JMLR: Workshop and Conference Proceedings 1–30

Polynomial inference of universal automata from membership and equivalence queries

Johanna Björklund johanna@cs.umu.se
Universitet Umeå, Department of Computing Science, S-90750 Umeå, Sweden

Henning Fernau and **Anna Kasprzik** fernau,kasprzik@informatik.uni-trier.de
Universität Trier, FB IV—Abteilung Informatik, D-54286 Trier, Germany

Abstract

We present a MAT learning algorithm that infers the universal automaton for a regular target language using a polynomial number of queries with respect to that automaton. The universal automaton is one of the numerous canonical characterizations for regular languages. Our learner is based on the concept of an observation table, and we adapt the necessary notions and definitions from the literature to the case of universal automata.

1. Introduction

Grammatical Inference

The area of Grammatical Inference (GI) is concerned with algorithms that extrapolate from limited information to infer a formal description of an unknown language. An important concept in this context is the convergence to a certain partition of the target language, which is obtained by splitting and merging sets (or, from the automaton perspective: states). In this paper, we present an algorithm with the objective of inferring the *universal automaton* (UA) for the language in question, and in doing so we restrict our attention to automata in which states are non-mergible by definition; see Grunsky et al. (2006). We may therefore adopt a general strategy of iteratively dividing states until the conditions for the desired type of description are met. Our tool of choice shall be the *observation table*, which in its most general interpretation fits the characteristics of universal automata more closely than those of any other kind of finite-state automaton. This also means that our way of obtaining an automaton from an observation table is distinctively different from any earlier approach such as Angluin (1987); Bollig et al. (2009).

Learning Models

When formalizing a learning task, the information source is of key importance. This can for instance be a finite set of positive examples, also known as a *text* (the limits of this source are discussed in Angluin (1987)), or a potentially infinite sequence of positive and negative examples, known as an *enumeration* (Gold, 1967). A substantial amount of work has also been devoted to algorithms that learn by querying an oracle. Angluin (1987) introduced the notion of a *minimal adequate teacher* (MAT) to allow for polynomial-time learning of regular languages. This is an oracle capable of answering two types of queries, membership and equivalence queries. Let L be the target language. An *equivalence query* (EQ) is of the form "Is $\mathcal{A}$ a correct description of L?", and is answered by the oracle either with a simple 'yes', or with a counterexample in the symmetric difference of $\mathcal{A}$ and L (that is, with an element in $c \in (L \setminus \mathcal{L}(\mathcal{A})) \cup (\mathcal{L}(\mathcal{A}) \setminus L)$). *Membership queries* (MQs), on the other hand, are of the type "Is w an element of L?" and are answered with 'yes' or 'no'. In the present article, we adopt the MAT model and require the learner to return the target universal automaton after a finite number of queries.

Learning Objects

While Angluin (1987) focused on learning regular languages by presenting state-minimal deterministic finite-state automata (DFA) as hypotheses to the teacher, our learner builds *universal automata* (UA), which constitute another kind of canonical description for regular languages. A survey of the theory of UA has been provided by Lombardy and Sakarovitch (2008). A third kind of FA besides DFA also providing a canonical description for regular languages, namely *residual finite-state automata* (RFSA; see Denis et al. (2001)), have already been considered in the MAT model by (Bollig et al., 2009). Interestingly, the hypotheses presented by the RFSA learner in Bollig et al. (2009) are not always ensured to be state-minimal RFSA but can seemingly be arbitrary non-deterministic automata (NFA); only the final, correct hypothesis is guaranteed to be the canonical RFSA of the target language.

Motivation and Results

We first consider two important facts on the descriptional complexity of UA in contrast to state-minimal DFA, comparing the number of states.

- There are regular languages whose state-minimal DFA is exponentially more succinct than the corresponding universal automaton.
- There are regular languages whose universal automaton is exponentially more succinct than the corresponding state-minimal DFA.

The first fact is explicitly shown in Lombardy and Sakarovitch (2008). The second fact is our first result in this paper, and also provides a solid motivation for studying these devices in the context of GI. As our main contribution, we will present a learning algorithm for universal automata that needs at most a cubic number of equivalence queries, measured in terms of the number of states of the universal automaton of the target language. This result contrasts those by Yokomori (1994); Denis et al. (2001); Bollig et al. (2009) which all refer to the state-minimal *deterministic* automaton when indicating the (polynomial) complexity of their respective learners for various kinds of special NFA. Our learnability result is based on the fact that a universal automaton consistent with the data seen so far can be easily computed from any observation table. In particular, the closedness and consistency conditions known from and necessary for other approaches can be relaxed here. In this mathematical sense, universal automata fit to observation tables better than any other canonical model of regular languages.

2. Preliminaries

Before we continue, it is useful to revise some of the notions and notations related to MAT learning, and to introduce a number of technical devices that will serve as our toolbox in the upcoming discussions.

2.1. Finite-State Automata

A *finite-state automaton (FA)* is a tuple $\mathcal{A} = \langle \Sigma, Q, I, F, \delta \rangle$ where

- Σ is a finite set of *alphabet symbols*,
- Q is the finite set of *states*,
- $I \subseteq Q$ is the set of *start* or *initial* states,
- $F \subseteq Q$ is the set of *accepting states*, and
- $\delta \subseteq Q \times \Sigma \times Q$ is the *transition relation*.

From the transition relation δ, we derive the functions $\delta^+ : Q \times \Sigma^* \longrightarrow 2^Q$ and $\delta^* : \Sigma^* \longrightarrow 2^Q$. Intuitively, $\delta^+(q, w)$ is the set of all states that can be reached from q on input $w \in \Sigma^*$, and $\delta^*(w)$ is the set of all states that can be reached from an initial state on w. More formally, δ^+ is given by $\delta^+(q, \varepsilon) = \{q\}$ and, for every $w = w'a \in \Sigma^+$,

$$\delta^+(q, w) = \{q'' \in Q \mid \exists q' \in \delta^+(q, w), q'' \in Q : \langle q', a, q'' \rangle \in \delta\} \ .$$

We can now define $\delta^* : \Sigma^* \longrightarrow 2^Q$ as

$$\delta^*(w) = \bigcup_{q \in I} \delta^+(q, w) \ .$$

With every state q, we shall associate two sets of strings, $\mathcal{P}_q$ and $\mathcal{F}_q$. Intuitively, $\mathcal{P}_q$ is the set of all strings that can end up in q (the *past* of q), and $\mathcal{F}_q$ is the set of all strings that can lead from q into an accepting state (the *future* of q). Again, more formally, for every state $q \in Q$, let $\mathcal{P}_q := \{s \in \Sigma^* \mid q \in \delta^*(s)\}$ and $\mathcal{F}_q := \{e \in \Sigma^* \mid \delta^+(q, e) \cap F \neq \emptyset\}$. A state q is *reachable* if $\mathcal{P}_q \neq \emptyset$ and *co-reachable* if $\mathcal{F}_q \neq \emptyset$. An automaton is *trim* if all of its states are reachable and co-reachable. By keeping only the states that are reachable and co-reachable we obtain the *trimmed version* of an automaton; this can be easily done in polynomial time and does not change the accepted language.

Note that unlike the classical definition of FA, the above definition of an automaton allows for multiple start states. This is motivated by the fact every state will be identified with a pair $\langle X, Y \rangle$ of strings, intuitively corresponding to the past and future of that state. A state $\langle X, Y \rangle$ will be classified as a start state whenever $\varepsilon \in X$, which can be true for more than one such pair.

It will sometimes be useful to identify the automaton $\mathcal{A}$ with the membership predicate for the language that it recognizes. Given $w \in \Sigma^*$, we thus write

- $\mathcal{A}(s) = 1$ if $\delta^*(s) \cap F \neq \emptyset$,

- $\mathcal{A}(s) = 0$ if $\delta^*(s) \cap Q \neq \emptyset$ but $\delta^*(s) \cap F = \emptyset$, and
- $\mathcal{A}(s) = *$ if $\delta^*(s) = \emptyset$.

The language accepted by $\mathcal{A}$ is $\mathcal{L}(\mathcal{A}) := \{s \in \Sigma^* \mid \mathcal{A}(s) = 1\}$. A string language is *regular* if it is accepted by an FA.

An FA $\mathcal{A}$ is *total* if, for every $a \in \Sigma$ and $q \in Q$, there is a transition $\langle q, a, q' \rangle \in \delta$. Furthermore, $\mathcal{A}$ is a *deterministic FA* (abbreviated DFA) if $\langle q, a, q' \rangle, \langle q, a, q'' \rangle \in \delta$ implies $q' = q''$, otherwise *non-deterministic* (an NFA). For DFA, we may abbreviate $\delta^*(s) = \{q\}$ to $\delta^*(s) = q$, and $\delta^+(s, e) = \{q\}$ to $\delta^+(s, e) = q$ without risk of confusion.

We also write $L(w) = 1$ if $w \in L$ for $w \in \Sigma^*$ and $L \subseteq \Sigma^*$, and $L(w) = 0$ if $w \notin L$.

2.2. Factors of a Language and Universal Automata

Let Σ be an alphabet and $L \subseteq \Sigma^*$ be a language. A pair $\langle X, Y \rangle$ with $X, Y \subseteq \Sigma^*$ is a *subfactor* of L if $XY \subseteq L$. A subfactor $\langle X, Y \rangle$ is a *factor* of L if it is *maximal* with respect to inclusion, in other words, if for every $X \subseteq X'$ and $Y \subseteq Y'$, $X'Y' \subseteq L$ implies $X' = X$ and $Y' = Y$. Henceforth, we denote by $\mathit{fac}(L)$ the set of all factors of L.

As shown by Lombardy and Sakarovitch (2008), a language L is regular if $\mathit{fac}(L)$ is finite. Moreover, we can view $Q = \mathit{fac}(L)$ as the state set of an FA $\mathcal{U}_L = \langle \Sigma, Q, I, F, \delta \rangle$ with

- $I = \{\langle X, Y \rangle \in \mathit{fac}(L) \mid \varepsilon \in X\}$,
- $F = \{\langle X, Y \rangle \in \mathit{fac}(L) \mid \varepsilon \in Y\}$,
- $\langle \langle X, Y \rangle, a, \langle X', Y' \rangle \rangle \in \delta$ if and only if $XaY' \subseteq L$.

This (unique!) automaton is the called the *universal automaton* of L. Note that for $\langle X, Y \rangle \in \mathit{fac}(L)$, the set X determines the set Y and vice versa via, for example, $Y = \bigcap_{x \in X} x^{-1}L$. The bijection has several interesting implications (Lombardy and Sakarovitch, 2008), e.g.:

$$\langle \langle X, Y \rangle, a, \langle X', Y' \rangle \rangle \in \delta \iff Xa \subseteq X' \iff aY' \subseteq Y \ .$$

We now turn to Theorem 1, which is our first main result and which provides ample mathematical justification for our interest in inference algorithms for universal automata:

Theorem 1 *There are regular languages whose universal automaton is exponentially more succinct than the corresponding state-minimal DFA.*

Proof We support Theorem 1 on the language family $L_k = \{a, b\}^*\{a\}\{a, b\}^k$. For every natural number k, the state-minimal DFA for L_k has $\Omega(2^k)$ states – see for instance Hopcroft et al. (2001, Sec. 2.3.6) (although the example seems to be first mentioned in Meyer and Fischer (1971), where they attribute it to Peterson).

The state-minimal NFA of L_k has the following states, where i is the initial state and q_k the final state:

$$\begin{aligned} i \text{ with } (\mathcal{P}_i, \mathcal{F}_i) &= (\{a, b\}^*, \{a, b\}^*\{a\}\{a, b\}^k) \\ q_j \text{ with } (\mathcal{P}_{q_j}, \mathcal{F}_{q_j}) &= (\{a, b\}^*\{a\}\{a, b\}^j,\ \{a, b\}^{k-j}) \text{ for } 0 \leq j \leq k \end{aligned}$$

Its transitions are $\langle i, c, i\rangle$, $\langle i, a, q_0\rangle$, and, for every $x \in \{a, b\}$ and $0 \le j < k$, $\langle q_j, x, q_{j+1}\rangle$. The universal automaton for L_k is similar to the above NFA for L_k, having the same state set but additional transitions $\langle q_j, a, q_0\rangle$ and $\langle q_j, x, i\rangle$ for every $x \in \{a, b\}$ and $0 \le j \le k$. The past and future languages of the states are now found by adding $(\emptyset, \{a, b\}^+\{a\}\{a, b\}^k)$ to every q_j, so

$$(\mathcal{P}_{q_j}, \mathcal{F}_{q_j}) \quad = \quad (\{a, b\}^*\{a\}\{a, b\}^j,\ \{a, b\}^{k-j} \cup \{a, b\}^+\{a\}\{a, b\}^k) \text{ for } 0 \le j \le k \enspace .$$

Since for every $0 \le j \le k$, we have $\{a, b\}^*\{a\}\{a, b\}^j\{a, b\}^+\{a\}\{a, b\}^k \subseteq L_k$, the new transitions do not alter the recognized language. ■

2.3. Observation tables

We will now introduce the central data structure of our learning algorithm. Let $L \subseteq \Sigma^*$ be the target language. A triple $T = \langle S, E, obs\rangle$ consisting of two finite sets $S, E \subseteq \Sigma^*$ and a function $obs : S \times E \longrightarrow \{0, 1\}$ is an *observation table* for L if

- S is *prefix-closed*,
- E is *suffix-closed*,
- $\varepsilon \in S$ and $\varepsilon \in E$, and
- obs is a total function with

$$obs(s, e) = \begin{cases} 1 & \text{if } se \in L \text{ is confirmed,} \\ 0 & \text{if } se \notin L \text{ is confirmed.} \end{cases}$$

3. Tables of subsets

The following thoughts, which only require some basic set theory, are fundamental for our approach. We could have stated them in more concrete terms, but this abstract approach is better to convey the basic ideas. Similar notions have been developed in Clark (2010d); Courcelle et al. (1991).

We consider a *universe* $U \times V$ and a *target* $T \subseteq U \times V$. By letting $\mathfrak{U} = 2^U$ and $\mathfrak{V} = 2^V$, we create a *frame* $\mathfrak{U} \times \mathfrak{V}$. An element $(X, Y) \in \mathfrak{U} \times \mathfrak{V}$ is a *subfactor* of T if $X \times Y \subseteq T$. A subfactor (X, Y) of T is a *factor* if, for every subfactor (X', Y') of T, $X' \supseteq X$ and $Y' \supseteq Y$ imply that $X = X'$ and $Y = Y'$.

A set $\mathfrak{C} \subseteq \mathfrak{U} \times \mathfrak{V}$ is a *cover with respect to* T if, for every $(x, y) \in T$, there is some $(X, Y) \in \mathfrak{C}$ with $x \in X$ and $y \in Y$. A cover $\mathfrak{C} \subseteq \mathfrak{U} \times \mathfrak{V}$ is a *subfactor cover* (or a *factor cover*) if each $(X, Y) \in \mathfrak{C}$ is a subfactor (or a factor, respectively).[1]

The reasoning behind these definitions is as follows: Consider an alphabet Σ, take $U = \Sigma^*$, $V = \Sigma^*$, and let L be the target language of some learning process. The language

1. Courcelle et al. (1991) would have termed a subfactor cover with respect to T a *rectangular decomposition of the relation* T for obvious geometric reasons; we did not use that terminology because "decomposition" hints at some non-overlapping set system, which would point to the wrong direction.

L then defines an infinite target table T_L given by $(u,v) \in T_L$ iff $uv \in L$. The condition $X \times Y \subseteq T_L$ is now clearly equivalent to $X \cdot Y \subseteq L$ (where $\cdot$ denotes concatenation, lifted to sets in the usual way). In the formal language terminology introduced in Section 2, $\langle X, Y\rangle$ is a (sub)factor of L iff (X, Y) is a (sub)factor of T_L, while a (sub)factor cover corresponds to a set of (sub)factors $\{\langle X_i, Y_i\rangle \mid i \in J\}$ of L with $\bigcup_{i\in J} X_i \cdot Y_i = L$.

By repeatedly appealing to the axiom of choice, the following assertion is easily seen:

Lemma 2 *Let $\mathfrak{C} \subseteq \mathfrak{U} \times \mathfrak{V}$ and let $T \subset T' \subseteq U \times V$ be two targets. If $\mathfrak{C}$ is a cover (subfactor cover) with respect to T then there is some cover (subfactor cover) $\mathfrak{C}' \subseteq \mathfrak{U} \times \mathfrak{V}$ extending $\mathfrak{C}$ in the sense that $\mathfrak{C} \subseteq \mathfrak{C}'$.*

It is tempting to claim the same for factor covers, but unfortunately it does not hold. This is witnessed by $U = \{1\}$, $V = \{a, b\}$, $T = \{(1, a)\}$, and $T' = T \cup \{(1, b)\}$. Here, $\{U \times \{a\}\}$ is a factor cover of T, but $U \times \{a\}$ is not a factor of T', so no cover of T' can both contain $U \times \{a\}$ and be a factor cover of T'. As we shall see, it is possible to obtain a result corresponding to Lemma 2 also for factor covers, but this requires additional notation.

To this end, let us fix a *sub-universe* $S \times E$ of $U \times V$ such that $S \subseteq U$ and $E \subseteq V$. This restriction induces a *sub-frame* $\mathfrak{S} \times \mathfrak{E}$, with $\mathfrak{S} \subseteq \mathfrak{U}$ and $\mathfrak{E} \subseteq \mathfrak{V}$. Again, let $T \subseteq U \times V$ be our target, and assume that $\mathfrak{C} \subseteq \mathfrak{U} \times \mathfrak{V}$ is a cover with respect to T. The *cover and target induced by* $S \times E$ is then

$$\mathfrak{C}|_{S\times E} = \{(X \cap S, Y \cap E) \mid (X, Y) \in \mathfrak{C}\},$$

and $T|_{S\times E} = T \cap (S \times E)$, respectively. The names are justified by the elementary Lemma 3.

Lemma 3 *Let $\mathfrak{C}$ be a cover with respect to T. Then $\mathfrak{C}|_{S\times E}$ is a cover with respect to $T|_{S\times E}$. Moreover, if $\mathfrak{C}$ is a subfactor cover then $\mathfrak{C}|_{S\times E}$ is a subfactor cover, as well.*

Proof Assume that $\mathfrak{C}|_{S\times E}$ is not a cover with respect to $T|_{S\times E}$. Then there is an element $(x, y) \in S \times E$ not covered by $\mathfrak{C}|_{S\times E}$. However, as $\mathfrak{C}$ is a cover, there is some $(X, Y) \in \mathfrak{C}$ with $(x, y) \in X \times Y$. Clearly, $(x, y) \in (X \cap S) \times (Y \cap E)$, i.e., we have found some element from $\mathfrak{C}|_{S\times E}$, namely $(X \cap S, Y \cap E)$, that covers (x, y), contradicting our assumption. The "moreover-part" is trivial. ■

Still, even if $\mathfrak{C}$ is a factor cover then this does not necessarily imply that $\mathfrak{C}|_{S\times E}$ is a factor cover as well. We again support our claim on an example.

Example 1 *Let $U \times V$ with $U = \{1, 2, 3, 4\}$ and $V = \{a, b, c\}$ be a universe and*

$$T = \{(1, a), (1, b), (1, c), (2, a), (2, b), (3, b), (4, a)\}$$

our target. Furthermore, let $S \times E$ be a sub-universe with $S = \{1, 2\}$ and $E = \{a, b\}$. Then, the target induced by $S \times E$ is $T|_{S\times E} = \{(1, a), (1, b), (2, a), (2, b)\}$. It is easy to see that the only factor cover with respect to $T|_{S\times E}$ is $\{S \times E\}$. If we look at the factor cover $\mathfrak{C} = \{\{1\} \times V, \{1, 2, 4\} \times \{a\}, \{1, 2, 3\} \times \{c\}\}$ of T then its restriction $\mathfrak{C}|_{S\times E} = \{\{1\} \times E, S \times \{a\}, S \times \{b\}\}$ is clearly not a factor cover of $T|_{S\times E}$. However, by Lemma 3, it is a subfactor cover of $T|_{S\times E}$.

For the reverse direction, where we enlarge rather than restrict the domain, it is possible to embed smaller factor covers into larger ones. We introduce a further notion that becomes important in this context. Let T be a target with the universe $U \times V$.

- For $X \subseteq U$, $(X, V[X])$ denotes the *right-maximal subfactor induced by* X, i.e., $V[X]$ is the largest subset of V such that $(X, V[X])$ is a subfactor of T, i.e., $X \times V[X] \subseteq T$.
- For $Y \subseteq V$, $(U[Y], Y)$ analogously denotes the *left-maximal subfactor induced by* Y.

Lemma 4

- *For* $X \subseteq U$*,* $(X, V[X])$ *is a subfactor with* $V[X] = \{v \in V \mid \forall x \in X : (x, v) \in T\}$*.*
- *For* $Y \subseteq V$*,* $(U[Y], Y)$ *is a subfactor with* $U[Y] = \{u \in U \mid \forall y \in Y : (u, y) \in T\}$*.*
- *For* $X \subseteq U$*,* $(U[V[X]], V[X])$ *is a factor, called* the factor induced by X.
- *For* $Y \subseteq V$*,* $(U[Y], V[U[Y]])$ *is a factor, called* the factor induced by Y.

Let $\mathfrak{C} \subseteq \mathfrak{U} \times \mathfrak{V}$ be a factor cover with respect to the target T.

Lemma 5 *If* $\mathfrak{C}$ *is a factor cover with respect to* $T|_{S \times E}$ *then there is a factor cover* $\mathfrak{C}'$ *with respect to* T *such that* $\mathfrak{C} \subseteq \mathfrak{C}'|_{S \times E}$*. This fact is testified by the embedding* $f : \mathfrak{C} \to \mathfrak{C}'$*,* $(X, Y) \mapsto (U[Y], V[U[Y]])$ *which satisfies* $X \subseteq U[Y]$ *and* $Y \subseteq V[U[Y]]$*.*

Proof For $(X, Y) \in \mathfrak{C}$, by definition $U[Y]$ is the maximal subset of U with $U[Y] \times Y \subseteq T$. Similarly, $V[U[Y]]$ is the maximal subset of V for which $U[Y] \times V[U[Y]] \subseteq T$. Since $\mathfrak{C}$ is a factor cover with respect to $T|_{S \times E}$, we conclude that $X \subseteq U[Y]$ and $Y \subseteq V[U[Y]]$. The existence of a factor cover $\mathfrak{C}'$ extending $f(\mathfrak{C})$ now follows along the lines of Lemma 2. ■

Remark 6 *Let us comment on the previous proof: It is worth noticing here that the definition of* f *using* $U[Y]$ *and* $V[U[Y]]$ *is not completely symmetric. We could have chosen to define* $f' : \mathfrak{C} \to \mathfrak{C}'$*,* $(X, Y) \mapsto (U[V[X]], V[X])$*; this would work out equally well, with minor adjustments to the upcoming proofs. However, the mapping* f' *would look different compared to* f *in concrete examples. For instance, let* $U = V = \{1, 2\}$ *and* $T = \{(1, 1), (1, 2), (2, 1)\}$ *with* $S = E = \{1\}$*. Then,* $\{\{(1, 1)\}\}$ *is a factor cover of* $T|_{S \times E} = \{(1, 1)\}$*. The maximal subset of* U *for which* $U[Y] \times Y \subseteq T$ *is* $U[Y] = \{1, 2\}$*. Then,* $V[U[Y]]$ *would equal* $\{1\}$*. Hence,* f *is defined by* $\{(1, 1)\} \mapsto \{(1, 1), (2, 1)\}$*. The factor cover* $\mathfrak{C}'$ *extending* $f(\mathfrak{C})$ *would be* $\{\{((1, 1), (2, 1)\}, \{(1, 1), (1, 2)\}\}$*. If we would have chosen to use* f'*, this would yield* $\{(1, 1)\} \mapsto \{(1, 1), (1, 2)\}$ *instead. Also note that it would be simply wrong to try a symmetric definition like: "Consider* $(X, Y) \mapsto (X_Y, Y_X)$*, where* X_Y *is the maximal subset of* U *for which* $X_Y \times Y \subseteq T$ *and* Y_X *is the maximal subset of* U *for which* $X \times Y_X \subseteq T$*"; in our example, we would have* $X_Y = U$ *and* $Y_X = V$*, which would give the set* $X_Y \times Y_X = U \times V$ *which is not even a subfactor.*

With Lemma 5 fresh in mind, let us return to our running example:

Example 2 *(cont'd) Starting from the factor cover* $\mathfrak{C}' = \{S \times E\}$ *of* $T|_{S\times E}$*, we can use the embedding* f *in the proof of Lemma 5, which has a fixed-point on* $S \times E$*, to find the cover* $\mathfrak{K} = \mathfrak{C} \uplus \mathfrak{C}' = \{\{1\} \times V, \{1,2,4\} \times \{a\}, \{1,2,3\} \times \{c\}, S \times E\}$ *of* T*. Note that both* $\mathfrak{K}$ *and* $\mathfrak{C}$ *are factor covers of* T*, even though one is a proper subset of the other. This shows that factor covers are not necessarily unique, and that they need not contain the same number of elements. Also, the claimed (rather trivial) inclusion* $\mathfrak{C}' \subseteq \mathfrak{K}|_{S\times E}$ *may be strict.*

If we increase T *slightly, setting* $T' = T \cup \{(2,c)\}$ *and using the same restricting set* $S \times E$ *we would then get* $\mathfrak{K} = \{\{1,2,4\} \times \{a\}, \{1,2,3\} \times \{c\}, S \times V\}$*. Moreover,* $f(S \times E) = S \times V$*.*

Lemma 7 will play an important role in the later analysis of our learning algorithm.

Lemma 7 *The embedding* $f : \mathfrak{C} \to \mathfrak{C}'$*,* $(X,Y) \mapsto (U[Y], V[U[Y]])$ *from Lemma 5 is injective and satisfies* $X = U[Y] \cap S$ *and* $Y = V[U[Y]] \cap E$*.*

Proof Assume the contrary, i.e., there are (X,Y) and (X',Y') such that $(U[Y], V[U[Y]]) = (U[Y'], V[U[Y']])$. In a sense, the mapping f is defined in two steps, first computing the first component from Y and then computing the second component from $U[Y]$. Let us treat the first of these steps. By Lemma 5, X is extended towards $U[Y]$ satisfying $U[Y] \times Y \subseteq T$. Observe that, since (X,Y) was a factor of $T|_{S\times E}$, $(U[Y] \setminus X) \cap S = \emptyset$ (†). Analogously, $(U[Y'] \setminus X') \cap S = \emptyset$. Clearly, $(U[Y], V[U[Y]]) = (U[Y'], V[U[Y']])$ implies that $U[Y] = U[Y']$ and hence that $U[Y] \cap S = U[Y'] \cap S$, which implies $X = X'$ due to (†). A similar argument applies for the second step, yielding $Y = Y'$. This proves the claim. ■

A related result is the following one.

Lemma 8 *If* (X_i, Y_i)*,* $i \in \{1, \ldots, r\}$*, are factors of a target* T *over a universe* $U \times V$ *then so are* $(\bigcap_{i=1}^r X_i, V[\bigcap_{i=1}^r X_i])$ *with* $\bigcup_{i=1}^r Y_i \subseteq V[\bigcap_{i=1}^r X_i]$ *and* $(U[\bigcap_{i=1}^r Y_i], \bigcap_{i=1}^r Y_i)$ *with* $\bigcup_{i=1}^r X_i \subseteq U[\bigcap_{i=1}^r Y_i]$*, provided that the intersections are not empty.*

Here, $Z[\cdot]$ is the induced-operator that we have introduced above. It is natural to think that $\bigcup_{i=1}^r Y_i = V[\bigcap_{i=1}^r X_i]$ but the following example proves that this is not always the case.

Example 3 *Let* $U = \{\varepsilon, a, aa, b\}$ *and* $V = \{\varepsilon, a, b, bb\}$ *be finite languages over* $\Sigma = \{a, b\}$*. Let*

$$T = \{(a,\varepsilon), (aa,\varepsilon), (\varepsilon,a), (a,a), (aa,a), (b,a), (b,b), (\varepsilon,bb), (aa,b)\}.$$

The factors of this target are: $F_1 = (\{\varepsilon, a, aa, b\}, \{a\})$*,* $F_2 = (\{a, aa\}, \{\varepsilon, a\})$*,* $F_3 = (\{aa\}, \{\varepsilon, a, bb\})$*,* $F_4 = (\{\varepsilon, aa\}, \{a, bb\})$*,* $F_5 = (\{b\}, \{a, b\})$*. Let* $F_i = (X_i, Y_i)$*. Note that, since the* F_i *are factors,* $U[Y_i] = X_i$ *and* $V[X_i] = Y_i$*. Consider more concretely* $F_2 = (X_2, Y_2)$ *and* $F_4 = (X_4, Y_4)$*. Then,* $X_2 \cap X_4 = \{aa\} = X_3$*. Hence,* $V[X_3] = Y_3 = Y_2 \cup Y_4$*. On the other hand,* $Y_2 \cap Y_4 = \{a\} = Y_1$*, while* $V[Y_1] = X_1 = \{\varepsilon, a, aa, b\}$ *is a proper superset of* $X_2 \cup X_4 = \{\varepsilon, a, aa\}$*.*

In the following sections, T will be alternatively interpreted as the (learning) target and as the observation table of a learning process. In the latter context, it is also interesting to note that the sets U and V of the previous example are prefix-closed and suffix-closed, respectively.

Proof (of Lemma 8) We will prove the assertion for the case $r = 2$; for $r > 2$, an easy induction argument shows the claim. By symmetry, it is sufficient to show that $(X_1 \cap X_2, V[X_1 \cap X_2])$ is a factor, provided that $X_1 \cap X_2 \neq \emptyset$.

- Due to Lemma 4, $(X_1 \cap X_2, V[X_1 \cap X_2])$ is a subfactor.
- Consider some arbitrary $y \in Y_1 \cup Y_2$, i.e., $y \in Y_1$ or $y \in Y_2$. Without loss of generality, assume that $y \in Y_1$. As (X_1, Y_1) is a subfactor, for each $x \in X_1$, $(x, y) \in T$. Hence, $y \in V[X_1 \cap X_2]$. So, $Y_1 \cup Y_2 \subseteq V[X_1 \cap X_2]$ follows.
- By definition of $V[X_1 \cap X_2]$ again, there is no $y \notin V[X_1 \cap X_2]$ satisfying $\forall x \in X_1 \cap X_2 : (x, y) \in T$.
- Assume that there is some $x \notin X_1 \cap X_2$ with $\forall y \in V[X_1 \cap X_2] : (x, y) \in T$. As $x \notin X_1 \cap X_2$, $x \notin X_1$ or $x \notin X_2$. Without loss of generality, consider the first case. We know that, for all $y \in V[X_1 \cap X_2]$, $(x, y) \in T$. As $Y_1 \subseteq V[X_1 \cap X_2]$ by the second item, for all $y \in Y_1$, $(x, y) \in T$. This implies $x \in X_1$, as (X_1, Y_1) is a factor, contradicting our assumption.

■

Remark 9 *Notice that the set of factors fac(T) of a target T has a natural partial order (or lattice) structure imposed on it by letting $(X; Y) \leq (X', Y')$ if $X = X \cap X'$. This is equivalent to the condition $Y' = Y \cap Y'$. The reader can verify this with Example 3, where we find $F_3 \leq F_2 \leq F_1$, $F_2 \leq F_4$ and $F_5 \leq F_1$.*

4. Properties of hypotheses

In this section, we establish the connection between observation tables and universal automata. We begin by defining the *factors of an observation table*, to be contrasted with the previously defined factors of a language.

Definition 10 (Factors (of a table)) *Let $T = \langle S, E, obs \rangle$ be an observation table. A* subfactor *of T is a pair $\langle X, Y \rangle$ with $X \subseteq S$ and $Y \subseteq E$ such that for all $s \in X$ and all $e \in Y$ we have $obs(s, e) = 1$. Analogously, a* factor *of T is a subfactor $\langle X, Y \rangle$ of T such that for every subfactor $\langle X', Y' \rangle$ of T with $X \subseteq X'$ and $X \subseteq X'$, we have $\langle X, Y \rangle = \langle X', Y' \rangle$. The set of all factors of T is denoted by fac(T).*

Note that in contrast to the classical representation of observation tables introduced above, there is a more general interpretation – clearly, any observation table corresponds to some subset $T \subseteq S \times E$, and the reader can easily verify that the according notions of (sub)factors as discussed in Section 3 coincide. For instance, the target T from Example 3 can be viewed as belonging to the following observation table:

	ε	a	b	bb
ε	0	1	0	1
a	1	1	0	0
aa	1	1	0	1
b	0	1	1	0

More generally, any $T \subseteq S \times E$ corresponds to an observation table, provided that S is a prefix-closed finite language over some alphabet Σ, E is some suffix-closed finite language over Σ, and that the table entries are consistent with concatenation, i.e., whenever $xy = x'y'$ for some words $x, x' \in S$ and $y, y' \in E$, then $(x, y) \in T$ if and only if $(x', y') \in T$.

To differentiate between the notion of factors based on Cartesian products discussed in Section 3 and the one based on the catenation product, we use parentheses $(,)$ in the first case and pointed brackets $\langle, \rangle$ in the second one.

Remark 11 *Observe that if $L \neq \emptyset$, then $\langle X, Y\rangle \in fac(L)$ implies $X \neq \emptyset$ and $Y \neq \emptyset$. An analogous statement is true for observation tables that contain a non-zero entry. As observation tables containing only zero entries would lead to the empty language as a hypothesis, and as this will be checked in the very first step in the learning algorithm that we will present, we can henceforth assume that $X \neq \emptyset$ and $Y \neq \emptyset$ for all $\langle X, Y\rangle \in fac(T)$.*

Definition 12 (Automaton associated to a table) *Let T be an observation table for a language L. The* associated automaton *derived from T is $\mathcal{A}_T = \langle \Sigma, \overline{Q_T}, \overline{I_T}, \overline{F_T}, \overline{\delta_T}\rangle$ with*

- $\overline{Q_T} = fac(T)$,
- $\overline{I_T} = \{\langle X, Y\rangle \in \overline{Q_T} \mid \varepsilon \in X\}$,
- $\overline{F_T} = \{\langle X, Y\rangle \in \overline{Q_T} \mid \varepsilon \in Y\}$, *and*
- *for every $a \in \Sigma$ and $\langle X, Y\rangle, \langle X', Y'\rangle \in \overline{Q_T}$, we have $\langle\langle X, Y\rangle, a, \langle X', Y'\rangle\rangle \in \overline{\delta_T}$ if and only if $X \cdot \{a\} \cdot Y' \subseteq L$.*

From $\mathcal{A}_T$, we obtain the associated hypothesis *$\mathcal{H}_T = \langle \Sigma, Q_T, I_T, F_T, \delta_T\rangle$ as the trimmed version of $\mathcal{A}_T$.*

In the following, $\mathcal{H}_T$ will be the hypothesis that our learner presents to the teacher, while the condition $X \cdot \{a\} \cdot Y' \subseteq L$ in the construction of $\mathcal{A}_T$ is checked via membership queries to the teacher. Since we allow such queries during the synthetization process, we are guaranteed to find $\mathcal{A}_T$ for *any* observation table T. This sets our algorithm apart from previous MAT learners which require additional properties in their observation tables such as consistency and closedness before synthesizing an automaton. We provide an example of an observation table T where $\mathcal{A}_T \neq \mathcal{H}_T$ in Appendix 7.3.

Remark 13 *For $a \in \Sigma$ we define $T \circ a := \langle S, E, obs_a\rangle$ such that*

$$obs_a(s, e) = \begin{cases} 1 & \textit{if } sae \in L \textit{ is confirmed,} \\ 0 & \textit{if } sae \notin L \textit{ is confirmed.} \end{cases}$$

The definition of $\mathcal{A}_T$ forces us to compute these auxiliary tables, and an efficient implementation should save the information thus gathered so as to economize with membership queries. This motivation is purely from a complexity point of view because whether we save or re-compute these entries has no bearing on the correctness of the learning algorithm.

Definition 14 *$\mathcal{A}_T$ is* strongly reachable *if, for all $a \in \Sigma$ and $\langle X', Y' \rangle \in \overline{Q_T}$ and all $xa \in X'$, there is some $\langle X, Y \rangle \in \overline{Q_T}$ such that $x \in X$ and $\langle \langle X, Y \rangle, a, \langle X', Y' \rangle \rangle \in \overline{\delta_T}$. Analogously, we can define strong co-reachability.*

Lemma 15 *If $\mathcal{A}_T$ is strongly reachable then it is trim, i.e., $\mathcal{A}_T = \mathcal{H}_T$.*

Proof We only prove that every state of $\mathcal{A}_T$ is reachable, as co-reachability can be seen by a similar argument. The proof is by induction on the length of the shortest words $w(q) \in X$ of state $q = \langle X, Y \rangle$. If $|w(q)| = 0$, then $\varepsilon \in X$, i.e., q is an initial state and hence reachable. Assume that the claim is true for all q with $|w(q)| \leq n$. Consider some state $q = \langle X, Y \rangle$ with $|w(q)| = n + 1$. Hence, $w(q) = xa$. As $\mathcal{A}_T$ is strongly reachable, there is some state $p = \langle W, Z \rangle$ with $x \in W$ and $\langle p, a, q \rangle \in \overline{\delta_T}$. Hence, q is reachable. ■

We will need an even slightly stronger notion in what follows.

Definition 16 *An observation table $T = \langle S, E, obs \rangle$ for a language $L \subseteq \Sigma^*$ is* stable *if*

1. *for every $s, s' \in S$ such that there is $ae \in \Sigma E$ with $L(sae) > L(s'ae)$, there is $e' \in E$ such that $L(se') > L(s'e')$; and*
2. *for every $e, e' \in E$ such that there is $sa \in S\Sigma$ with $L(sae) > L(sae')$, there is $s' \in S$ such that $L(s'e) > L(s'e')$.*

Note that this is similar to Angluin's consistency condition in her LSTAR algorithm.

Lemma 17 *If T is stable then $\mathcal{A}_T$ is strongly reachable.*

Proof To show that $\mathcal{A}_T$ is strongly reachable, we proceed as follows. Let $\langle X', Y' \rangle$ be a factor of T such that there is a string $xa \in X'$. Let $\langle X, Y \rangle$ be a factor of T such that $x \in X$ and X is of minimal size. Since S is prefix-closed, this factor exists.

If $(\langle X, Y \rangle, a, \langle X', Y' \rangle) \in \delta_T$ then we are done. Otherwise, there is $z \in X$ and $y' \in Y'$ such that $L(zay') = 0$. Since $L(xay') = 1$, there is $y'' \in E$ such that $L(xy'') > L(zy'')$, so by adding y'' to Y we reduce the size of X while keeping $x \in X$, thus contradicting the minimality assumption. ■

We propose the procedure `MakeStable`(T): Look for $s, s' \in S$ such that there is $ae \in \Sigma E$ with $L(sae) > L(s'ae)$ but there is no $e' \in E$ such that $L(se') > L(s'e')$. Add ae to E and fill up the table with MQs. Symmetrically, strings can be added to S.

Lemma 18 *Every time we add an element from $S \cdot \Sigma$ to S, or from $\Sigma \cdot E$ to E in order to make T stable, the number of factors in T increases.*

Proof *Sketch.* Suppose we add e' to E due to Condition 1 in Definition 16. Every factor $\langle X, Y\rangle$ of T with $s \in X$ also had $s' \in X$ since no element in Y can prevent it from being so. By adding e' to E, $\langle X, Y\rangle$ splits into $\langle X, Y\rangle$ and $\langle X', S[X']\rangle$ with $X' := \{s \in X \mid se' \in L\}$. A similar argument holds when we enlarge S instead. ■

As we will make sure that any hypothesis automaton our learner conjectures is from a stable observation table T, we assume stability for all tables in the remainder of this section. This also implies that $fac(T)$ is the state set of any automaton $\mathcal{A}_T$ we consider.

The next lemma looks obvious but needs a not completely trivial induction argument.

Lemma 19 *Let* $\langle X, Y\rangle \in Q_T$. *Then* $X = \mathcal{P}_{\langle X,Y\rangle} \cap S$ *and* $Y = \mathcal{F}_{\langle X,Y\rangle} \cap E$.

Proof We only prove $X = \mathcal{P}_{\langle X,Y\rangle} \cap S$ since the part for the future set of $\langle X, Y\rangle$ follows from a symmetrical argument.

Let $\langle X, Y\rangle \in Q_T$. We have to prove the following two assertions:

1. If $w \in X$ then $w \in \mathcal{P}_{\langle X,Y\rangle} \cap S$, and
2. If $w \in \mathcal{P}_{\langle X,Y\rangle} \cap S$ then $w \in X$.

The proof is by induction on the length of w. As the induction base, consider $w = \varepsilon$.

1. Since $\varepsilon \in X$ implies $\langle X, Y\rangle \in I_T$ we have $\varepsilon \in \mathcal{P}_{\langle X,Y\rangle}$ by definition of the set $\mathcal{P}_{\langle X,Y\rangle}$.
2. If $\varepsilon \in \mathcal{P}_{\langle X,Y\rangle} \cap S$ then $\langle X, Y\rangle$ must be an initial state of $\mathcal{A}_T$, as our automata do not have transitions on the empty word. By definition, this means that $\varepsilon \in X$.

Now, assume the claim to hold for all states and for all words w of length up to n. Consider some w with $|w| = n + 1$. Hence, $w = ua \in S$ for some $u \in \Sigma^n$ and $a \in \Sigma$. The remainder of the proof differs for the respective parts:

1. Consider $w = ua \in X$. As $\mathcal{A}_T$ is strongly reachable (Lemma 17), there is a table factor $\langle X', Y'\rangle$ with $u \in X'$ and $\langle\langle X', Y'\rangle, a, \langle X, Y\rangle\rangle \in \delta_T$. By the induction hypothesis, $u \in \mathcal{P}_{\langle X',Y'\rangle} \cap S$ since S is prefix-closed. As $\langle\langle X', Y'\rangle, a, \langle X, Y\rangle\rangle \in \delta_T$, $w \in \mathcal{P}_{\langle X,Y\rangle} \cap S$.
2. Assume $w \in \mathcal{P}_{\langle X,Y\rangle} \cap S$. Let $\langle X', Y'\rangle$ be a state that can be passed when leading $w = ua$ into $\langle X, Y\rangle$, with $\langle\langle X', Y'\rangle, a, \langle X, Y\rangle\rangle \in \delta_T$. By the choice of $\langle X', Y'\rangle$, $u \in \mathcal{P}_{\langle X',Y'\rangle} \cap S$ since S is prefix-closed. By the induction hypothesis, $u \in X'$. By the definition of δ_T, in particular for all $y \in Y$, we find that $obs_a(u, y) = 1$. Since $w = ua$, $obs(w, y) = 1$ for all $y \in Y$. As $\langle X, Y\rangle$ is a table factor, we conclude that $w \in X$.

■

We now turn our attention to a notion of consistency well-known in Learning Theory but less frequently addressed explicitly in Grammatical Inference:

Definition 20 $\mathcal{A}$ *is* T-consistent *if* $\mathcal{A}(se) = obs(s, e)$ *for every* $\langle s, e\rangle \in S \times E$.

Lemma 21 *The automaton $\mathcal{A}_T$ is T-consistent.*

Proof Let $\langle s, e\rangle \in S \times E$ with $obs(s, e) = 1$. There is a factor $\langle X, Y\rangle = \langle S[\{e\}], E[S[\{e\}]]\rangle$ such that $s \in X$ and $e \in Y$. Assuming that T is stable, by Lemma 19, $X = \mathcal{P}_{\langle X,Y\rangle} \cap S$ and $Y = \mathcal{F}_{\langle X,Y\rangle} \cap E$, so $\langle X, Y\rangle \in \delta_T^*(s)$ and $\delta_T^+(\langle X, Y\rangle, s) \subseteq F_T$, and consequently $\mathcal{A}_T(se) = 1$.

For the opposite direction, assume that there is an accepting run of $\mathcal{A}_T$ on se. After having read all of s, $\mathcal{A}_T$ must be in some state $\langle X, Y\rangle$ from which it can continue to an accepting state. We thus know that $s \in \mathcal{P}_{\langle X,Y\rangle} \cap S$ and that $e \in \mathcal{F}_{\langle X,Y\rangle} \cap E$. By Lemma 19, $s \in X$ and $e \in Y$ but since $\langle X, Y\rangle$ is a factor, this yields $obs(s, e) = 1$. ■

So, in the following we can assume that $\mathcal{A}_T$ is T-consistent. This will be an important property when we prove the correctness of our inference algorithm. Moreover, we can establish the following lemma.

Lemma 22 *If the states $\langle X, Y\rangle, \langle X_1, Y_1\rangle, \ldots, \langle X_r, Y_r\rangle \in Q_T$ are such that the language $\mathcal{F}_{\langle X,Y\rangle}$ fulfils $\mathcal{F}_{\langle X,Y\rangle} \subseteq \bigcup_{i=1}^r \mathcal{F}_{\langle X_i,Y_i\rangle}$ then we have $Y \subseteq \bigcup_{i=1}^r Y_i$.*

Proof If the claim were wrong then there would be some $w \in Y \subseteq E$ not contained in any of the Y_i. However, since Lemma 19 tells us that $Y \subseteq \mathcal{F}_{\langle X,Y\rangle}$, there must be some $\mathcal{F}_{\langle X_i,Y_i\rangle}$ containing w. Pick some arbitrary $v \in X_i$. As $vw \in X_i\mathcal{F}_{\langle X_i,Y_i\rangle}$, we have $vw \in \mathcal{L}(\mathcal{A}_T)$ due to Lemma 19 which proves that $X_i \subseteq \mathcal{P}_{\langle X_i,Y_i\rangle}$. By T-consistency, $obs(v, w) = 1$. As v was arbitrary, this shows that $obs(v, w) = 1$ for all $v \in X_i$. Hence, $\langle X_i, Y_i'\rangle \in fac(T)$ for some $Y_i \cup \{w\} \subseteq Y_i'$, contradicting our assumption of $\langle X_i, Y_i\rangle \in Q_T$. ■

Another important property of observation tables is *closedness*. In our framework, this corresponds to the notion of *saturation*. Although this property is not automatically satisfied, we will argue that this does not matter for our learning algorithm.

Definition 23 *An observation table T for the language L is* saturated *if for every pair of table factors $\langle X, Y\rangle, \langle X', Y'\rangle$ with $XaY' \subseteq L$, there is some $x \in X$ such that $xa \in X'$ and there is some $y \in Y'$ such that $ay \in Y$.*

Lemma 24 *Let T be a saturated observation table. For every natural number r and choice of r table factors $\langle X_i, Y_i\rangle$, $i \in \{1, \ldots, r\}$, it holds that*

$$\bigcap_{i=1}^{r} \mathcal{P}_{\langle X_i,Y_i\rangle} \neq \emptyset \text{ if and only if } \bigcap_{i=1}^{r} X_i \neq \emptyset \ .$$

Proof The "if" direction is immediate from Lemma 22.

Therefore, let x be a string of minimal length that is witness to the falsity of the opposite direction of the lemma (so the lemma holds for every proper prefix of x). We note that x cannot be the empty string because a state $\langle X, Y\rangle$ is in $\delta_T^*(x)$ if and only if it is an initial state, which it is if and only if $\varepsilon \in X$.

Therefore, let $x = ua$ for some string $u \in \Sigma^*$ and symbol $a \in \Sigma$, and let $\{\langle X_i, Y_i\rangle \mid i \in \{1, \ldots, r\}\} = \delta_T^*(x)$. Moreover, let $\langle W_i, Z_i\rangle$ with $i \in \{1, \ldots, r\}$ be a selection of factors in $\delta_T^*(u)$ such that $\{\langle\langle W_i, Z_i\rangle, a, \langle X_i, Y_i\rangle\rangle \mid i \in \{1, \ldots, r\}\} \subseteq \delta_T$.

By Lemma 8 and the minimality of x, the factor $\langle W, Z\rangle = \langle \bigcap_{i=1}^{r} W_i, E[\bigcap_{i=1}^{r} W_i]\rangle$ exists and since W is a subset of every W_i the set $\{\langle\langle W, Z\rangle, a, \langle X_i, Y_i\rangle\rangle \mid i \in \{1, \ldots, r\}\}$ is contained in δ_T. Property 23 now lets us pick an arbitrary $w \in W$ such that $wa \in X_j$ for some $j \in \{1, \ldots, r\}$, and because of $WaY_i \subseteq L$ for every Y_i, the maximality of the factors and the containment of $wa \in X_j \subseteq S$, we have that $wa \in X_i$ for every $i \in \{1, \ldots, r\}$. This obviously contradicts our assumption concerning x. ■

We propose the following procedure: `MakeSaturated`(T) looks for two factors $p = \langle X, Y\rangle$ and $q = \langle X', Y'\rangle$ such that $XaY' \subseteq L$ and $xa \notin X'$ for all $x \in X$. Pick the shortest $t \in X$.

1. If $|t| \leq |Q_T|$ then add ta to S.
2. If $|t| > |Q_T|$ then consider a decomposition $t = uvw$ with $|uw| \leq |Q_T|$. Add uwa and all prefixes of uw to S.

Then update the observation table by membership queries. Analogously, the case of $y \in Y'$ but $ay \notin Y$ is treated, thereby extending the set E. For an example, see Appendix 7.4.

Note that the second case remains by pigeon hole, similar to the standard proof of the pumping lemma for regular languages.

Lemma 25 *Let $T = \langle S, E, obs\rangle$ be an observation table for L and let $T' = \langle S', E', obs'\rangle$ be the observation table resulting from* `MakeSaturated`(T). *Provided that the procedure always terminates, T' is a saturated observation table for L.*

Proof Consider two factors $p = \langle X, Y\rangle$ and $q = \langle X', Y'\rangle$ of T' with $XaY' \subseteq L$. Our procedure `MakeSaturated`(T) guarantees that among the shortest strings in X, there is some x with $xa \in X'$. A symmetric argument applies for the enlargements of E. ■

We will ensure termination when presenting our learner but will assume it for now. We are now going to state the main result of this section.

Theorem 26 *$\mathcal{A}_T$ is the universal automaton for $\mathcal{L}(\mathcal{A}_T)$.*

We prove this by two lemmata. The first, Lemma 27, shows that the states of $\mathcal{A}_T$ satisfy the defining property of universal automata. This is sufficient for the claim, as the second, Lemma 28, shows that the states correspond to factors of the language recognized by $\mathcal{A}_T$.

Lemma 27 *Let $\langle X, Y\rangle, \langle X', Y'\rangle \in Q_T$ and $a \in \Sigma$. We have $\langle\langle X, Y\rangle, a, \langle X', Y'\rangle \in \delta_T$ if and only if $\mathcal{P}_{\langle X,Y\rangle} \cdot \{a\} \cdot \mathcal{F}_{\langle X',Y'\rangle} \subseteq \mathcal{L}(\mathcal{A}_T)$.*

Proof The "only if" direction follows from the definition of δ_T, and of the past and future languages of a state.

The "if" direction is shown as follows. Due to Lemma 19, $X \subseteq \mathcal{P}_{\langle X,Y\rangle}$ and $Y' \subseteq \mathcal{F}_{\langle X',Y'\rangle}$. Hence, $X \cdot \{a\} \cdot Y' \subseteq \mathcal{L}(\mathcal{A}_T)$. Due to the T-consistency of $\mathcal{A}_T$, for all $s \in X$ and all $e \in Y'$ we have $obs_a(s, e) = 1$, meaning that $sae \in L$. Hence, $\langle\langle X, Y\rangle, a, \langle X', Y'\rangle\rangle \in \delta_T$. ■

Lemma 28 *For all $q \in Q_T$, the pair $\langle \mathcal{P}_q, \mathcal{F}_q \rangle$ is a factor of $\mathcal{L}(\mathcal{A}_T)$.*

Proof To prove the claim, we can assume $T = \langle S, E, obs \rangle$ to be saturated by Lemma 25. In the following, let $q = \langle X, Y \rangle$. Clearly, $\langle \mathcal{P}_q, \mathcal{F}_q \rangle$ is a subfactor of $\mathcal{L}(\mathcal{A}_T)$. To violate maximality, there is some $s \notin \mathcal{P}_q$ with $\{s\} \cdot \mathcal{F}_q \subseteq \mathcal{L}(\mathcal{A}_T)$ or some $e \notin \mathcal{F}_q$ with $\mathcal{P}_q\{e\} \subseteq \mathcal{L}(\mathcal{A}_T)$. By symmetry, it is sufficient to discuss the first of these cases. So, we will prove by induction on the length of s that whenever we have $\{s\} \cdot \mathcal{F}_q \subseteq \mathcal{L}(\mathcal{A}_T)$ we also have $s \in \mathcal{P}_q$.

Assume then that $\{s\} \cdot \mathcal{F}_q \subseteq \mathcal{L}(\mathcal{A}_T)$, from which we obtain $\{s\} \cdot Y \subseteq \mathcal{L}(\mathcal{A}_T)$ by applying Lemma 19. If $s = \varepsilon$ then $\mathcal{F}_q \subseteq \mathcal{L}$, and hence q is an initial state and ε is trivially in $\mathcal{P}_q$. This proves the base case of the induction.

For the inductive step, assume the claim to be true for all s of length up to n and consider some $s = ua$ of length $n + 1$. Let $\delta^*_T(s) = \{\langle X_i, Y_i \rangle \mid 1 \leq i \leq r\}$ for some $r \in \mathbb{N}$.

As $\{s\} \cdot \mathcal{F}_q \subseteq \mathcal{L}(\mathcal{A}_T)$, we have $\delta_T(s) \neq \emptyset$ and moreover $\mathcal{F}_q \subseteq \bigcup_{i=1}^{r} \mathcal{F}_{\langle X_i, Y_i \rangle}$. By Lemma 22 this yields

$$Y \subseteq \bigcup_{i=1}^{r} Y_i \ .$$

Let us consider certain factors of T in sequence:

- For every $i \in \{1, \ldots, r\}$, let $\langle Z_i, W_i \rangle \in \delta^*_T(u)$ and $\langle \langle Z_i, W_i \rangle, a, \langle X_i, Y_i \rangle \rangle \in \delta_T$.
- For every $i \in \{1, \ldots, r\}$, let $X'_i = S[Y_i \cap Y] \supseteq X'_i \cup X$ and $Y'_i = Y_i \cap Y$. Since Y_i and Y overlap, this factor exists, and as $Y_i \cap Y \subseteq Y_i$, we have $\langle \langle Z_i, W_i \rangle, a, \langle X'_i, Y'_i \rangle \rangle \in \delta_T$.
- Let $\langle Z, W \rangle$ fulfil $Z = \bigcap_{i=1}^{r} Z_i$ and $W = E[\bigcap_{i=1}^{r} Z_i] \subseteq \bigcup_{i=1}^{r} W_i$. Since $u \in \bigcap_{i=1}^{r} \mathcal{P}_{\langle Z_i, W_i \rangle}$, the intersection of $Z = \bigcap_{i=1}^{r} Z_i$ is not empty, as T is saturated and due to Lemma 24. Moreover, these are factors by Lemma 8. Furthermore, $\langle \langle Z, W \rangle, a, \langle X, Y \rangle \rangle \in \delta_T$ as we have $zay \in L$ for every $z \in Z$ and $y \in Y$.

Now, $\{u\} \cdot \mathcal{F}_{\langle Z, W \rangle} \subseteq L(\mathcal{A}_T)$, and thus $u \in \mathcal{P}_{\langle Z, W \rangle}$ by the induction hypothesis. In combination with $\langle \langle Z, W \rangle, a, \langle X, Y \rangle \rangle \in \delta_T$, we obtain $s \in \mathcal{P}_{\langle X, Y \rangle}$. ∎

This concludes the proof of Theorem 26. This theorem constitutes the backbone of our learning algorithm, which we are going to present in the following section.

5. Inference algorithm

We propose the following learning algorithm for the inference of universal automata within the MAT model. We assume that the target alphabet Σ is given to the learner in advance.

Initialization Our learner starts out with an initial table $T_0 = \langle S_0, E_0, obs_0 \rangle$, defined by $S_0 = E_0 = \{\varepsilon\}$. A single membership query (MQ) thus suffices to complete the table. The learner synthesizes the automaton $\mathcal{A}_{T_0}$ from T_0 and asks an equivalence query (EQ) with argument $\mathcal{A}_{T_0}$.

Loop If the teacher accepts $\mathcal{A}_{T_i}$ as the universal automaton for the target language L then the algorithm terminates successfully. Otherwise, the learner recieves a counter-example w_i. On the receipt of w_i, the learner adds all prefixes of w_i to S_i and all suffixes to E_i, yielding the sets S_{i+1} and E_{i+1}. The table T_{i+1} is then completed using MQs, and made stable and saturated using `MakeStable` and `MakeSaturated`, and $\mathcal{A}_{T_{i+1}}$ becomes the next conjecture submitted to the teacher.

We give two example runs of our learner in Appendix 7.1.

This algorithm satisfies a number of properties which we state in a sequence of lemmata. First of all, recall that due to Lemma 21 the learner's hypothesis automaton $\mathcal{A}_T$ is always T-consistent. Furthermore, we have:

Lemma 29 *Either $\mathcal{L}(\mathcal{A}_{T_0}) = \emptyset$ or there is some subset $A \subseteq \Sigma$ such that $\mathcal{L}(\mathcal{A}_{T_0}) = A^*$.*

Proof When constructing T_0, the algorithm checks if $\varepsilon \in L$ via an MQ. If $\varepsilon \notin L$ then the automaton $\mathcal{A}_{T_0}$ will have an empty state set Q_0 and no transitions. If $\varepsilon \in L$ then $\mathcal{A}_{T_0}$ will have a singleton state set Q_0. In that case we also have $I_0 = F_0 = Q_0$. Upon building the transitions of $\mathcal{A}_{T_0}$, the algorithm first checks if any $a \in \Sigma$ is in the target language L (via MQs) and adds loop transitions to the only state accordingly. If $L \cap \Sigma = \emptyset$, no loop transitions are added, which is reflected by the case $A = \emptyset$ in the formulation of the claim. ■

The languages mentioned in Lemma 29 are exactly those that can be accepted by any universal automaton with at most one state, which shows that our algorithm would need only one equivalence query for the corresponding target automata.

Lemma 30 *For each $i \geq 0$, if $\mathcal{A}_{T_{i+1}}$ is presented as a hypothesis, then there is an injective embedding $f_i : Q_i \to Q_{i+1}$ with the property that whenever $\langle X, Y \rangle \mapsto \langle X', Y' \rangle$ then $X = X' \cap S_i$ and $Y = Y' \cap E_i$. A similar statement is true for the intermediate automata obtained before calling `MakeStable` or `MakeSaturated`.*

Proof We use notations introduced in Sec. 3. First, observe that $T_i = T_{i+1}|_{S_i \times E_i}$. Clearly, $Q_i = \mathit{fac}(T_i)$ is a factor cover of T_i. Hence, Lemmas 5 and 7 provide an injective embedding into some factor cover of T_{i+1} which is clearly contained in $Q_{i+1} = \mathit{fac}(T_{i+1})$. The claimed properties $X = X' \cap S_i$ and $Y = Y' \cap E_i$ translate from Lemma 7. ■

Lemma 31 *For each $i \geq 0$, if the automaton $\mathcal{A}_{T_{i+1}}$ is presented as a hypothesis and if the embedding $f_i : Q_i \to Q_{i+1}$ is bijective then $f_i^{-1} : Q_{i+1} \to Q_i$ is an automaton morphism. Moreover, the induced mapping $d_i : \delta_{T_{i+1}} \to \delta_{T_i}$ is injective.*

Proof To avoid a special case, observe that since our algorithm always makes progress in the sense of changing its hypothesis between two rounds, no set of states and no set of transitions considered in this lemma can be empty, as the only possibility to obtain the empty language or the language $\{\varepsilon\}$ as a hypothesis would be with $\mathcal{A}_{T_0}$; we refer to Lemma 29. It remains to show that, whenever there is an a-transition from q to p in $\mathcal{A}_{T_{i+1}}$ then there is an a-transition between the corresponding states $f_i^{-1}(q)$ and $f_i^{-1}(p)$. More concretely, we know that $q, p \in \mathit{fac}(T_{i+1})$, i.e., $q = \langle X_q, Y_q \rangle$ and $p = \langle X_p, Y_p \rangle$. Moreover, Lemma 7 explains that $f_i^{-1}(q) = q' = \langle X'_q, Y'_q \rangle$ with $X'_q = X_q \cap S_i$ and $f_i^{-1}(p) = p' = \langle X'_p, Y'_p \rangle$ with

$X'_p = X_p \cap E_i$. By definition, $\langle q, a, p \rangle \in \delta_{T_{i+1}}$ if $xay \in L$ for all $x \in X_q$ and all $y \in Y_p$. Hence, we have $xay \in L$ for all $x \in X'_q$ and $y \in Y'_p$, so that $\langle q, a, p \rangle \in \delta_{T_{i+1}}$ implies $\langle q', a, p' \rangle \in \delta_{T_i}$ as claimed. Clearly, $d_i : \langle q, a, p \rangle \mapsto \langle q', a, p' \rangle \in \delta_{T_i}$ is injective since f_i is a bijection. ■

Let $\mathcal{U}_L$ be the universal automaton for the target language L with state set $Q = fac(L)$. The following assertion can be seen by the same arguments as in the proof of Lemma 30.

Lemma 32 *For each $i \geq 0$, if $\mathcal{A}_{T_{i+1}}$ is presented as a hypothesis then there exists an injective embedding $f_i : Q_i \to Q$ with the property that whenever $\langle X, Y \rangle \mapsto \langle X', Y' \rangle$ then $X = X' \cap S_i$ and $Y = Y' \cap E_i$.*

Theorem 33 *Our proposed learning algorithm converges to the target automaton $\mathcal{U}_L$ after at most $\max\{1, |\Sigma| n^3\}$ many equivalence queries, where n is the number of states of $\mathcal{U}_L$.*

Proof Due to Lemma 32, any hypothesis automaton has at most as many states as $\mathcal{U}_L$. Moreover, Lemma 30 shows that $n_i \leq n_{i+1}$, where $n_j = |Q_j|$ is the number of states of the jth hypothesis. This together with Theorem 26 and the fact that universal automata are unique up to renaming of states shows that the learning algorithm will finally yield the target automaton $\mathcal{U}_L$. In the following reasoning, let m_j denote the number of transitions of the jth hypothesis. Clearly, $m_j \leq |\Sigma| n_j^2$. Since we always have $\mathcal{A}_{T_j} \neq \mathcal{A}_{T_{j+1}}$ due to the received counterexamples, we can observe two types of progress:

1. Either, $n_j < n_{j+1}$.
   Since we always have $n_{j+1} \leq n$, this type of progress can happen at most n times.

2. Or, $n_j = n_{j+1}$ but $m_j > m_{j+1}$.
   This case is due to Lemma 31. Since we constantly have $m_j \leq |\Sigma| n_j^2 \leq |\Sigma| n^2$, this type of progress can happen at most $|\Sigma| n^2$ times.

These two observations together show the claimed upper bound on convergence speed. ■

This implies that, for instance, the language L_k defined in the proof of Theorem 1 can be derived in $O(k^3)$ EQs, while a classical LSTAR learner as introduced by Angluin (1987) would need $\Omega(2^k)$ many EQs. This shows that learning universal automata could be (at least occasionally) much faster than learning DFA.

Remark 34 *An argument similar to the proof of Theorem 33, based on Lemma 30, shows that the procedures `MakeStable` and `MakeSaturated` terminate, as they will always either increase the number of states or add only a small number of table entries a fixed number of times. More precisely, if we have a smart teacher that never provides us with unnessarily long counterexamples, then the length of any counterexample is bounded by the number n of states of the target automaton. So, each update necessary after an equivalence query would add at most n rows and n columns to the table. Due to Lemma 18, `MakeStable` allows a similar estimate. `MakeSaturated` might add in the worst case n rows and n columns per case (given by two factors $p = \langle X, Y \rangle$ and $q = \langle X', Y' \rangle$ such that $XaY' \subseteq L$ and $xa \notin X'$ for all $x \in X$), and there could be at most a quadratic number of cases, as we would repair* *each case by selecting a smallest $t \in X$ and adding ta to X'. So, at most a cubic number of rows and columns are added here. This dominates the worst case, so that we can conclude that a cubic number of times, at most a cubic number of rows and columns are added to the table. This shows that there are at most $O(n^6)$ rows and columns in the observation table at the termination of the learning algorithm, so that the table contains at most $O(n^{12})$ many entries, which also upper-bounds the total number of membership queries ever made by the algorithm. Of course, if the teacher is not so smart, bigger tables might occur.*

6. Discussions, Conclusions and Future Research

We have presented a novel MAT learner for regular languages. In this section, we are going to explain some connections and differences to alternative MAT learners and also point to directions of future research.

6.1. Comparison With Other MAT Learners

To discuss similarities and differences to the famous LSTAR learner of Angluin (1987), we introduce some more notations. Let $T = \langle S, E, obs \rangle$ be an observation table. For $s \in S$, let $row[s] = \{e \in E \mid obs(s,e) = 1\}$. Slightly abusing the notation introduced in Sec. 3, $\langle \{s\}, row[s] \rangle$ is the factor induced by $\{s\}$, as $row[s] = E[\{s\}]$ is the right-maximal subfactor of $\{s\}$. For $row[s] \neq \emptyset$, let $\langle S[row[s]], row[s] \rangle$ be the *row factor* of s. Likewise, let $col[e] = \{s \in S \mid obs(s,e) = 1\}$ and $\langle col[e], E[col[e]] \rangle$ be the *column factor* of e. Let $\mathfrak{R}$ and $\mathfrak{C}$ collect all row and column factors, respectively.

Remark 35 *If $\langle X, Y \rangle$ is any factor, then $(X,Y) \in \mathfrak{R} \cup \mathfrak{C}$. This also gives an algorithm for computing $fac(T)$: As long as there exists some yet uncovered (s,e) with $obs(s,e) = 1$, compute the row and column factor of s resp. e and add them to the cover.*

Remark 36 *If LSTAR constructs a hypothesis from T, then the hypothesized automaton has as state set $\mathfrak{R}$. By way of contrast, our algorithm's hypothesis automaton $\mathcal{A}_T$ has state set $fac(T) = \mathfrak{R} \cup \mathfrak{C}$.*

Another difference lies in the definition of a transition function for LSTAR. Define $row[s]_a = \{e \in E \mid obs_a(s,e) = 1\}$ for $a \in \Sigma$. We obtain a transition $\langle s, a, s' \rangle$ for $s, s' \in S$ and $a \in \Sigma$ if $row[s]_a = row[s']$. If for all $a \in \Sigma$ and all $s \in S$ there is some $s' \in S$ with $row[s]_a = row[s']$ then we call T *closed* and all states in the resulting automaton are reachable. If for all $a \in \Sigma$ and all $s, s' \in S$ with $row[s] = row[s']$ we have $row[s]_a = row[s']_a$ then we call T *consistent* and the resulting automaton is deterministic. Note that this way of constructing an automaton from an observation table surely differs from our method.

We compare a run of LSTAR and of our learner in the appendix, assuming that the teacher gives the same counterexample to both learners whenever possible. We observe that

- our learner needs one more EQ. This could be corrected by letting it start with a table containing all strings of length 1 both in S and in E but then we would also have to ask more MQs in order to fill in the additional cells.

- in this small example, apart from our learner's additional EQ at the beginning the sequence of hypotheses of both learners can be made to coincide if we require LSTAR to eliminate a failure state before submitting its hypothesis to the teacher.
  As a proposal for future work we ask: Is this true for any run of these two learners in those cases where the universal automaton and the non-total state-minimal DFA for the target language are identical (assuming that the teacher reacts with the same counterexamples to the same hypotheses)?

Another MAT learner which we could compare to ours is the learner for residual finite-state automata (RFSA) in Bollig et al. (2009). The canonical RFSA for a language L can also be exponentially more succinct than the state-minimal DFA, and is as most as big. In this case, the derivation of an automaton from a table is still based on the concept of rows but the notion of identity between rows (equality between sets) is replaced by a covering relation (subset relation). In case of successful learning, only equivalence classes of L that are strict supersets of the union of all other classes they contain are admitted as states. So, in a sense, this type of automata fit less well into our framework, discussing factors of tables etc. However, it might be an idea to translate the mentioned covering relation to our more general setting as a line of future research.

6.2. Upper Bounds on The Number of Equivalence Queries

We underline that while Yokomori (1994); Denis et al. (2001); Bollig et al. (2009) all refer to the state-minimal *deterministic* automaton when indicating the (polynomial) complexity of their respective learners for various kinds of special NFA, we give an algorithm with polynomial runtime in terms of the non-deterministic *target* automaton, which can be exponentially more succinct. We conjecture that there is a close connection to the notion of *polynomial characterizability* by de la Higuera (1997) – this property is fulfilled by DFA but not by NFA in general (modulo the complexity-theoretic assumption that $P \neq NP$) de la Higuera (1997), nor by residual finite-state automata Bollig et al. (2009) (also see Kasprzik (2012) for more discussion). We surmise that for universal automata it is again fulfilled, which would yield a further explanation for our advantageous result.

6.3. Comments on Distributional Learning

Also note that our way of deriving an automaton from an observation table differs from those in Angluin (1987) or Bollig et al. (2009) for residual finite-state automata (RSFA) inasmuch as we do not base it on rows alone (orginally formulated as sequences of 0s and 1s induced by the labeling sets) but on concrete subsets of the labeling sets, along with the respective consequences. However, this approach is quite close to the notion of *distributional learning* developed by Clark (2010b,c) for context-free grammars (a MAT learner for CFGs can be found in Clark (2010d)).

Let us finally remark that also the framework developed by Clark in a series of papers, e.g., Clark (2010a,b, 2011) fits into the framework developed in Sec. 3. In its basic setting, Clark associates to each language L the set of subwords $Sub(L) = \{u \in \Sigma^* \mid \exists l, r \in \Sigma^* : lur \in L\}$ and the set of contexts $\mathcal{C}(L) = \{(l, r) \in \Sigma^* \times \Sigma^* \mid \exists u \in \Sigma^* : lur \in L\}$. Let $U = Sub(\Sigma^*)$ and $V = \mathcal{C}(\Sigma^*)$. Then, L again yields a target $T = \{(u, (l, r)) \mid lur \in L\}$.

The lattice structure mentioned in Remark 9 is central to Clark's approach. To generalize the learning strategies that we develop for universal automata towards such targets would be a challenging question for future research.

6.4. Generalizing the Setting: Alternative Learning Scenarios and Objects

As indicated in the last sections of Lombardy and Sakarovitch (2008), we may find that a generalization of our approach towards the learning of subsets of monoids, not only free monoids, is possible. We are only aware of text learning results for algebraic structures, see Stephan and Ventsov (2001). We also encourage to further our approach to learning other structures such as trees, matrices of symbols, or tuples of strings.

Alternatively, we can look into other learning models, taking universal automata as our target descriptions. For instance, see García et al. (2008) for an alternative universal automata learner from positive and negative examples relying on the state merging paradigm.

The presentation of LSTAR in terms of our approach resembles Courcelle et al. (1991). In particular, LSTAR factor covers are described by the row set $\mathfrak{R}$ and hence satisfy:

> Each $(s, e) \in S \times E$ with $se \in L$ is covered by exactly one $(X, E[X]) \in \mathfrak{C}(Q)$, where s uniquely determines X (equivalence class decomposition of S).

This coincides with what Courcelle et al. (1991, Def. 1.4) call a *deterministic decomposition*. They observe that there are always canonical (minimal) such decompositions and they also show that these naturally correspond to state-minimal DFAs, i.e., the hypothesis space of the LSTAR algorithm, which provides an alternative explanation of some of the results of Angluin (1987). These connections are interesting, all the more so as Courcelle et al. (1991) also provide applications of their approach to certain kinds of top-down tree automata and to regular ω-languages. From a formal-language point of view, this raises the question if objects like universal automata exist in those contexts as well. From the viewpoint of Grammatical Inference, developing MAT learning algorithms for such universal automata would then be the challenge.

References

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

B. Bollig, P. Habermehl, C. Kern, and M Leucker. Angluin-style learning of NFA. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAR*, pages 1004–1009, 2009.

A. Clark. Towards general algorithms for grammatical inference. In *Algorithmic Learning Theory, ALT*, volume 6331 of *LNAI*, pages 11–30. Springer, 2010a.

A. Clark. Learning context-free grammars with the syntactic concept lattice. In *International Colloquium on Grammatical Inference, ICGI*, volume 6339 of *LNCS*, pages 38–51. Springer, 2010b.

A. Clark. Efficient, correct, unsupervised learning of context-sensitive languages. In *Fourteenth Conference on Computational Natural Language Learning, CoNLL*, pages 28–37, 2010c.

A. Clark. Distributional learning of some context-free languages with a minimally adequate teacher. In *International Colloquium on Grammatical Inference, ICGI*, volume 6339 of *LNCS*, pages 24–37. Springer, 2010d.

A. Clark. A learnable representation for syntax using residuated lattices. In P. de Groote, M. Egg, and L. Kallmeyer, editors, *Formal Grammar – 14th International Conference,FG 2009*, volume 5591 of *LNCS*, pages 183–198. Springer, 2011.

B. Courcelle, D. Niwinski, and A. Podelski. A geometrical view of the determinization and minimization of finite-state automata. *Math. Systems Theory*, 24(2):117–146, 1991.

C. de la Higuera. Characteristic sets for polynomial grammatical inference. *Machine Learning*, 27:125–138, 1997.

F. Denis, A. Lemay, and A. Terlutte. Learning regular languages using RFSA. In *Algorithmic Learning Theory, ALT*, volume 2225 of *LNCS*, pages 348–363. Springer, 2001.

P. García, M. Vázquez de Parga, G. I. Álvarez, and J. Ruiz. Universal automata and NFA learning. *Theoretical Computer Science*, 407(1-3):192–202, 2008.

E. M. Gold. Language identification in the limit. *Inf. and Control*, 10(5):447–474, 1967.

I. Grunsky, O. Kurganskyy, and I. Potapov. On a maximal NFA without mergible states. In D. Grigoriev, J. Harrison, and E. A. Hirsch, editors, *Computer Science — Theory and Applications, First International Computer Science Symposium in Russia, CSR*, volume 3967 of *LNCS*, pages 202–210. Springer, 2006.

J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation.* Reading (MA): Addison-Wesley, 2nd edition, 2001.

A. Kasprzik. *Formal Tree Languages and Their Algorithmic Learnability.* PhD thesis, Fachbereich IV, Universität Trier, Germany, 2012.

S. Lombardy and J. Sakarovitch. The universal automaton. In E. Grädel, J. Flum, and W. Thomas, editors, *Logic and Automata: History and Perspectives*, pages 457–504. Amsterdam University Press, 2008.

A. Meyer and M. Fischer. Economies of description by automata, grammars, and formal systems. In *Proceedings of the 12th Annual Symposium on Switching and Automata Theory, SWAT*, pages 188–191, 1971.

F. Stephan and Y. Ventsov. Learning algebraic structures from text. *Theoretical Computer Science*, 268(2):221–273, 2001.

T. Yokomori. Learning non-deterministic finite automata from queries and counterexamples. In *Machine Intelligence 13*, pages 169–189, 1994.

7. Appendix

7.1. Two example runs

We give two example runs of our learner. The target language for the first is an instance of L_k as introduced in the proof of Theorem 1 which implies that the target automaton is exponentially more succinct than the corresponding state-minimal DFA.

Example 4 *The target is* $L_4 = \{a,b\}^*\{a\}\{a,b\}^4$ *("the last symbol but 4 is an 'a' "). The learner starts out with the observation table*

	ε
ε	0

and derives from it the automaton $\langle \Sigma, \emptyset, \emptyset, \emptyset, \emptyset \rangle$.
The teacher reacts by giving the positive counterexample aaaaa. This leads to the table

	ε	a	a^2	a^3	a^4	a^5
ε	0	0	0	0	0	1
a	0	0	0	0	1	1
a^2	0	0	0	1	1	1
a^3	0	0	1	1	1	1
a^4	0	1	1	1	1	1
a^5	1	1	1	1	1	1

and to a corresponding state set

$$\begin{aligned}
q_0 &= \langle\{\varepsilon, a, a^2, a^3, a^4, a^5\}, \{a^5\}\rangle \\
q_1 &= \langle\{a, a^2, a^3, a^4, a^5\}, \{a^4, a^5\}\rangle \\
q_2 &= \langle\{a^2, a^3, a^4, a^5\}, \{a^3, a^4, a^5\}\rangle \\
q_3 &= \langle\{a^3, a^4, a^5\}, \{a^2, a^3, a^4, a^5\}\rangle \\
q_4 &= \langle\{a^4, a^5\}, \{a, a^2, a^3, a^4, a^5\}\rangle \\
q_5 &= \langle\{a^5\}, \{\varepsilon, a, a^2, a^3, a^4, a^5\}\rangle
\end{aligned}$$

where q_0 is the only start state and q_5 is the only accepting state. The transitions of the resulting automaton can be seen in Figure 1.
The teacher reacts by giving the positive counterexample abbbb. This leads to the table

	ε	a	a^2	a^3	a^4	a^5	b	b^2	b^3	b^4	ab^4
ε	0	0	0	0	0	1	0	0	0	0	1
a	0	0	0	0	1	1	0	0	0	1	1
a^2	0	0	0	1	1	1	0	0	1	1	1
a^3	0	0	1	1	1	1	0	1	1	1	1
a^4	0	1	1	1	1	1	1	1	1	1	1
a^5	1	1	1	1	1	1	1	1	1	1	1
ab	0	0	0	1	0	1	0	0	1	0	1
ab^2	0	0	1	0	0	1	0	1	0	0	1
ab^3	0	1	0	0	0	1	1	0	0	0	1
ab^4	1	0	0	0	0	1	0	0	0	0	1

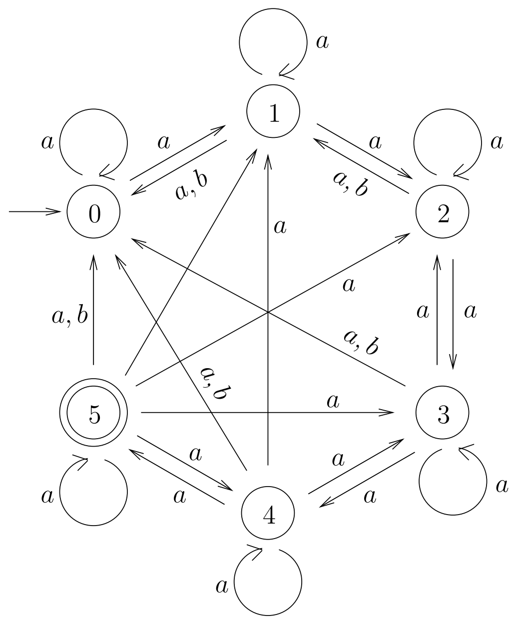

Figure 1: Example 4, second hypothesis

and to a corresponding state set

$$\begin{array}{lcl} p_0 & = & \langle\{\varepsilon, a, a^2, a^3, a^4, a^5, ab, ab^2, ab^3, ab^4\}, \{a^5, ab^4\}\rangle \\ p_1 & = & \langle\{a, a^2, a^3, a^4, a^5\}, \{a^4, a^5, b^4, ab^4\}\rangle \\ p_2 & = & \langle\{a^2, a^3, a^4, a^5\}, \{a^3, a^5, b^3, ab^4\}\rangle \\ p_3 & = & \langle\{a^3, a^4, a^5\}, \{a^2, a^5, b^2, ab^4\}\rangle \\ p_4 & = & \langle\{a^4, a^5\}, \{a, a^5, b, ab^4\}\rangle \\ p_5 & = & \langle\{a^5\}, \{\varepsilon, a^5, ab^4\}\rangle \end{array}$$

where p_0 is the only start state and p_5 is the only accepting state. Note that the number of states has not changed with respect to the previous hypothesis. The transitions of the resulting automaton can be seen in Figure 2. The past $\mathcal{L}_{p_i}$ and the future $\mathcal{C}_{p_i}$ corresponding to the states in $\{p_0, \ldots, p_5\}$ are:

$$\begin{array}{ll|l} p_0: & \{a,b\}^* & \{a,b\}^*\{a\}\{a,b\}^4 = L_4 \\ p_1: & \{a,b\}^*\{a\} & \{a,b\}^+\{a\}\{a,b\}^4 \cup \{a,b\}^4 \\ p_2: & \{a,b\}^*\{a\}\{a,b\} & \{a,b\}^+\{a\}\{a,b\}^4 \cup \{a,b\}^3 \\ p_3: & \{a,b\}^*\{a\}\{a,b\}^2 & \{a,b\}^+\{a\}\{a,b\}^4 \cup \{a,b\}^2 \\ p_4: & \{a,b\}^*\{a\}\{a,b\}^3 & \{a,b\}^+\{a\}\{a,b\}^4 \cup \{a,b\} \\ p_5: & \{a,b\}^*\{a\}\{a,b\}^4 = L_4 & \{a,b\}L_4 \cup \{\varepsilon\} \end{array}$$

These are exactly the factors of L_4 and the hypothesis is the universal automaton for L_4.

Example 5 *Let the target language be $\overline{L_3}$, i.e, the complement of $L_3 = \{a,b\}^*\{a\}\{a,b\}^3$ ("the last symbol but 3 must* not *be an 'a' "). The learner starts out with the table*

	ε
ε	1

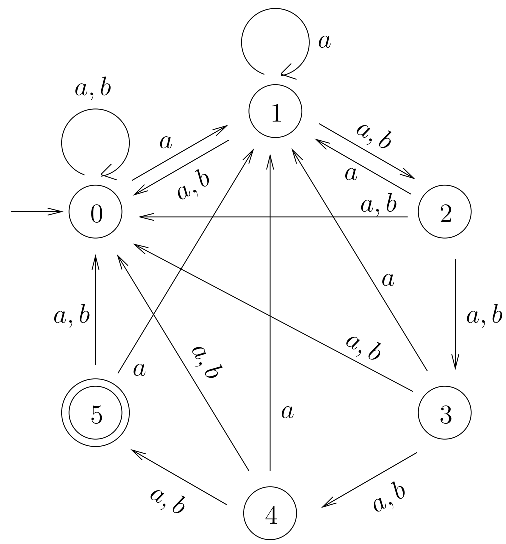

Figure 2: Example 4, final hypothesis

and derives from it the total all-accepting automaton with respect to Σ.
The teacher reacts by giving the negative counterexample aaaa. This leads to the table

	ε	a	a^2	a^3	a^4
ε	1	1	1	1	0
a	1	1	1	0	0
a^2	1	1	0	0	0
a^3	1	0	0	0	0
a^4	0	0	0	0	0

and to a corresponding state set

$$\begin{array}{lcl} q_0 & = & \langle\{\varepsilon\}, \{\varepsilon, a, a^2, a^3\}\rangle \\ q_1 & = & \langle\{\varepsilon, a\}, \{\varepsilon, a, a^2\}\rangle \\ q_2 & = & \langle\{\varepsilon, a, a^2\}, \{\varepsilon, a\}\rangle \\ q_3 & = & \langle\{\varepsilon, a, a^2, a^3\}, \{\varepsilon\}\rangle \end{array}$$

where all states are accepting and start states. The transitions of the resulting automaton can be seen in Figure 3.

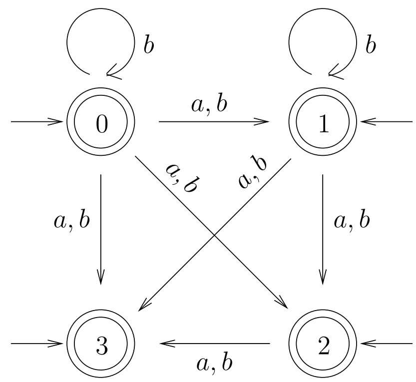

Figure 3: Example 5, second hypothesis

The teacher reacts by giving the positive counterexample abaaa. This leads to the table

	ε	a	a^2	a^3	a^4	ba^3	aba^3
ε	1	1	1	1	0	1	1
a	1	1	1	0	0	1	1
a^2	1	1	0	0	0	1	1
a^3	1	0	0	0	0	1	1
a^4	0	0	0	0	0	1	1
ab	1	1	0	1	0	1	1
aba	1	0	1	0	0	1	1
aba^2	0	1	0	0	0	1	1
aba^3	1	0	0	0	0	1	1

and to a corresponding state set

$$\begin{aligned}
p_0 &= \langle\{\varepsilon\}, \{a, a^2, a^3, ba^3, aba^3\}\rangle \\
p_1 &= \langle\{\varepsilon, a, a^2, a^3, ab, aba^3\}, \{\varepsilon, ba^3, aba^3\}\rangle \\
p_2 &= \langle S, \{ba^3, aba^3\}\rangle \\
p_3 &= \langle\{\varepsilon, a\}, \{\varepsilon, a, a^2, ba^3, aba^3\}\rangle \\
p_4 &= \langle\{\varepsilon, a, a^2, ab\}, \{\varepsilon, a, ba^3, aba^3\}\rangle \\
p_5 &= \langle\{\varepsilon, a, a^2, ab, aba^2\}, \{a, ba^3, aba^3\}\rangle \\
p_6 &= \langle\{\varepsilon, a, aba\}, \{a^2, ba^3, aba^3\}\rangle \\
p_7 &= \langle\{\varepsilon, ab\}, \{a^3, ba^3, aba^3\}\rangle \\
p_8 &= \langle\{\varepsilon, a, a^2, ab, aba^2\}, \{ba^3, aba^3\}\rangle
\end{aligned}$$

where all states are start states and p_0, p_1, p_3, p_4 *are accepting.*
The transitions of the resulting automaton can be seen in Figure 4 where all non-labeled edges are 'a, b'-edges. The hypothesis is the universal automaton for $\overline{L_3}$.

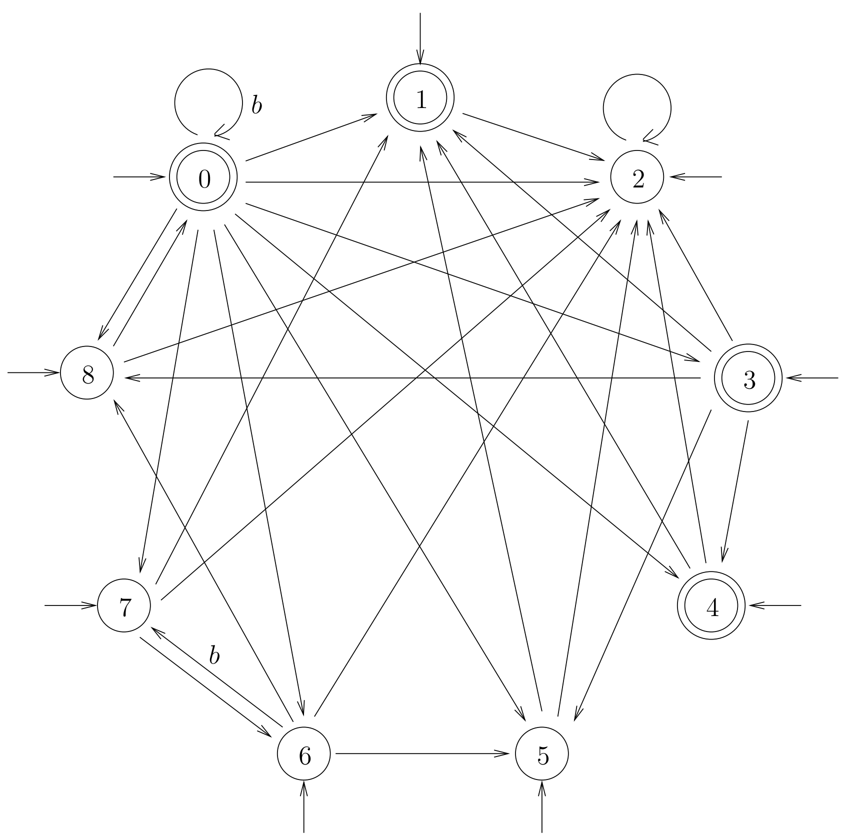

Figure 4: Example 5, final hypothesis (non-labeled edges are ‘a, b’-edges)

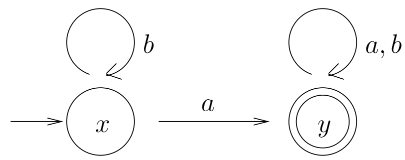

Figure 5: First (non-empty) hypothesis of LSTAR and of our learner

7.2. A run of our learner and a run of LSTAR

We assume familiarity with Angluin's learner LSTAR for DFA (Angluin, 1987). Let the target language be $L = \{a\}\{a,b\}^*$. LSTAR starts out with a table

	ε
ε	0
a	1
b	0

and then closes it to obtain

	ε
ε	0
a	1
b	0
aa	1
ab	1

which yields its first hypothesis as shown in Figure 5 with states $x = \langle 0 \rangle$ and $y = \langle 1 \rangle$.

——————— *(switch)* ———————

Meanwhile, our learner starts out with a table

	ε
ε	0

and derives from it the automaton $\langle \Sigma, \emptyset, \emptyset, \emptyset, \emptyset \rangle$. Let us say that the teacher reacts with the (shortest possible) counterexample a. The learner builds a table

	ε	a
ε	0	1
a	1	1

and derives from it the automaton in Figure 5 with states $x = \langle S, \{a\} \rangle$ and $y = \langle \{a\}, E \rangle$. Obviously, the hypotheses of both learners at this stage are now isomorphic.

Let the next given counterexample be ba.

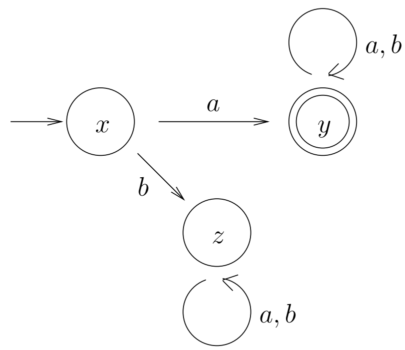

Figure 6: Second hypothesis of LSTAR

——————— *(switch)* ———————

LSTAR reacts to the counterexample ba by building a closed table

	ε	a	ba
ε	0	1	0
a	1	1	1
b	0	0	0
aa	1	1	1
ab	1	1	1
ba	0	0	0
bb	0	0	0

and obtains the FA shown in Figure 6 with states $x = \langle 0, 1, 0\rangle$, $y = \langle 1, 1, 1\rangle$, and $z = \langle 0, 0, 0\rangle$, which is the state-minimal total DFA for L.

——————— *(switch)* ———————

Our learner reacts to the counterexample ba by building the table

	ε	a	b	ba
ε	0	1	0	0
a	1	1	1	1
b	0	0	0	0
ba	0	0	0	0

and obtains the automaton shown in Figure 7 with states $x = \langle\{\varepsilon, a\}, \{a\}\rangle$ and $y = \langle\{a\}, E\rangle$, which is the universal automaton for L and also the non-total state-minimal DFA for L.

We observe that

- our learner needs one more EQ. This could be corrected by letting it start with a table containing all strings of length 1 both in S and in E but then we would also have to ask more MQs in order to fill in the additional cells.

- in this small example, apart from our learner's little "hickup" at the beginning the sequence of hypotheses of both learners can be made to coincide if we require LSTAR

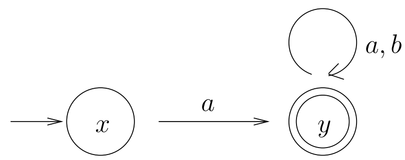

Figure 7: Second (non-empty) hypothesis of our learner

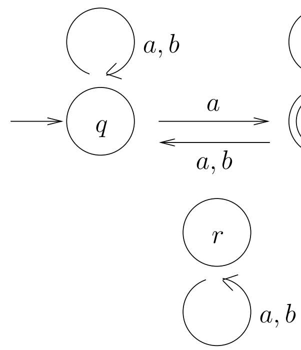

Figure 8: The (non-trim) automaton $\mathcal{A}_T$

to eliminate a failure state before submitting its hypothesis to the teacher. Is this true for any run of these two learners in cases where the universal automaton and the non-total state-minimal DFA for the target language are identical (assuming that the teacher reacts with the same counterexamples to the same hypotheses)?

7.3. An example for $\mathcal{A}_T \neq \mathcal{H}_T$

The language L contains all strings except those that have a 0 in the following table T_L.

	ε	a	b	bb
ε	0	1	0	1
a	1	1	0	0
aa	1	1	0	1
b	0	1	1	0

Let us consider a table T for L as follows:

	ε	a	b
ε	0	1	0
a	1	1	0
b	0	1	1

We have factors $q = \langle\{\varepsilon, a, b\}, \{a\}\rangle$ and $p = \langle\{a\}, \{\varepsilon, a\}\rangle$ and $r = \langle\{b\}, \{a, b\}\rangle$. The automaton $\mathcal{A}_T$ is given in Figure 8. We find that the factor r is not reachable. As a consequence, $\mathcal{A}_T$ would not accept bb and is thus not T-consistent. However, we remark that in a run of our learner no intermediate stage would yield such a table. The learner starts out with T_0:

	ε
ε	0

then gets for example the (shortest) counterexample a and builds T_1:

	ε	a
ε	0	1
a	1	1

The automaton $\mathcal{A}_{T_1}$ is exactly the trimmed version of $\mathcal{A}_T$ we have derived from the "problematic table" T. Mind that $\mathcal{A}_{T_1}$ already contains b-transitions as the learner checks them via MQs. However, bb is not yet in the table and can be given as a counterexample.

7.4. A table and its saturated version

We consider the language L defined in Appendix 7.3 right above. The table T_L is not saturated: For the pair $\langle X,Y\rangle, \langle X',Y'\rangle \in \mathit{fac}(T_L)$ with $\langle X,Y\rangle = \langle X',Y'\rangle = \langle\{\varepsilon,a,aa,b\},\{a\}\rangle$, which fulfils $X\{a\}Y' \subseteq L$, there is no element $aa \in Y = \{a\}$. Saturating the table yields

	ε	a	b	bb
ε	0	1	0	1
a	1	1	0	0
aa	1	1	0	1
b	0	1	1	0
ba	1	1	1	1

and possibly some more rows or columns. However, from this small cutout alone we can already see that by saturation we get at least one more factor due to the new last row.