_\mu(u)_p^{-1} \ .$$

In the latter case, $h_\mu(t)_q = 0 = h_\mu(u)_q$ because M is deterministic (see [8, Lemma 3.6]). Consequently,

$$h_\mu(t)_q \cdot h_\mu(t)_p^{-1} = 0 = h_\mu(u)_q \cdot h_\mu(u)_p^{-1} \ .$$

In the induction step we assume that $c = \sigma(t_1, \ldots, t_{i-1}, c', t_{i+1}, \ldots, t_k)$ for some $\sigma \in \Sigma_k$, context $c' \in C_\Sigma$, position $i \in [1, k]$, and $t_1, \ldots, t_k \in T_\Sigma$. Then

$$\begin{aligned}
&h_\mu(\sigma(t_1, \ldots, t_{i-1}, c', t_{i+1}, \ldots, t_k)[t])_q \cdot h_\mu(t)_p^{-1} \\
&= h_\mu(\sigma(t_1, \ldots, t_{i-1}, c'[t], t_{i+1}, \ldots, t_k))_q \cdot h_\mu(t)_p^{-1} \\
&= \Big(\sum_{q_1, \ldots, q_k \in Q} \mu_k(\sigma)_{q_1 \cdots q_k, q} \cdot h_\mu(c'[t])_{q_i} \cdot \prod_{i \in [1,k] \setminus \{i\}} h_\mu(t_i)_{q_i} \Big) \cdot h_\mu(t)_p^{-1} \\
&= \Big(\sum_{q_1, \ldots, q_k \in Q} \mu_k(\sigma)_{q_1 \cdots q_k, q} \cdot h_\mu(c'[u])_{q_i} \cdot \prod_{i \in [1,k] \setminus \{i\}} h_\mu(t_i)_{q_i} \Big) \cdot h_\mu(u)_p^{-1} \\
&= h_\mu(\sigma(t_1, \ldots, t_{i-1}, c'[u], t_{i+1}, \ldots, t_k))_q \cdot h_\mu(u)_p^{-1} \\
&= h_\mu(\sigma(t_1, \ldots, t_{i-1}, c', t_{i+1}, \ldots, t_k)[u])_q \cdot h_\mu(u)_p^{-1}
\end{aligned}$$

where the third equality holds by the induction hypothesis and distributivity. □

Lemma 15. *Let $\mathcal{T} = (E, T, C)$ be an observation table, and let $t, u \in T$ be such that $t \equiv_C u$. For every $c \in C$*

$$(\psi, c[t]) \cdot (\psi(\mathcal{T}), t)^{-1} = (\psi, c[u]) \cdot (\psi(\mathcal{T}), u)^{-1} \ .$$

Proof. By $t \equiv_C u$ there exists an $a \in A \setminus \{0\}$ such that for every context $c' \in C$ we have $(\psi, c'[t]) = a \cdot (\psi, c'[u])$. Consequently,

$$(\psi, t) = a \cdot (\psi, u) \quad \text{and} \quad (\psi, c[t]) = a \cdot (\psi, c[u]) \ . \tag{2}$$

1. First, let $(\psi, c[t]) = 0$. By (2) also $(\psi, c[u]) = 0$, which proves the statement.
2. Second, let $(\psi, c[t]) \neq 0$ and $(\psi, t) \neq 0$. Then we can again conclude with the help of (2) that $(\psi, c[u]) \neq 0$ and $(\psi, u) \neq 0$. Further,

$$\begin{aligned}&(\psi, c[t]) \cdot (\psi(\mathcal{T}), t)^{-1} = (\psi, c[t]) \cdot (\psi, t)^{-1} = (\psi, c[u]) \cdot (\psi, u)^{-1} \\ &= (\psi, c[u]) \cdot (\psi(\mathcal{T}), u)^{-1}\end{aligned}$$

   where the second equality holds by (2).
3. Finally, let $(\psi, c[t]) \neq 0$ and $(\psi, t) = 0$. We again immediately note that $(\psi, c[u]) \neq 0$ and $(\psi, u) = 0$ by (2). Since $t, u \notin L_C$,

$$\begin{aligned}&(\psi, c[t]) \cdot (\psi(\mathcal{T}), t)^{-1} = (\psi, c[t]) \cdot \big((\psi, c[t]) \cdot (\psi, c[\mathcal{T}(t)])^{-1}\big)^{-1} \\ &= (\psi, c[\mathcal{T}(t)]) = (\psi, c[\mathcal{T}(u)]) \\ &= (\psi, c[u]) \cdot \big((\psi, c[u]) \cdot (\psi, c[\mathcal{T}(u)])^{-1}\big)^{-1} = (\psi, c[u]) \cdot (\psi(\mathcal{T}), u)^{-1}\end{aligned}$$

   where the third equality holds because $t \equiv_C u$. □

Now we are ready with the two auxiliary lemmata. It remains to prove that the recursive call of Extend meets the pre-conditions of Extend. It is clear, that $\mathcal{T}$ is a complete observation table, but we need to prove that $c[\mathcal{T}(u)]$ is also a counterexample. This is achieved in the next lemma.

Lemma 16. *Let $\mathcal{T} = (E, T, C)$ be a complete observation table, $u \in T$, and $c \in C_\Sigma$ such that $u \equiv_{C \cup \{c\}} \mathcal{T}(u)$. If $(S(\mathcal{M}(\mathcal{T})), c[u]) \neq (\psi, c[u])$, then also $(S(\mathcal{M}(\mathcal{T})), c[\mathcal{T}(u)]) \neq (\psi, c[\mathcal{T}(u)])$.*

Proof. Let $\mathcal{M}(\mathcal{T}) = (E, \Sigma, \mathcal{A}, F, \mu)$. We distinguish two cases: First, let $h_\mu(c[u])_q = 0$ for every $q \in Q$. Then also $h_\mu(c[\mathcal{T}(u)])_q = 0$ for every $q \in Q$ because $\mathcal{M}(\mathcal{T})$ is deterministic and $h_\mu(u)_{\mathcal{T}(u)} \neq 0$ and $h_\mu(\mathcal{T}(u))_{\mathcal{T}(u)} \neq 0$ by Lemma 12 (see also Lemma 14). Clearly, $(S(\mathcal{M}(\mathcal{T})), c[u]) = 0$ and $(S(\mathcal{M}(\mathcal{T})), c[\mathcal{T}(u)]) = 0$. Consequently, $(\psi, c[u]) \neq 0$ and $(\psi, c[\mathcal{T}(u)]) \neq 0$ because $u \equiv_{C \cup \{c\}} \mathcal{T}(u)$. This proves the statement in the first case.

Second, let $q \in Q$ be such that $h_\mu(c[u])_q \neq 0$. Note that $h_\mu(u)_{\mathcal{T}(u)} \neq 0$ and $h_\mu(\mathcal{T}(u)_{\mathcal{T}(u)} \neq 0$ by Lemma 12. Then

$$\begin{aligned}&(S(\mathcal{M}(\mathcal{T})), c[u]) \cdot h_\mu(u)^{-1}_{\mathcal{T}(u)} = \sum_{p \in \{q\} \cap F} h_\mu(c[u])_p \cdot h_\mu(u)^{-1}_{\mathcal{T}(u)} \\ &= \sum_{p \in \{q\} \cap F} h_\mu(c[\mathcal{T}(u)])_p \cdot h_\mu(\mathcal{T}(u))^{-1}_{\mathcal{T}(u)} \\ &= (S(\mathcal{M}(\mathcal{T})), c[\mathcal{T}(u)]) \cdot h_\mu(\mathcal{T}(u))^{-1}_{\mathcal{T}(u)} \qquad (3)\end{aligned}$$

where the second equality is by Lemmata 12 and 14. We now reason as follows.

$$
\begin{aligned}
& (S(\mathcal{M}(\mathcal{T})), c[\mathcal{T}(u)]) \\
&= (S(\mathcal{M}(\mathcal{T})), c[u]) \cdot h_\mu(\mathcal{T}(u))_{\mathcal{T}(u)} \cdot h_\mu(u)^{-1}_{\mathcal{T}(u)} && \text{by (3)} \\
&= (S(\mathcal{M}(\mathcal{T})), c[u]) \cdot (\psi(\mathcal{T}), \mathcal{T}(u)) \cdot (\psi(\mathcal{T}), u)^{-1} && \text{by (1)} \\
&\neq (\psi, c[u]) \cdot (\psi(\mathcal{T}), \mathcal{T}(u)) \cdot (\psi(\mathcal{T}), u)^{-1} \\
&= (\psi, c[\mathcal{T}(u)]) && \text{by Lemma 15} \qquad \square
\end{aligned}
$$

The previous lemma justifies the recursive call of EXTEND. It remains to check whether the recursion terminates (see [12, Lemma 5.3]). For this we consider a call EXTEND$(\mathcal{T}, t)$ triggered in Line 8 of Algorithm 1. Since the recursive call of EXTEND also has $\mathcal{T}$ as parameter, we now fix a complete observation table $\mathcal{T} = (E, T, C)$ for all invocations of EXTEND that are triggered by the considered call EXTEND$(\mathcal{T}, t)$. Moreover, let $v\colon T_\Sigma \to \mathbb{N}$ be the mapping that assigns to every $u \in T_\Sigma$ the number of occurrences of subtrees of u that are not in E. Next we show that every call in our chain of invocations strictly decreases $v(t)$ where t is the second parameter of the call to EXTEND. Suppose we now consider the call EXTEND$(\mathcal{T}, t)$, and let $t = c[u]$ be the decomposition as given in Line 1 of Algorithm 3. Without regard of the occurrence of the recursive call to EXTEND, it is of the form EXTEND$(\mathcal{T}, c[\mathcal{T}(u)])$. By Line 1 in Algorithm 3 we have $u \in \Sigma(E) \setminus E$. So $v(t) = \text{size}(c)$ and $v(c[\mathcal{T}(u)]) = \text{size}(c) - 1$ because $\mathcal{T}(u) \in E$ and E is trivially subtree-closed (i.e., if $e \in E$ then also all subtrees of e are in E). Thus the recursion must terminate and hence each call of EXTEND terminates.

Corollary 17 (of Theorem 13). *Provided that ψ is deterministically recognizable, Algorithm 1 terminates and returns a minimal deterministic wta recognizing ψ.*

5 An example

Let us show, how the algorithm learns the tree series ψ recognized by the wta of Example 2. We start (Line 1) with the initial empty observation table $\mathcal{T}_0 = (\emptyset, \emptyset, \{\Box\})$. The constructed (Line 3) wta $M_0 = (\emptyset, \Sigma, \mathcal{A}, \emptyset, \mu)$ recognizes the tree series that maps every tree to 0. We have seen in Example 4 that $(\psi, \sigma(\mathrm{A}, \sigma(\mathrm{l}, \mathrm{B}))) = 3.125 \cdot 10^{-2}$, so suppose the equivalence query (Line 4) is answered with $t_1 = \sigma(\mathrm{A}, \sigma(\mathrm{l}, \mathrm{B}))$. Consequently,

we will call $\textsc{Extend}(\mathcal{T}_0, t_1)$. Inside the call, we first decompose t_1 into $c_1 = \sigma(\Box, \sigma(\mathrm{l}, \mathrm{B}))$ and $u_1 = \mathrm{A}$. Consequently, we return

$$\textsc{Complete}(\emptyset, \{u_1\}, \{\Box, c_1\}) = (\{u_1\}, \{u_1\}, \{\Box, c_1\}) = \mathcal{T}_1$$

in Line 8 of Algorithm 3. We thus finished the first loop in Algorithm 1.

The wta $\mathcal{M}(\mathcal{T}_1)$ will only have the non-final state A and the nonzero tree representation entry $\mu_0(\mathrm{A})_\mathrm{A} = 1$. Hence t_1 is still a counterexample, and we might assume that t_1 is returned by the teacher. Thus we call $\textsc{Extend}(\mathcal{T}_1, t_1)$. There we first decompose t_1 into the context $c_2 = \sigma(\mathrm{A}, \sigma(\Box, \mathrm{B}))$ and $u_2 = \mathrm{l}$. Consequently, the call returns

$$\textsc{Complete}(\{u_1\}, \{u_1, u_2\}, \{\Box, c_1, c_2\}) = (\{u_1, u_2\}, \{u_1, u_2\}, \{\Box, c_1, c_2\})$$

because $(\psi, \sigma(\mathrm{l}, \sigma(\mathrm{l}, \mathrm{B}))) = 0$. Let $\mathcal{T}_2$ be the complete observation table displayed above. This concludes the second iteration.

In the third iteration, we can still use t_1 as counterexample and the decomposition $c_3 = \sigma(\mathrm{A}, \sigma(\mathrm{l}, \Box))$ and $u_3 = \mathrm{B}$. The call to $\textsc{Extend}$ then returns $\mathcal{T}_3 = (\{u_1, u_2\}, \{u_1, u_2, u_3\}, \{\Box, c_1, c_2, c_3\})$ because we have $\mathrm{A} \equiv_{\{\Box, c_1, c_2, c_3\}} \mathrm{B}$. Another iteration with the counterexample t_1 again yields the decomposition c_3 and u_3. Now u_3 was already seen before and $\mathrm{A} \equiv_{\{\Box, c_1, c_2, c_3\}} \mathrm{B}$, so we return $\textsc{Extend}(\mathcal{T}_3, \sigma(\mathrm{A}, \sigma(\mathrm{l}, \mathrm{A})))$. In that call, we decompose the second argument into $c_4 = \sigma(\mathrm{A}, \Box)$ and $u_4 = \sigma(\mathrm{l}, \mathrm{A})$ and return

$$\begin{aligned}&\textsc{Complete}(\{u_1, u_2\}, \{u_1, \ldots, u_4\}, \{\Box, c_1, \ldots, c_4\})\\ &= (\{u_1, u_2, u_4\}, \{u_1, \ldots, u_4\}, \{\Box, c_1, \ldots, c_4\}) = \mathcal{T}_4 \ .\end{aligned}$$

We will not demonstrate the construction of the wta $\mathcal{M}(\mathcal{T}_4)$ but will give an elaborate example at the end. For the moment, rest assured that t_1 is still a counterexample (because $\mathcal{M}(\mathcal{T}_4)$ has no final states). The decomposition of t_1 will be c_3 and u_3. As previously, this yields the recursive call $\textsc{Extend}(\mathcal{T}_4, \sigma(\mathrm{A}, \sigma(\mathrm{l}, \mathrm{A})))$. Now the decomposition will be $c_5 = \Box$ and $u_5 = \sigma(\mathrm{A}, \sigma(\mathrm{l}, \mathrm{A}))$ and $\textsc{Extend}$ will return

$$\begin{aligned}&\textsc{Complete}(\{u_1, u_2, u_4\}, \{u_1, \ldots, u_5\}, \{\Box, c_1, \ldots, c_4\})\\ &= (\{u_1, u_2, u_4, u_5\}, \{u_1, \ldots, u_4\}, \{\Box, c_1, \ldots, c_4\}) = \mathcal{T}_5 \ .\end{aligned}$$

Note that u_5 is a final state of $\mathcal{M}(\mathcal{T}_5)$ and that t_1 is no longer a counterexample. If we continue with $t_2 = \sigma(\mathrm{A}, \sigma(\mathrm{h}, \sigma(\mathrm{u}, \mathrm{B})))$ until it is no longer a counterexample, then we obtain

$$\begin{aligned}\mathcal{T}_8 = (&\{u_1, u_2, u_4, u_5, \mathrm{u}\}, \{u_1, \ldots, u_5, \mathrm{h}, \mathrm{u}, \sigma(\mathrm{u}, \mathrm{A})\},\\ &\{\Box, c_1, \ldots, c_4, \sigma(u_1, \sigma(\Box, \sigma(\mathrm{u}, u_3))), \sigma(u_1, \sigma(u_2, \sigma(\Box, u_3)))\}) \ .\end{aligned}$$

Next we select $t_3 = \sigma(\sigma(\mathrm{t}, \sigma(\mathrm{m}, \mathrm{A})), \sigma(\mathrm{l}, \sigma(\mathrm{n}, \mathrm{B})))$ as counterexample and continue in the same manner. We obtain $\mathcal{T}_{11}$ as

$$(\{\mathrm{A}, \mathrm{l}, \sigma(\mathrm{l}, \mathrm{A}), \sigma(\mathrm{A}, \sigma(\mathrm{l}, \mathrm{A})), \mathrm{u}\}, \{u_1, \ldots, u_5, \mathrm{h}, \mathrm{u}, \sigma(\mathrm{u}, \mathrm{A}), \mathrm{m}, \mathrm{t}, \mathrm{n}\}, C')$$

for some $C' \subseteq C_\Sigma$. At last, let us construct the wta $\mathcal{M}(\mathcal{T}_{11})$. By Definition 11 we obtain the wta $(Q, \Sigma, \mathcal{A}, F, \mu)$ with

- $Q = \{\mathrm{A}, \mathrm{l}, \sigma(\mathrm{l}, \mathrm{A}), \sigma(\mathrm{A}, \sigma(\mathrm{l}, \mathrm{A})), \mathrm{u}\}$
- $F = \{\sigma(\mathrm{A}, \sigma(\mathrm{l}, \mathrm{A}))\}$; and
- the nonzero tree representation entries

$$\begin{aligned}
1 &= \mu_0(\mathrm{A})_\mathrm{A} = \mu_0(\mathrm{B})_\mathrm{A} = \mu_0(\mathrm{l})_\mathrm{l} = \mu_0(\mathrm{h})_\mathrm{l} \\
1 &= \mu_0(\mathrm{n})_\mathrm{u} = \mu_0(\mathrm{t})_\mathrm{u} = \mu_0(\mathrm{u})_\mathrm{u} = \mu_0(\mathrm{m})_\mathrm{u} \\
1 &= \mu_2(\sigma)_{\mathrm{l}\,\mathrm{A},\sigma(\mathrm{l},\mathrm{A})} \\
0.125 &= \mu_2(\sigma)_{\mathrm{u}\,\mathrm{A},\mathrm{A}} \\
0.03125 &= \mu_2(\sigma)_{\mathrm{A}\,\sigma(\mathrm{l},\mathrm{A}),\sigma(\mathrm{A},\sigma(\mathrm{l},\mathrm{A}))} \quad .
\end{aligned}$$

Clearly, $\mathcal{M}(\mathcal{T}_{11})$ recognizes exactly ψ. In the next iteration, the teacher thus approves $\mathcal{M}(\mathcal{T}_{11})$. The returned wta has only 5 states (compared to the 6 states of the wta in Example 2). By Corollary 17 the returned wta is minimal. Thus, the learning algorithm might also be used to minimize deterministic wta but it is rather inefficient at that task.

6 Complexity analysis

Our formal runtime complexity analysis follows the approach of [11]. In [12] a similar analysis is outlined but not actually shown. Our computation model will be the random access machine and we assume that the multiplicative semifield operations (including taking the inverse and equality tests) and the queries to the teacher can be performed in constant time. Finally, we assume that the algorithm terminates with the deterministic wta $(Q, \Sigma, \mathcal{A}, F, \mu)$. In the sequel, let

$$m = \mathrm{card}(\{(\sigma, q, q_1, \ldots, q_k) \mid \mu_k(\sigma)_{q_1 \cdots q_k, q} \neq 0\})$$

and $n = \mathrm{card}(Q)$. Let $r = \max\{k \mid \Sigma^{(k)} \neq \emptyset\}$, and let $\mathcal{T} = (E, T, C)$ be a complete observation table encountered during the run of the algorithm. Let us start with the complexity of Complete.

Proposition 18 (cf. [11, Lemma 4.7]). *Within time $\mathcal{O}(mn)$ the call* Complete$(E, T \cup \{u\}, C \cup \{c\})$ *returns.*

Proof. First we check for each $t \in T \setminus E$ whether the new context c splits t and $\mathcal{T}(t)$; i.e., whether $t \equiv_{C \cup \{c\}} \mathcal{T}(t)$. Suppose that with each $t \in T \setminus E$ we store the coefficient a required in Definition 7 for $t \equiv_C \mathcal{T}(t)$. Now we simply need to check whether this coefficient also qualifies for $t \equiv_{\{c\}} \mathcal{T}(t)$. These simple checks require $\mathcal{O}(m)$ because $\mathrm{card}(T) \leq m$.

Should the check fail for some t_1 and t_2 that previously have been in the same equivalence class, then we need to compare them to each other. For each t these comparisons can amount up to $\mathcal{O}(n)$ because $\mathrm{card}(E) \leq n$.

Now it only remains to classify the new tree u provided that $u \notin T$. We simply compare u to each identified representative. Clearly, this requires us to check all contexts $C \cup \{c\}$. This takes $\mathcal{O}(n(m+n))$, which can also be given as $\mathcal{O}(2mn)$ because $n \leq m$. Thus the overall complexity is $\mathcal{O}(mn)$. □

With the previous proposition we can state the complexity of a call to EXTEND.

Proposition 19 (cf. [11, Lemma 4.6]). *The call* EXTEND$(\mathcal{T}, t)$ *returns in time* $\mathcal{O}(\mathrm{size}(t)mnr)$.

Proof. We already argued that at most $\mathrm{size}(t)$ recursive calls might be triggered by this call. In each invocation, we need to perform the decomposition into $c[u]$. In [11, Lemma 4.5], it is shown how this can be achieved in time $\mathcal{O}(nr)$. Thus it is also in $\mathcal{O}(mr)$. Using a similar technique, we can also test whether $u \in T$ in time $\mathcal{O}(mr)$. Finally, if $u \in T$, then the check $u \equiv_{C \cup \{c\}} \mathcal{T}(u)$ can be performed in constant time because we can assume that a pointer to $\mathcal{T}(u)$ and the required coefficient for Definition 7 is stored with u. Thus, we only need to confirm that coefficient for the new context c. Altogether, this yields that the call to EXTEND returns in time $\mathcal{O}(\mathrm{size}(t)mnr)$. □

Proposition 20 (cf. [11, Lemma 4.7]). *The wta* $\mathcal{M}(\mathcal{T})$ *can be constructed in time* $\mathcal{O}(mr)$.

Let s be the size of the largest counterexample returned by the teacher. Our simple and straightforward complexity analysis yields the following overall complexity (cf. $\mathcal{O}(mn^2(n+s)r)$ for the algorithm of [12]).

Theorem 21. *Our devised learning algorithm runs in time* $\mathcal{O}(sm^2nr)$.

Proof. We already saw that at most $m + n \leq 2m$ calls to EXTEND can happen before termination. Thus, we obtain the statement. □

Acknowledgements

The author would like to thank Heiko Vogler and Frank Drewes for lively discussions. Further, the author wants to express cordial thanks to the referees of the draft version of this paper. Their insight and criticism enabled the author to improve the paper.

References

1. Dana Angluin. Inductive inference of formal languages from positive data. *Inform. and Control*, 45(2):117–135, 1980.
2. Dana Angluin. Learning regular sets from queries and counterexamples. *Inform. and Comput.*, 75(2):87–106, 1987.
3. Dana Angluin. Queries and concept learning. *Machine Learning*, 2(4):319–342, 1987.
4. Dana Angluin. Queries revisited. In *Proc. 12th Int. Conf. Algorithmic Learning Theory*, volume 2225 of LNCS, pages 12–31. Springer, 2001.
5. Björn Borchardt. The Myhill-Nerode theorem for recognizable tree series. In *Proc. 7th Int. Conf. Developments in Language Theory*, volume 2710 of LNCS, pages 146–158. Springer, 2003.
6. Björn Borchardt. A pumping lemma and decidability problems for recognizable tree series. *Acta Cybernet.*, 16(4):509–544, 2004.
7. Björn Borchardt. *The Theory of Recognizable Tree Series*. PhD thesis, Technische Universität Dresden, 2005.
8. Björn Borchardt and Heiko Vogler. Determinization of finite state weighted tree automata. *J. Autom. Lang. Combin.*, 8(3):417–463, 2003.
9. Frank Drewes and Johanna Högberg. Learning a regular tree language from a teacher. In *Proc. 7th Int. Conf. Developments in Language Theory*, volume 2710 of LNCS, pages 279–291. Springer, 2003.
10. Frank Drewes and Johanna Högberg. Extensions of a MAT learner for regular tree languages. In *Proc. 23rd Annual Workshop of the Swedish Artificial Intelligence Society*, pages 35–44. Umeå University, 2006.
11. Frank Drewes and Johanna Högberg. Query learning of regular tree languages: How to avoid dead states. *Theory of Comput. Syst.*, 40(2):163–185, 2007.
12. Frank Drewes and Heiko Vogler. Learning deterministically recognizable tree series. *J. Automata, Languages and Combinatorics*, 2007. to appear.
13. Jason Eisner. Simpler and more general minimization for weighted finite-state automata. In *Human Language Technology Conf. of the North American Chapter of the Association for Computational Linguistics*, pages 64–71, 2003.
14. Ferenc Gécseg and Magnus Steinby. *Tree Automata*. Akadémiai Kiadó, Budapest, 1984.
15. Ferenc Gécseg and Magnus Steinby. Tree languages. In *Handbook of Formal Languages*, volume 3, chapter 1, pages 1–68. Springer, 1997.
16. E. Mark Gold. Language identification in the limit. *Inform. and Control*, 10(5):447–474, 1967.
17. Amaury Habrard and Jose Oncina. Learning multiplicity tree automata. In *Proc. 8th Int. Colloquium Grammatical Inference*, volume 4201 of LNAI, pages 268–280. Springer, 2006.
18. Ehud Y. Shapiro. *Algorithmic Program Debugging*. ACM Distinguished Dissertation. MIT Press, 1983.