Learning Residual Finite-State Automata Using Observation Tables

Anna Kasprzik
kasprzik@informatik.uni-trier.de

September 16, 2009

Keywords: Algorithmic learning theory, grammatical inference, observation tables, residual languages, non-determinism, regular languages

Abstract

In the area of grammatical inference the problem of how to infer a description of a formal language (e.g., a grammar or an automaton) algorithmically from given examples or other kinds of information sources is considered. One of the best studied classes with respect to this conception of learnability is the class of regular languages.

A significant part of the learning algorithms for regular languages of which Angluin's seminal algorithm (Angluin 1982) was one of the first use the concept of an *observation table*. A table fulfilling certain conditions represents a minimal deterministic finite-state automaton (DFA), and if the given information entered into the table is sufficient the learning algorithm succeeds and the derived automaton is the minimal DFA for the language in question. There is a one-to-one correspondence between the states of the minimal DFA and the equivalence classes of the language under the Myhill-Nerode congruence relation.

Since the minimal DFA can have exponentially more states than a minimal non-deterministic automaton (NFA) for the same language and since for many applications a small number of states is a desirable feature it seems worth considering if we cannot infer a NFA instead. Denis et al. (2001) introduce a special case of NFA, so-called residual finite-state automata (RFSAs), where each state represents a residual language of the language recognized by the automaton. There is a unique minimal RFSA for every regular language.

We define a two-step learning algorithm for RFSAs based on the concept of observation tables by first using an existing algorithm for minimal DFAs to build a table for the reversal of the language in question and then showing that we can derive the minimal RFSA for the language itself from this table after some simple modifications. We compare this algorithm to two other table-based algorithms of which one is an Angluin-style algorithm by Bollig et al. (2009) which infers a RFSA directly, and the other is another two-step algorithm proposed by the author. The comparison

concentrates on the criterion of query complexity. We find as a result that in theory the direct algorithm of Bollig et al. does not outperform the combination of any of the known near-optimal algorithms inferring the minimal DFA with the modifications we present, although it behaves much better in practice.

1 Introduction

In the area of grammatical inference the problem of how to infer – or "learn" – a description of a formal language (e.g., a grammar or an automaton) algorithmically from given examples or other kinds of information sources is considered. A range of conceivable learning settings have been formulated and based on those quite an amount of learning algorithms have been developed. One of the best studied classes with respect to this conception of learnability is the class of *regular languages* in the Chomsky Hierarchy.

A significant part of the algorithms that have been developed, of which Angluin's seminal algorithm L^* for regular string languages [1] was one of the first, use the concept of an *observation table*. In an observation table the rows are labeled by elements from some set that can be taken as prefixes, the columns are labeled by elements from some set that can be taken as suffixes, and each cell of the table contains a Boolean value indicating the membership status of the concatenation of the two labeling elements with respect to the language L the learner is trying to infer (provided that this status is known from the available information). If the table fulfils certain conditions then we can immediately derive a deterministic finite-state automaton (DFA) from it, and if the information entered into the table is sufficient then this automaton is isomorphic to the minimal DFA $\mathcal{A}_L$ for L. There is a one-to-one correspondence between the states of $\mathcal{A}_L$ and the equivalence classes of L under the syntactic congruence relation on which the Myhill-Nerode theorem (see for example [4]) is based. The elements labeling the rows of the table from which $\mathcal{A}_L$ is derived can be seen as representatives for these equivalence classes, and the ones labeling the columns as experiments by which it is possible to prove that two of those representatives do indeed belong to different equivalence classes and should thus represent different states of $\mathcal{A}_L$.

Unfortunately there is a price to pay for the uniqueness of the minimal DFA: In the worst case it can have exponentially more states than a minimal nondeterministic finite-state automaton (NFA) for the same language, and since for many applications a small number of states is a desirable feature it seems worthwile to consider the question if we cannot find a way to obtain such a nondeterministic automaton instead. In [5], Denis et al. introduce a special case of NFAs, so-called *residual finite-state automata (RFSAs)*, where each state represents a residual language of the language that is recognized (for the definition of a residual language see Section 2). These have the advantageous property that there is a unique minimal RFSA $\mathcal{R}_L$ for every regular language L. Denis et al. [6, 7, 8] have also presented several learning algorithms for RFSAs, which,

however, all work by adding or deleting states in an automaton according to the information contained in a given sample.

We demonstrate in Section 3 that we can easily define a two-step learning algorithm for RFSAs based on the concept of an observation table by first using an (almost arbitrary) existing algorithm for minimal DFAs to build a table with certain properties for the reversal (see Section 2) of the language L in question and then showing that it is possible to derive the minimal RFSA $\mathcal{R}_L$ for L itself from this table after some simple and cheap necessary modifications. In Subsection 3.2 we compare this algorithm to two other observation table-based algorithms of which one is an incremental Angluin-style algorithm by Bollig et al. [9] which infers a RFSA directly, and the other is another two-step algorithm proposed by the author of this paper. The comparison mainly focuses on the criterion of query complexity. We find as a result that at least in theory the direct algorithm of Bollig et al. [9] does not outperform the combination of any of the known near-optimal algorithms inferring the minimal DFA with the modifications presented in Subsection 3.1 (although Bollig et al. [9] can show statistically that their algorithm behaves much better in practice).

We study two-step algorithms in the hope that their modularity will contribute to an even clearer view of what is going on in algorithms that retrieve the syntactic equivalence classes of a regular language using an observation table. Furthermore, by exploiting and experimenting with the different forms of duality between

- a language and its reversal,
- prefixes and suffixes,
- rows and columns, and
- equivalence classes and residual languages

and their interconnections we would like to underline the universal useability of the concept of observation table as a means to execute and record such a retrieval process at the same time without having to deal with the idiosyncrasies of an underlying mechanism such as an automaton (for an extensive discussion concerning observation tables and other kinds of suitable representations also see [14]).

2 Basic notions and definitions

Definition 1 *An* observation table *is a triple* $T = (S, E, obs)$ *with* $S, E \subseteq \Sigma^*$ *finite, non-empty for some alphabet* Σ *and* $obs : S \times E \longrightarrow \{0, 1\}$ *a function with*

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

For an observation table $T = (S, E, obs)$*, the* row *of an element* $s \in S$ *is* $row(s) := \{(e, obs(s, e)) | e \in E\}$*, and the* column *of an element* $e \in E$ *is*

$col(e) := \{(s, obs(s,e))|s \in \text{RED}\}$. *Let* $row(S)$ *denote the set* $\{row(s)|s \in S\}$ *and* $col(E)$ *the set* $\{row(e)|e \in E\}$.

S is partitioned into two sets RED and BLUE where $uv \in \text{RED} \Rightarrow u \in \text{RED}$ for $u, v \in \Sigma^*$ (*prefix-closedness*), and $\text{BLUE} := \{sa \in S \setminus \text{RED}|s \in \text{RED},\ a \in \Sigma\}$, i.e., BLUE contains the one-symbol extensions of RED elements that are not in RED.

Definition 2 *Two elements* $r, s \in S$ *are* obviously different *(denoted by* $r <> s$*) iff* $\exists e \in E$ *such that* $obs(r,e) \neq obs(s,e)$.

Definition 3 *Let* $T = (S, E, obs)$ *with* $S = \text{RED} \cup \text{BLUE}$ *be an observation table.*

- T *is* closed *iff* $\neg\exists s \in \text{BLUE} : \forall r \in \text{RED} : r <> s$.
- T *is* consistent *iff* $\forall s_1, s_2 \in \text{RED},\ s_1a, s_2a \in S,\ a \in \Sigma : row(s_1) = row(s_2) \Rightarrow row(s_1a) = row(s_2a)$.

Definition 4 *A* finite-state automaton *is a tuple* $\mathcal{A} = (\Sigma, Q, Q_0, F, \delta)$ *with*

- *finite input alphabet* Σ,
- *finite non-empty state set* Q,
- *set of start states* $Q_0 \subseteq Q$,
- *set of final accepting states* $F \subseteq Q$, *and*
- *a transition function* $\delta : Q \times \Sigma \longrightarrow 2^Q$.

If Q_0 *is a singleton and* δ *maps a set containing at most one state to any pair in* $Q \times \Sigma$ *the automaton is* deterministic *(a* DFA*), otherwise* non-deterministic *(a* NFA*). If* δ *maps a non-empty set of states to every pair in* $Q \times \Sigma$ *the automaton is* total*, otherwise* partial.

The transition function can always be extended to $\delta : Q \times \Sigma^* \longrightarrow 2^Q$ *defined by* $\delta(q, \varepsilon) = \{q\}$ *and* $\delta(q, wa) = \delta(\delta(q, w), a)$ *for* $q \in Q$, $a \in \Sigma$, *and* $w \in \Sigma^*$.

Let $\delta(Q', w)$ *denote the set* $\bigcup\{\delta(q, w)|q \in Q'\}$ *for* $Q' \subseteq Q$ *and* $w \in \Sigma^*$. *A state* $q \in Q$ *is* reachable *if there is a string* $w \in \Sigma^*$ *such that* $q \in \delta(Q_0, w)$. *A state* $q \in Q$ *is* useful *if there are strings* $w_1, w_2 \in \Sigma^*$ *such that* $q \in \delta(Q_0, w_1)$ *and* $\delta(q, w_2) \cap F \neq \emptyset$, *otherwise* useless.

The language accepted by $\mathcal{A}$ *is* $\mathcal{L}(\mathcal{A}) = \{w \in \Sigma^*|\delta(Q_0, w) \cap F \neq \emptyset\}$. *A language that is accepted by a finite-state automaton is generally referred to as* recognizable *or* regular.

From any observation table $T = (S, E, obs)$ with $S = \text{RED} \cup \text{BLUE}$ and $\varepsilon \in E$ we can derive a finite-state automaton $\mathcal{A}_T = (\Sigma, Q_T, Q_{T0}, F_T, \delta_T)$ defined by

- $Q_T = row(\text{RED})$,
- $Q_{T0} = \{row(\varepsilon)\}$,
- $F_T = \{row(s)|obs(s, \varepsilon) = 1,\ s \in \text{RED}\}$, and

- $\delta_T(row(s), a) = \{q \in Q_T | \neg(q <> row(sa)),\ s \in \text{RED},\ a \in \Sigma,\ sa \in S\}$.

$\mathcal{A}_T$ is deterministic iff T is consistent (this follows straight from the definition of δ_T). The DFA for a regular language L derived from a closed and consistent table has the minimal number of states (see [1], Theorem 1). This automaton is also called the *canonical DFA $\mathcal{A}_L$ for L* and is unique up to a bijective renaming of states. However, if $\mathcal{A}_L$ is required to be total it must contain a "failure state" receiving all non-prefixes of L (if there are any).

The Myhill-Nerode equivalence relation $\equiv_L$ is defined as follows:

$$r \equiv_L s \text{ iff } re \in L \Leftrightarrow se \in L \text{ for all } r, s, e \in \Sigma^*.$$

The *index* of L is $I_L := |\{[s_0]_L | s_0 \in \Sigma^*\}|$ where $[s_0]_L$ is the equivalence class containing s_0.

Theorem 1 (**Myhill-Nerode theorem** – see for example [4])
I_L is finite iff L is a regular language, i.e., iff it can be recognized by a finite-state automaton.

The total canonical DFA $\mathcal{A}_L$ has exactly I_L states, and each state can be seen to represent an equivalence class under $\equiv_L$. Observe the correspondence between the table $T = (S, E, obs)$ representing $\mathcal{A}_L$ and the equivalence classes of L under $\equiv_L$: S contains strings whose rows are candidates for *states* in $\mathcal{A}_L$, and the elements of E – also called '*contexts*' in the following – can be taken as *experiments* which prove that two strings in S do indeed belong to different equivalence classes and that thus their rows should represent two different states.

Definition 5 *The* reversal $\overline{w}$ *of a string $w \in \Sigma^*$ is defined inductively by $\overline{\varepsilon} := \varepsilon$ and $\overline{aw} := \overline{w}a$ for $a \in \Sigma$, $w \in \Sigma^*$. The reversal of a set $X \subseteq \Sigma^*$ is defined as $\overline{X} := \{\overline{w} | w \in X\}$. The reversal of an automaton $\mathcal{A} = (\Sigma, Q, Q_0, F, \delta)$ is defined as $\overline{\mathcal{A}} := (\Sigma, Q, F, Q_0, \overline{\delta})$ with $\overline{\delta}(q', w) = \{q \in Q | q' \in \delta(q, w)\}$ for $q' \in Q$ and $w \in \Sigma^*$.*

Obviously, $\overline{\mathcal{L}(\mathcal{A})} = \mathcal{L}(\overline{\mathcal{A}})$. Note that due to the duality of left and right congruence the reversal of a regular string language must be regular as well.

Definition 6 *The* residual language *of a language $L \subseteq \Sigma^*$ with regard to a string $w \in \Sigma^*$ is defined as $w^{-1}L := \{v \in \Sigma^* | wv \in L\}$. A residual language $w^{-1}L$ is called* prime *iff $\bigcup\{v^{-1}L | v^{-1}L \subsetneq w^{-1}L\} \subsetneq w^{-1}L$ (i.e., if it does* not *equal the union of other residual languages of L properly contained in it), otherwise it is called* composed.

The Myhill-Nerode theorem can be reinterpreted to show that the set of distinct residual languages of a language L is finite iff L is regular. There is a natural bijection between the residual languages of L and the states of the minimal DFA $\mathcal{A}_L = (\Sigma, Q_L, \{q_L\}, F_L, \delta_L)$ defined by the mapping $w^{-1}L \mapsto q'$ for all $w \in \Sigma^*$ with $\delta_L(q_L, w) = \{q'\}$.

Let $L_q := \{w | \delta(q, w) \cap F \neq \emptyset\}$ for some regular language $L \subseteq \Sigma^*$, some automaton $\mathcal{A} = (\Sigma, Q, Q_0, F, \delta)$ recognizing L, and $q \in Q$.

Definition 7 *A* residual finite-state automaton (RFSA) *is a NFA* $\mathcal{A} = (\Sigma, Q, Q_0, F, \delta)$ *such that* L_q *is a residual language of* $\mathcal{L}(\mathcal{A})$ *for all states* $q \in Q$*.*

Definition 8 *The* canonical RFSA $\mathcal{R}_L = (\Sigma, Q_R, Q_{R0}, F_R, \delta_R)$ *for* $L \subseteq \Sigma^*$ *is defined by*

- $Q_R = \{w^{-1}L | w^{-1}L \text{ is prime}\}$,
- $Q_{R0} = \{w^{-1}L \in Q_R | w^{-1}L \subseteq L\}$,
- $F_R = \{w^{-1}L | \varepsilon \in w^{-1}L\}$, *and*
- $\delta_R(w^{-1}L, a) = \{v^{-1}L \in Q_R | v^{-1}L \subseteq (wa)^{-1}L\}$ *for* $a \in \Sigma$*.*

$\mathcal{R}_L$ is minimal with respect to the number of states (see [5], Theorem 1). Note that as empty residual languages correspond to failure states and prime residual languages are non-empty by definition the canonical RFSA $\mathcal{R}_L$ never contains a failure state.

3 Inferring a RFSA using an observation table

3.1 A "parasitic" two-step algorithm

The algorithm we are going to present in this subsection infers the canonical RFSA for some regular language L from a suitable combination of available information sources. An information source can be an oracle answering membership queries ('Is this string contained in the language?') or equivalence queries ('Is A a correct automaton for L?' – including the return of a counterexample $c \in (L \setminus \mathcal{L}(A)) \cup (\mathcal{L}(A) \setminus L)$ in case of a negative answer) or a positive or negative sample of L fulfilling certain properties, and there are possibly other kinds of sources that could be taken into consideration as well. Suitable combinations known from the literature are for example an oracle answering membership and equivalence queries (a so-called *minimally adequate teacher*, or MAT), an oracle answering membership queries and positive data, or positive and negative data (for references see below).

We proceed as follows: In a first step we use an existing algorithm in order to build an observation table $T' = (\text{RED}' \cup \text{BLUE}', E', obs')$ representing the canonical DFA for the reversal $\overline{L}$ of L. Eligible algorithms for various learning settings can be found for example in [1] (Angluin's seminal algorithm L^*, for MAT learning), [13] (ALTEX, an algorithm learning regular tree languages from a membership oracle and positive data, which can be readapted to strings by treating them as non-branching trees), or [17] (the meta-algorithm GENMODEL covers MAT learning, learning from membership queries and positive data, and learning from positive and negative data, and can be adapted to a range of other combinations of information sources as well). All of these algorithms are based on the principle of adding elements to the set labeling the rows of the table (representing candidates for states in the canonical DFA) until it is

	e	e_1	e_2	e_3	e_4
s_1	1	0	1	1	0
s_2	1	1	0	1	1
s_3	1	0	1	0	0
s_4	0	0	0	0	1

Figure 1: An example for a coverable column (labeled by e)

closed, and/or separating contexts (i.e., suffixes by which it can be shown that two states should be distinct in the derived automaton) to the set labeling the columns of the table until it is consistent – additions of one kind potentially resulting in the necessity of the other and vice versa – and, once the table is both closed and consistent, deriving an automaton from it that is either the canonical DFA in question or can be rejected by a counterexample obtained from one of the available information sources, which is then evaluated and used to start the cycle afresh. Obviously, since the sources only provide information about L and not its reversal $\overline{L}$, we must interfere with the process in a minimal way by adapting data and queries accordingly: Strings and automata have to be reversed (see Section 2) before submitting them to an oracle, as well as given samples and counterexamples have to be reversed before using them in the construction of the table T'.

In the second step we submit T' to the following modifications:

- ▷ Only keep one representative for every distinct row occurring in the table in RED' (but leave BLUE' unchanged), and only keep one representative for every distinct column in E'. The choice of which element to keep can be simplified for example by always settling on the first element of the set in length-lexical order.
- ▷ Eliminate all representatives of rows and columns containing only 0s. Let the resulting table be $T'' = (\text{RED}'' \cup \text{BLUE}'', E'', obs'')$.
- ◊ Eliminate all representatives of *coverable* columns, i.e., all $e \in E''$ with

  $$\begin{aligned}&\exists e_1, \ldots, e_n \in E'' : \forall s \in \text{RED}'' : \\ &[obs''(s, e) = 0 \Rightarrow \forall i \in \{1, \ldots, n\} : obs''(s, e_i) = 0] \wedge \\ &[obs''(s, e) = 1 \Rightarrow \exists i \in \{1, \ldots, n\} : obs''(s, e_i) = 1].\end{aligned}$$

  For example, the column labeled by e in Figure 1 would be eliminated because its 1s are all "covered" by the columns labeled by e_1, e_2, and e_3.

Note that the first two modifications mainly serve to trim down the table in order to make the third modification less costly. In fact, most algorithms mentioned above can easily be remodeled such that they build tables in which there are no rows or columns consisting of 0s anyway (as rows consisting of 0s correspond to the failure state and columns containing only 0s cannot make any distinction

between elements labeling the rows), and in which the elements labeling the rows in the RED part of the table are all pairwise obviously different already such that no row is represented twice.

The table thus modified shall be denoted by $T = (\text{RED} \cup \text{BLUE}, E, obs)$ and the automaton derived from it by $\mathcal{A}_T = (\Sigma, Q_T, Q_{T0}, F_T, \delta_T)$ with $F_T = F_{T'}$ (this has to be stated explicitly for the case in which ε has been eliminated by $\Diamond$). Since we have kept a representative for every distinct row and since all pairs of RED$'$ elements that are distinguished by the contexts eliminated by $\Diamond$ must be distinguished by at least one of the contexts covering those as well $\mathcal{A}_T$ still represents the canonical DFA for $\overline{L}$ (but without a failure state).

We use the new table T to define a NFA $\mathcal{R} = (\Sigma, Q_R, Q_{R0}, F_R, \delta_R)$ with

- $Q_R = \{q \subseteq \text{RED} | \exists e \in E : s \in q \Leftrightarrow obs(s, e) = 1\}$,
- $Q_{R0} = \{q \in Q_R | \forall s \in q : obs'(s, \varepsilon) = 1\}$
  (here we have to consult obs' again in case ε has been eliminated by $\Diamond$),
- $F_R = \{q \in Q_R | \varepsilon \in q\}$, and
- $\delta_R(q_1, a) = \{q_2 | q_2 \subseteq \overline{\delta_T}(q_1, a)\}$ for $q_1, q_2 \in Q_R$ and $a \in \Sigma$, and $\overline{\delta_T}$ is the transition function of the reversal of $\mathcal{A}_T$.

Observe that every state in Q_R corresponds to a column in T. Since every element of RED represents an equivalence class of $\overline{L}$ under the Myhill-Nerode relation every state in Q_R also corresponds to a unique set of equivalence classes, and the associated column can be seen to represent the characteristic function of that set. The following theorem states the main result of the paper:

Theorem 2 *$\mathcal{R}$ is (isomorphic to) the canonical RFSA for L.*

The proof will make use of Theorem 3 from [5], repeated as Theorem 3 below.

Definition 9 *Let $A = (\Sigma, Q, Q_0, F, \delta)$ be a NFA, and define*

$$Q^{\diamond} := \{p \subseteq Q | \exists w \in \Sigma^* : \delta(Q_0, w) = p\}.$$

A state $q \in Q^{\diamond}$ is said to be coverable *iff there exist $q_1, \ldots, q_n \in Q^{\diamond} \setminus \{q\}$ for $n \geq 1$ such that $q = \bigcup_{i=1}^{n} q_i$. Otherwise q is said to be* non-coverable.

Theorem 3 *Let L be a regular language and let $B = (\Sigma, Q_B, Q_{B0}, F_B, \delta_B)$ be a NFA such that $\overline{B}$ is a RFSA recognizing $\overline{L}$ whose states are all reachable. In that case $C(B) = (\Sigma, Q_C, Q_{C0}, F_C, \delta_C)$ with*

- $Q_C = \{p \in Q_B^{\diamond} | p \text{ is not coverable}\}$,
- $Q_{C0} = \{p \in Q_C | p \subseteq Q_{B0}\}$,
- $F_C = \{p \in Q_C | p \cap F_B \neq \emptyset\}$, *and*
- $\delta_C(p, a) = \{p' \in Q_C | p' \subseteq \delta_B(p, a)\}$ *for $p \in Q_C$ and $a \in \Sigma$*

is the canonical RFSA recognizing L.

As a further important result it has also been shown in [5], Section 5, that in a RFSA for some regular language L whose states are all reachable the non-coverable states correspond exactly to the prime residual languages of L and that consequently Q_C can be naturally identified with the set of states of the canonical RFSA for L. **Proof of Theorem 2:** Observe that $\mathcal{A}_T$ meets the conditions for $\overline{B}$ in Theorem 3:

- All states of $\mathcal{A}_T$ are reachable because $\mathcal{A}_T$ contains no useless states,
- $\mathcal{A}_T$ is a RFSA because every DFA without useless states is a RFSA ([5], Section 3),
- and $\mathcal{L}(\mathcal{A}_T) = \overline{L}$.

Since $\mathcal{A}_T$ contains no useless states $\mathcal{A}_T$ and $\overline{\mathcal{A}_T}$ have the same number of states and transitions, and therefore we can set $B = \overline{\mathcal{A}_T} = (\Sigma, Q_T, F_T, Q_{T0}, \overline{\delta_T})$. Assuming for the present that there is a bijection between Q_R and Q_C it is rather trivial to see that

- there is a bijection between $Q_{R0} = \{q \in Q_R | \forall x \in q : obs'(x, \varepsilon) = 1\}$ and $Q_{C0} = \{p \in Q_C | p \subseteq F_T\}$ due to $F_T = \{x \in \text{RED} | obs'(x, \varepsilon) = 1\}$,
- there is a bijection between $F_R = \{q \in Q_R | \varepsilon \in q\}$ and $F_C = \{p \in Q_C | p \cap Q_{T0} \neq \emptyset\}$ due to the fact that $Q_{T0} = \{\varepsilon\}$, and that
- for every $q \in Q_R$, $p \in Q_C$, and $a \in \Sigma$ such that q is the image of p under the bijection between Q_R and Q_C, $\delta_R(q, a) = \{q_2 \in Q_R | q_2 \subseteq \overline{\delta_T}(q, a)\}$ is the image of $\delta_C(p, a) = \{p' \in Q_C | p' \subseteq \overline{\delta_T}(p, a)\}$.

It remains to show that there is indeed a bijection between Q_R and the set of prime residual languages of L, represented by Q_C. First of all, consider Proposition 1 from [5]:

Lemma 1 *Let $A = (\Sigma, Q, Q_0, F, \delta)$ be a RFSA. For every prime residual language $w^{-1}\mathcal{L}(A)$ there exists a state $q \in \delta(Q_0, w)$ such that $L_q = w^{-1}\mathcal{L}(A)$.*

From the definition of Q_R it is clear that $\mathcal{R}$ is a RFSA: As noted above, every state in Q_R corresponds to a column in T, labeled by a context $e \in E$, and also to the set of equivalence classes $[s]_{\overline{L}}$ such that $se \in \overline{L}$ for $s \in \text{RED}$. As a consequence the reversal of the union of this set of equivalence classes equals the residual language $\overline{e}^{-1}L$, and therefore every state in Q_R corresponds to exactly one residual language of L. According to Lemma 1, there is a state in Q_R for each prime residual language of L, and hence every prime residual language of L is represented by exactly one column occurring in the table.
By $\Diamond$ we have eliminated the columns that are covered by other columns in the table. If a column is not coverable in the table the corresponding state in Q_R is not coverable either: Consider a column in the table which can be covered by a set of columns of which at least some do not occur in the table. Due to Lemma

1, these columns can only correspond to composed residual languages of L. If we were to add representatives of these columns to the table they would have to be eliminated again directly because of the restrictions imposed by $\Diamond$. This means that if a column is coverable at all it can always be covered completely by restricting oneself to columns that correspond to prime residual languages of L as well, and these are all represented in the table. Therefore Q_R cannot contain any coverable states.
Consequently, the correspondence between Q_R and the set of prime residual languages of L is one-to-one, and we have shown that $\mathcal{R}$ is isomorphic to the canonical RFSA for L. □

Corollary 1 *Let L be a regular language. The number of prime residual languages of L equals the minimal number of contexts that is needed in order to distinguish between the states of the canonical DFA for L.*

A remark: If the algorithm generating the original table yields a partial automaton (which is the general case for ALTEX and also for GENMODEL in the settings of learning from membership queries and positive data, and from positive and negative data) then of course the canonical RFSA cannot be maximal with respect to the transitions either.

Also note that we can skip the relatively cumbersome third modification $\Diamond$ in the second part of our algorithm if we restrict the target to a subclass of the regular languages:

Definition 10 *A DFA is called* bideterministic *(also known as* 0-reversible*) iff its reversal is deterministic as well. A language is bideterministic if there is a biDFA recognizing it.*

Theorem 4 *All residual languages of a bideterministic language are disjoint.*

(See [2]). This implies that a table for a bideterministic language cannot contain coverable columns and that for every column the residual language represented by it must be prime.

3.2 Comparison to other algorithms: Query complexity

An obvious advantage of the algorithm described above is the trivial fact that it directly benefits from any past, present, and future research on algorithms that infer minimal DFAs via observation tables, and at least until the present moment there is a huge gap between the amount of research that has been done on algorithms inferring DFAs and the amount of research on algorithms inferring NFAs – or RFSAs, for that matter.

One point of interest in connection with the concepts presented here is the study of further kinds of information sources that could be used as input and in particular suitable combinations thereof (see [17] for a tentative discussion).

Another point of interest is complexity. Since the second part of the algorithm consists of some very cheap comparisons of 0s and 1s only of which $\Diamond$

is the most complex[1] the determining factor is of course the complexity of the chosen underlying algorithm. One of the standard criteria for evaluating an algorithm is its time complexity, but depending on the different learning settings there are other measures that can be taken into consideration as well, one of which we will briefly address in the following.

For algorithms that learn via queries a good criterion is the number of queries needed, obviously. The prototype of all query learning algorithms, Angluin's [1] MAT learning algorithm L^*, which can be seen in a slightly adapted[2] version L^*_{col} in Figure 2, needs I_L equivalence queries and roughly $O(|\Sigma| \cdot |c_0| \cdot I_L^2)$ membership queries, where I_L is the index of the regular language $L \subseteq \Sigma^*$ and $|c_0|$ is the length of the longest given counterexample. By some modifications the number of membership queries can be improved to $O(|\Sigma| I_L^2 + I_L log|c_0|)$ which according to [14] is optimal up to constant factors. On the other hand, it has been shown in [15] that it is possible to decrease the number of equivalence queries to sublinearity at the price of increasing the number of membership queries exponentially.

initialize $T := (S, E, obs)$ `with` $S =$ `RED` $\cup$ `BLUE` `and` `BLUE` $=$ `RED` $\cdot \Sigma$
  by `RED` $:= \{\varepsilon\}$ `and` $E := \{\varepsilon\}$
repeat until `EQ` $=$ `yes`
  while T `is not closed and not consistent`
    if T `is not closed`
      `find` $s \in$ `BLUE` `such that` $row(s) \notin row($`RED`$)$
      `RED` $:=$ `RED` $\cup \{s\}$ *(and update the table via MQs)*
    if T `is not consistent`
      `find` $s_1, s_2 \in$ `RED`, $a \in \Sigma$, $e \in E$ `such that` $s_1 a, s_2 a \in S$
        `and` $\neg(s_1 <> s_2)$ `and` $obs(s_1 ae) \neq obs(s_2 ae)$
      $E := E \cup \{ae\}$ *(and update the table via MQs)*
  perform equivalence test
  if `EQ` $=$ `0 get counterexample` $c \in (L \setminus \mathcal{L}(\mathcal{A}_T)) \cup (\mathcal{L}(\mathcal{A}_T) \setminus L)$
    $E := E \cup Suff(c)$ *(and update the table via MQs)*
return $\mathcal{A}_T$

Figure 2: L^*_{col}

Recently, Bollig et al. [9] have presented a MAT learning algorithm for RF-

[1] Algorithmically, $\Diamond$ could be handled as follows: For every $e \in E''$ build the set $\{e_1, \ldots, e_n\} \subseteq E''$ such that $\forall i \in \{1, \ldots, n\} : \forall s \in \text{RED}'' : obs''(s, e) = 0 \Rightarrow obs''(s, e_i) = 0$. Then check if for all $s \in \text{RED}''$ with $obs''(s, e) = 1$ there is $i \in \{1, \ldots, n\}$ with $obs''(s, e_i) = 1$. If the test is positive eliminate e from E''. This can be done in polynomial time and is obviously bounded by the number of rows and columns in the table. It has been shown experimentally in [7] that for languages recognized by randomly generated DFAs the number of inclusion relations between residual languages represented in that DFA is extremely small, which gives rise to the hope that the full test does not have to be executed too often at all.

[2] The adaptation consists in the fact that instead of adding the counterexample and all its prefixes to S it is added with all its suffixes to E. This idea is due to [16] and does not change the result.

SAs using an observation table that keeps very close to the deterministic variant L^*_{col} mentioned above. To this end, they introduce the notions of *RFSA-closedness* and *RFSA-consistency*.

Definition 11 *Let $T = (S, E, obs)$ be an observation table. We say that a row labeled by $s \in S$ is coverable iff $\exists s_1, \ldots, s_n \in S$ (or is coverable by the rows of $s_1, \ldots, s_n \in S$ iff)*

$$\forall e \in E : [obs(s,e) = 0 \Rightarrow \forall i \in \{1, \ldots, n\} : obs(s_i, e) = 0] \wedge \\ [obs(s,e) = 1 \Rightarrow \exists i \in \{1, \ldots, n\} : obs(s_i, e) = 1].$$

Let $ncov(S) \subseteq row(S)$ denote the set of non-coverable rows labeled by elements from S.

Definition 12 *Let $T = (S, E, obs)$ be an observation table. Let us say that a row $r \in row(S)$* includes *another row $r' \in row(S)$, denoted by $r' \sqsubseteq r$, iff $\forall e \in E : obs(s', e) = 1$ $\Rightarrow obs(s, e) = 1$ for all $s, s' \in S$ with $row(s) = r$ and $row(s') = r'$.*

Definition 13 *A table $T = (\text{RED} \cup \text{BLUE}, E, obs)$ is* RFSA-closed *iff every row $r \in row(\text{BLUE})$ is coverable by some rows $r_1, \ldots, r_n \in ncov(\text{RED})$.*

Definition 14 *A table $T = (\text{RED} \cup \text{BLUE}, E, obs)$ is* RFSA-consistent *iff $row(s_1) \sqsubseteq row(s_2)$ implies $row(s_1 a) \sqsubseteq row(s_2 a)$ for all $s_1, s_2 \in S$ and all $a \in \Sigma$.*

From a RFSA-closed and RFSA-consistent table $T = (\text{RED} \cup \text{BLUE}, E, obs)$ Bollig et al. derive a non-deterministic finite-state automaton $\mathcal{R} = (\Sigma, Q_R, Q_{R0}, F_R, \delta_R)$ defined by

- $Q_R = ncov(\text{RED})$,
- $Q_{R0} = \{r \in Q_R | r \sqsubseteq row(\varepsilon)\}$,
- $F_R = \{r \in Q_R | \forall s \in \text{RED} : row(s) = r \Rightarrow obs(s, \varepsilon) = 1\}$, and
- $\delta_R(row(u), a) = \{r \in Q_R | r \sqsubseteq row(sa)\}$ with $row(s) \in Q_R$ and $a \in \Sigma$.

Theorem 5 ([9]) *Let T be a RFSA-closed and RFSA-consistent table and let $\mathcal{R}_T$ be the NFA derived from T. Then $\mathcal{R}_T$ is a canonical RFSA.*

The pseudo-code of the algorithm NL^* by Bollig et al. is given in Figure 3. See [9] for the proof of Theorem 5 and the proof that the automaton derived from the table established by NL^* is indeed the canonical RFSA for the regular language L in question.

When comparing NL^* and the corresponding concepts given above to the algorithm described in the previous subsection one should not fail to observe how establishing and exploiting a table for L itself instead of its reversal also brings about an exchange of the roles of rows and columns as representatives of states in the resulting RFSA.

initialize $T := (S, E, obs)$ `with` $S = \texttt{RED} \cup \texttt{BLUE}$ `and` $\texttt{BLUE} = \texttt{RED} \cdot \Sigma$
  by $\texttt{RED} := \{\varepsilon\}$ `and` $E := \{\varepsilon\}$
repeat until `EQ = yes`
  while T `is not RFSA-closed and not RFSA-consistent`
    if T `is not RFSA-closed`
      `find` $s \in \texttt{BLUE}$ `such that` $row(s) \in ncov(S) \setminus ncov(\texttt{RED})$
      $\texttt{RED} := \texttt{RED} \cup \{s\}$ *(and update the table via MQs)*
    if T `is not RFSA-consistent`
      `find` $s \in S$, $a \in \Sigma$, $e \in E$ `such that` $obs(sae) = 0$ `and`
        $obs(s'ae) = 1$ `for some` $s' \in S$ `with` $row(s') \sqsubseteq row(s)$
      $E := E \cup \{ae\}$ *(and update the table via MQs)*
  perform equivalence test
  if `EQ = 0 get counterexample` $c \in (L \setminus \mathcal{L}(\mathcal{A}_T)) \cup (\mathcal{L}(\mathcal{A}_T) \setminus L)$
    $E := E \cup Suff(c)$ *(and update the table via MQs)*
return $\mathcal{A}_T$

Figure 3: NL^*, the NFA (RFSA) version of L^*_{col}

The theoretical query complexity of NL^* amounts to at most $O(I_L^2)$ equivalence queries and $O(|\Sigma| \cdot |c_0| \cdot I_L^3)$ membership queries. This exceeds the maximal number of queries needed by L^*_{col} in both cases which is due to the fact that with NL^* adding a context does not always directly lead to an increase of the number of states in the automaton derived from the table. However, the authors of [9] also show that their algorithm statistically outperforms L^*_{col} in practice, which is partly due to the fact that the canonical RFSA for a language is often significantly smaller than its canonical DFA (see [5]). Nevertheless it is a noteworthy fact that apparently inferring an automaton with potentially exponentially less states than the minimal DFA for a language seems to be at least as complex or even more so!

Inspired by Bollig et al. [9] we propose another parasitic two-step algorithm that uses an existing algorithm with access to a membership oracle to establish a table $T' = (\text{RED}' \cup \text{BLUE}', E', obs')$ representing the canonical DFA for a language L and modifies it as follows:

- $\triangleright'$ Eliminate all representatives of rows and columns containing only 0s.
- $\Diamond'$ For every $s \in \text{RED}'$ and every final state q_F of $\mathcal{A}_{T'}$ add an (arbitrary) string e to E' such that $\delta_{T'}(row(s), e) = \{q_F\}$. Fill up the table via membership queries.

Let $T = (\text{RED} \cup \text{BLUE}, E, obs)$ be the resulting table. Note that since T' already contains the maximal number of possible distinct rows T is still closed and therefore RFSA-closed (this is easy to see from the definitions of closedness and RFSA-closedness). T is RFSA-consistent as well: Recall that every element $s \in S$ represents a residual language $s^{-1}L$ of L (see Section 2). If T was not RFSA-consistent we could find elements $s_1, s_2 \in S$, $e \in E$, and $a \in \Sigma$ with

$row(s_1) \sqsubseteq row(s_2)$ but $obs(s_1a, e) = 1 \wedge obs(s_2a, e) = 0$. However, $ae \in s_1^{-1}L$ and $row(s_1) \sqsubseteq row(s_2)$ imply that $ae \in s_2^{-1}L$, and hence $obs(s_2a, e) = 0$ cannot be true.

From the table T we derive an automaton $\mathcal{R} = (\Sigma, Q_R, Q_{R0}, F_R, \delta_R)$ as in [9] (see above). The NFA $\mathcal{R}$ is the canonical RFSA for L. This follows directly from Theorem 5 and the fact that T contains a representative for every residual language of L.

The algorithm outlined above needs $I_L \cdot |F_L|$ membership queries in addition to the number of queries needed by the algorithm establishing the original table but it does not require any more equivalence queries. As equivalence queries are usually deemed very expensive (deciding the equivalence of two NFAs is even PSPACE-complete, see for example [4]) this can be counted as an advantage. Also note that if we restrict the target to bideterministic languages again the table does not have to be modified and no additional queries have to be asked at all which is due to the result that for bideterministic languages the canonical DFA and the canonical RFSA coincide (see [11]).

4 Conclusion

Two-step algorithms have the advantage of modularity: Their components can be exchanged and improved individually and therefore more easily adapted to different settings and inputs as well whereas non-modular algorithms are generally stuck with their parameters. One may doubt the efficiency of the two-step algorithms presented here by observing that the second step partly destroys the work of the first, but as we have seen, as long as algorithms inferring the minimal DFA are so much less complex than the ones inferring the minimal RFSA the two-step version still outperforms the direct one.

Concerning future research: It should be relatively easy to adapt NL^* to other learning settings such as learning from positive data and a membership oracle or from positive and negative data in order to establish a more universal pattern for algorithms that infer a RFSA via an observation table similar to the generalization for DFAs attempted in [17].
The definition of RFSA has been extended to trees in [10] where the authors also announce the development of corresponding learning algorithms. For trees the role of concatenation operation is taken by the replacement of a special node in a tree context by another tree:

Definition 15 *The set T_Σ of trees over a ranked alphabet Σ is defined as the smallest set with $f \in T_\Sigma$ for every $f \in \Sigma_0$ and $f[t_1, \ldots, t_n] \in T_\Sigma$ for every $f \in \Sigma_n$ and $t_1, \ldots, t_n \in T_\Sigma$. Let $\Box$ be a special symbol of rank 0. A tree $c \in T_{\Sigma \cup \{\Box\}}$ in which $\Box$ occurs exactly once is a* context*. For a context c and $s \in T_\Sigma$, $c[[s]]$ is the tree obtained by substituting s for $\Box$ in c.*

Definitions 11 to 13 can be applied in the tree case directly.

Definition 16 *A table $T = (\text{RED} \cup \text{BLUE}, E, obs)$ is* RFSA-consistent *iff $row(s_i) \sqsubseteq$*

$row(t_i) \Rightarrow row(f[s_1, \ldots, s_i, \ldots, s_n]) \sqsubseteq row(f[t_1, \ldots, t_i, \ldots, t_n])$ *for all* $s_i, t_i \in S$, $1 \leq i \leq n$, $f \in \Sigma_n$.

The notions of RFSA-closedness and RFSA-consistency once carried over the adaptation of algorithms using this notions (like the ones described in Subsection 3.2) to ordinary two-dimensional trees and, as is made evident in [18] and [19], even to trees over arbitrarily many dimensions (so-called *multi-dimensional trees*) should be fairly straightforward as well. Unfortunately this is not the case for our algorithm from Subsection 3.1 which is due to the fact that it seems rather difficult to present a sensible definition of the reversal of a tree.

References

[1] Angluin, D.: Learning regular sets from queries and counterexamples. Information and Computation 75(2), 87–106 (1987)

[2] Angluin, D.: Inference of reversible languages. JACM, vol. 29(3), pp. 741–765 (1982)

[3] Drewes, F., Högberg, J.: Learning a regular tree language from a teacher. In: DLT 2003. LNCS, vol. 2710, pp. 279–291. Springer (2003)

[4] Hopcroft, J. E. and Ullmann, J. D.: Introduction to Automata Theory, Languages, and Computation. Addison-Wesley Longman (1990)

[5] Denis, F., Lemay, A., and Terlutte, A.: Residual Finite State Automata. In: STACS 2001. LNCS, vol. 2010, pp. 147–155. Springer (2001)

[6] Denis, F., Lemay, A., and Terlutte, A.: Learning regular languages using non-deterministic finite automata. In: ICGI 2000. LNCS, vol. 1891, pp. 39–50. Springer (2000)

[7] Denis, F., Lemay, A., and Terlutte, A.: Learning regular languages using RFSA. In: ALT 2001. LNCS, vol. 2225, pp. 348–363. Springer (2001)

[8] Denis, F., Lemay, A., and Terlutte, A.: Some classes of regular languages identifiable in the limit from positive data. In: Grammatical Inference – Algorithms and Applications. LNCS, vol. 2484, pp. 269–273. Springer (2003)

[9] Bollig, B., Habermehl, P., Kern, C., and Leucker, M.: Angluin-style learning of NFA. In: Online Proceedings of IJCAI 21 (2009). Available from: `http://ijcai.org/papers09/contents.php`

[10] Carme, J., Gilleron, R., Lemay, A., Terlutte, A., and Tommasi, M.: Residual finite tree automata. In: DLT 2003. LNCS, vol. 2710, pp. 171–182. Springer (2003)

[11] Latteux, M., Roos, Y., and Terlutte, A.: Minimal NFA and biRFSA languages. RAIRO Theoretical Informatics and Applications, vol. 43, pp. 221–237 (2009)

[12] Latteux, M., Lemay, A., Roos, Y., and Terlutte, A.: Identification of biRFSA languages. TCL, vol. 356(1), pp. 212–223 (2006)

[13] Besombes, J. and Marion, J.-Y.: Learning Tree Languages from Positive Examples and Membership Queries. In: ALT 2003. LNCS, vol. 3244, pp. 440–453. Springer (2003)

[14] Balcazar, J.L., Diaz, J., Gavalda, R., and Watanabe, O.: Algorithms for learning finite automata from queries – a unified view. In: Advances in Algorithms, Languages, and Complexity, pp. 53–72 (1997)

[15] Balcazar, J.L., Diaz, J., Gavalda, R., and Watanabe, O.: The query complexity of learning DFA. New Generation Computing, vol. 12(4), pp. 337–358. Springer (1994)

[16] Maler, O. and Pnueli, A.: On the learnability of infinitary regular sets. In: Proceedings of the 4th Annual Workshop on Computational Learning Theory, pp. 128–136. Morgan Kaufmann (1991)

[17] Kasprzik, A.: Meta-Algorithm GENMODEL: Generalizing over three learning settings using observation tables. Technical report 09-2, University of Trier (2009)

[18] Kasprzik, A.: Making finite-state methods applicable to languages beyond context-freeness via multi-dimensional trees. In J. Piskorski, B. Watson, A. Yli-Jyrä (eds): Post-proceedings of FSMNLP 2008, pp. 98–109. IOS Press (2009)

[19] Kasprzik, A.: A learning algorithm for multi-dimensional trees, or: Learning beyond context-freeness. In A. Clark, F. Coste, L. Miclet (eds): ICGI 2008. LNAI, vol. 5278, pp. 111–124. Springer (2008)