Efficient Inference of Mealy Machines

Gerco van Heerdt
4167503

Bachelor thesis in Computer Science
Radboud University Nijmegen

June 30, 2014

Supervisor/first assessor
Dr. A. Silva
alexandra@cs.ru.nl

Second assessor
Prof. Dr. F.W. Vaandrager
F.Vaandrager@cs.ru.nl

Abstract

Automata learning is increasingly being applied to ease the testing and comparing of complex systems. We formally reconstruct an efficient algorithm for the inference of Mealy machines, prove its correctness, and show that equivalence queries are not required for non-minimal hypotheses. In fact, we are able to evade those by applying a minor optimization to the algorithm. As a corollary, standard conformance testing methods can be used directly for equivalence query approximations within this algorithm.

1 Introduction

Model checking plays a central role in the verification of the correctness of software. The main condition required to apply this technique is the availability of a model of the system. In the form of an automaton, this model can be intersected with other automata describing illegal behavior in order to verify correctness properties [13]. However, in practice these models are often outdated or not created at all.

Automata learning provides a solution to this problem. The seminal algorithm of Angluin [5] learns a DFA in polynomial time by assuming the ability to make certain queries regarding the target language. This algorithmic pattern has been carried on in order to learn other types of automata, including infinitary languages [20], Mealy machines [25, 27, 29], more generic input/output automata [30, 23], and even certain data languages [10, 15]. Besides the possibilities for model checking [24, 14], Angluin's algorithm can be used for the derivation of program specifications [4] and the inference of security protocols [28].

Mealy machines [21] form a particularly interesting type of automata, as they are conveniently simple and yet able to effectively model many reactive systems. The inference algorithm may use an abstraction scheme [1] to deal with data parameters. This has, for example, been applied in order to learn models of the biometric passport [2] and bank cards [3].

These examples indicate the relevance of efficiency in terms of the number of queries made by the algorithm—these queries may require interaction with a physical system, which is often time-consuming. Several authors have proposed optimizations in this respect [26, 27, 29], but the formal proofs for (generalizations of) Mealy machines given by Vilar [30] and Niese [23] handle only adaptations of the original algorithm.

The main contribution of this thesis is a self-contained formal reconstruction of Angluin's algorithm for Mealy machines, focused on minimizing its query complexity. To achieve this, we combine the informal constructions presented by Steffen et al. [29] with formal proofs partially adapted from Angluin [5] and Niese [23].

After defining Mealy machines and their semantics (Section 3), we proceed to finding that the minimal Mealy machine (Section 3.1) provides a suitable canonical model to aim the inference algorithm at (Section 3.2). The proof for this directly yields a method to construct that minimal machine. Centered around this construction we build our adaptation of Angluin's algorithm for Mealy machines in Section 4. The algorithm constructs a series of hypothesis automata, whereof we explore the minimality in Section 4.1. In Section 4.2 we present a correctness proof that can easily be extended to certain variations on the algorithm. We then exploit this to adapt our algorithm and proofs to the optimized method proposed by Rivest and Schapire [26] (Section 4.3). This optimization sacrifices the guarantee of minimality

for intermediate hypotheses. In Section 4.3.1 we discuss efforts made to overcome this problem and conclude with a solution that preserves the efficiency of the algorithm. Finally, by analyzing the query complexities of the discussed algorithms we find in Section 4.4 that our adaptations inherit the efficiency of the original algorithms for regular languages. We conclude in Section 5 with a discussion on our contributions, related work, and possible future directions.

2 Preliminaries

First we recall some basic formal language theory and fix the relevant notation. An *alphabet* Σ is a set of symbols. We denote by Σ^* the set of all finite sequences of elements from Σ. Such a sequence is called a *string*. We use ε to denote the *empty string* and write $|u|$ for the length of a string u.

Let $u \cdot v$ denote the *concatenation* of two strings u and v. This operation appends the symbols of v to u, resulting in another string. For concrete strings in examples we will often leave out the operator, writing just uv instead. A *prefix* of a string u is a string v such that $u = v \cdot w$ for some string w. Similarly, a *suffix* of u is a string v such that for some string w we have $u = w \cdot v$.

A *language* over Σ is a subset of Σ^*. We define the language of all nonempty strings over Σ as $\Sigma^+ = \Sigma^* \setminus \{\varepsilon\}$. The concatenation of two languages L_1 and L_2 is given by

$$L_1 \cdot L_2 = \{u \cdot v \mid u \in L_1 \wedge v \in L_2\}.$$

We let $\varnothing$ denote the empty set and use the notation Y^X to refer to the set of functions $X \to Y$, where X and Y can be any two sets.

3 Mealy Machines

We fix a finite *input alphabet* A and an *output alphabet* B.

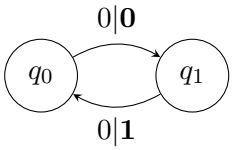
Definition 1. A *Mealy machine* is a tuple (Q, q_0, δ, γ), where

- Q is a finite set of *states*,
- $q_0 \in Q$ is the *initial state*,
- $\delta : Q \times A \to Q$ is the *transition function*, and
- $\gamma : Q \times A \to B$ is the *output function*.

The Mealy machine $M = (Q, q_0, \delta, \gamma)$ starts in the state q_0. At any point in time, M is in some state q. If it then reads the input symbol a, M outputs

the output symbol given by $\gamma(q, a)$ and advances to the state provided by $\delta(q, a)$.

Mealy machines can be visualized using a state transition graph such as the following, which defines a Mealy machine over $A = \{0\}$ and $B = \{\mathbf{0}, \mathbf{1}\}$.



The nodes in these graphs represent the states of the Mealy machine, where the label q_0 designates the initial state. The intended meaning of an arrow $q \xrightarrow{a|b} r$ for any $q, r \in Q$, $a \in A$, and $b \in B$ is to define $\delta(q, a) = r$ and $\gamma(q, a) = b$.

The observable behavior of a Mealy machine is characterized by the output strings it assigns to all non-empty input strings. Note that these output strings are completely determined by the last output symbols generated for all the prefixes of the corresponding input strings, so we can reduce the characterization by only giving the last output symbol produced for each input string. For example, we can fully describe the Mealy machine above by stating that it assigns the output symbol **0** to every input string with an even length and **1** to the input strings with an odd length.

For any Mealy machine $M = (Q, q_0, \delta, \gamma)$, let $\delta^*_M : A^* \to Q$ be the function that returns the state reached by M after processing a given input string. Formally, we define δ^*_M by

$$\begin{cases} \delta^*_M(\varepsilon) = q_0 \\ \delta^*_M(u \cdot a) = \delta(\delta^*_M(u), a) \qquad \text{for any } u \in A^* \text{ and } a \in A. \end{cases}$$

Definition 2 (Semantics of Mealy machines)**.** The semantics of a Mealy machine $M = (Q, q_0, \delta, \gamma)$ is given by the function $\gamma^+_M : A^+ \to B$, which for all $u \in A^*$ and $a \in A$ is defined by

$$\gamma^+_M(u \cdot a) = \gamma(\delta^*_M(u), a).$$

We say that M is a *realization* of γ^+_M. The functions realized by Mealy machines are called *rational functions*.

Additionally, we let $\Gamma_M : Q \to B^{A^+}$ provide semantics for specific states. For any $q \in Q$ and $a \in A$, we define

$$\begin{cases} \Gamma_M(q)(a) = \gamma(q, a) \\ \Gamma_M(q)(a \cdot u) = \Gamma_M(\delta(q, a))(u) \qquad \text{for all } u \in A^+. \end{cases}$$

Having these two definitions will be helpful in proofs later in this document. We connect them with the following lemma.

Lemma 1. *For any $u \in A^*$ and $v \in A^+$, $\Gamma_M(\delta_M^*(u))(v) = \gamma_M^+(u \cdot v)$.*

Proof. To prove this lemma, we first show, by induction on the length of u, that

$$\Gamma_M(\delta_M^*(u))(v) = \Gamma_M(q_0)(u \cdot v). \tag{1}$$

If $u = \varepsilon$, we have

$$\begin{aligned} \Gamma_M(\delta_M^*(\varepsilon))(v) &= \Gamma_M(q_0)(v), && \text{by the definition of } \delta_M^*, \\ &= \Gamma_M(q_0)(\varepsilon \cdot v). \end{aligned}$$

For $u = w \cdot a$ with $w \in A^*$ and $a \in A$ we assume for all $v \in A^+$ that $\Gamma_M(\delta_M^*(w))(v) = \Gamma_M(q_0)(w \cdot v)$. Then

$$\begin{aligned} \Gamma_M(\delta_M^*(w \cdot a))(v) &= \Gamma_M(\delta(\delta_M^*(w), a))(v), && \text{by the definition of } \delta_M^*, \\ &= \Gamma_M(\delta_M^*(w))(a \cdot v), && \text{by the definition of } \Gamma_M, \\ &= \Gamma_M(q_0)(w \cdot a \cdot v), && \text{by the induction hypothesis.} \end{aligned}$$

This proves (1).

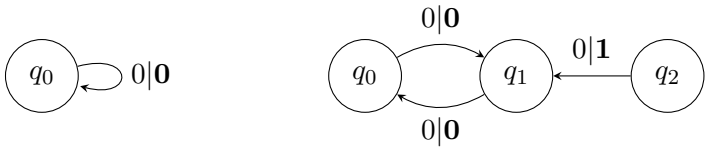
Now note that we can decompose $v = w \cdot a$ with $w \in A^*$ and $a \in A$, so

$$\begin{aligned} \Gamma_M(\delta_M^*(u))(w \cdot a) &= \Gamma_M(q_0)(u \cdot w \cdot a), && \text{by (1)}, \\ &= \Gamma_M(\delta_M^*(u \cdot w))(a), && \text{by (1)}, \\ &= \gamma(\delta_M^*(u \cdot w), a), && \text{by the definition of } \Gamma_M, \\ &= \gamma_M^+(u \cdot w \cdot a), && \text{by the definition of } \gamma_M^+. \end{aligned}$$

This concludes the proof of Lemma 1. □

3.1 Minimality

Mealy machines are not unique for the functions they realize. For example, the following two Mealy machines, with $A = \{0\}$ and $B = \{\mathbf{0}, \mathbf{1}\}$, both assign the output $\mathbf{0}$ to every input.



A first step towards learning a realization of a given rational function is to decide which Mealy machine this should be. An obvious choice is a machine having the desirable property of a minimal number of states. We will verify here that these Mealy machines are unique and later that they cover all rational functions.

Based on the work of Kalman [17] we identify two sources of redundancy for an automaton. The first consists of states that cannot be reached from

the initial state. These states can be removed without changing the semantics of the automaton. The second source of redundancy consists of states that cannot be distinguished from each other in terms of their observable behavior. Such states could thus be merged. We formalize and combine these notions in the following definition, composed of definitions similar to those given by Arbib and Manes [6], after which we will verify that it provides the properties that we require.

Definition 3 (Minimal Mealy machines)**.** A Mealy machine M is called *reachable* if all of its states can be reached, i.e., if δ_M^* is surjective. M is called *observable* if all pairs of different states can be distinguished by some continuation. That is, Γ_M has to be injective. M is said to be *minimal* if it is reachable and observable.

Theorem 1. *A minimal Mealy machine M has a minimal number of states. Up to isomorphism, M is the unique minimal realization of γ_M^+.*

Proof. Let $M = (Q, q_0, \delta, \gamma)$. Suppose there exists another Mealy machine $M' = (Q', q_0', \delta', \gamma')$ with $\gamma_{M'}^+ = \gamma_M^+$ and $|Q'| < |Q|$. Since M is observable, there must be some state that can be distinguished from all states of M', i.e., there must be some $q \in Q$ such that for all $q' \in Q'$, $\Gamma_M(q) \neq \Gamma_{M'}(q')$. By the reachability of M and Lemma 1 this contradicts our assumption that $\gamma_{M'}^+ = \gamma_M^+$. Thus M has to be state-minimal.

Adjust M' such that M' is also minimal and $|Q| = |Q'|$. We will demonstrate the existence of an isomorphism between M and M'.

Define $\phi : Q \to Q'$ by $\phi(\delta_M^*(u)) = \delta_{M'}^*(u)$ for any $u \in A^*$, exploiting the reachability of M. To see that ϕ is well-defined, note that, for any $u \in A^*$ and $v \in A^+$,

$$\begin{aligned}
\Gamma_M(\delta_M^*(u))(v) &= \gamma_M^+(u \cdot v), && \text{by Lemma 1,} \\
&= \gamma_{M'}^+(u \cdot v) \\
&= \Gamma_{M'}(\delta_{M'}^*(u))(v), && \text{by Lemma 1.}
\end{aligned}$$

It follows directly that $\Gamma_M(\delta_M^*(u)) = \Gamma_{M'}(\delta_{M'}^*(u))$ for any $u \in A^*$. For all $u, u' \in A^*$ we then have

$$\Gamma_M(\delta_M^*(u)) = \Gamma_M(\delta_M^*(u')) \iff \Gamma_{M'}(\delta_{M'}^*(u)) = \Gamma_{M'}(\delta_{M'}^*(u')).$$

Since M' is observable, it follows that ϕ is well-defined. Furthermore, with the observability of M we conclude that ϕ is one-to-one. Then it must also be onto, since $|Q| = |Q'|$.

It remains to show that ϕ is a Mealy machine homomorphism, i.e., that it preserves the initial state, the transition function, and the output function. For the initial state we have

$$\begin{aligned}
\phi(q_0) &= \phi(\delta_M^*(\varepsilon)), && \text{by the definition of } \delta_M^*, \\
&= \delta_{M'}^*(\varepsilon), && \text{by the definition of } \phi, \\
&= q_0', && \text{by the definition of } \delta_{M'}^*.
\end{aligned}$$

For the transition function we have, for any $u \in A^*$ and $a \in A$,

$$\begin{aligned}
\phi(\delta(\delta^*_M(u), a)) &= \phi(\delta^*_M(u \cdot a)), && \text{by the definition of } \delta^*_M, \\
&= \delta^*_{M'}(u \cdot a), && \text{by the definition of } \phi, \\
&= \delta'(\delta^*_{M'}(u), a), && \text{by the definition of } \delta^*_{M'}, \\
&= \delta'(\phi(\delta^*_M(u)), a), && \text{by the definition of } \phi.
\end{aligned}$$

Finally, for the output function it holds that

$$\begin{aligned}
\gamma(\delta^*_M(u), a) &= \gamma^+_M(u \cdot a), && \text{by the definition of } \gamma^+_M, \\
&= \gamma^+_{M'}(u \cdot a) \\
&= \gamma'(\delta^*_{M'}(u), a), && \text{by the definition of } \gamma^+_{M'}, \\
&= \gamma'(\phi(\delta^*_M(u)), a), && \text{by the definition of } \phi.
\end{aligned}$$

Thus M is the unique (up to isomorphism) minimal realization of γ^+_M. □

Intuitively, it should be clear that any rational function has a minimal realization. The main objective of the next section is to arrive at a constructive proof for this.

3.2 Regularity

Angluin's algorithm [5] works by iteratively refining a representation of the equivalence classes of the right congruence used by the Myhill-Nerode theorem [22]. In order to apply the algorithm to functions that can be represented by Mealy machines, we will present an adaptation of this congruence for Mealy machines and then show that it can be used to construct a minimal Mealy machine for any rational function. This provides the basis for a similar construction that needs to be performed by the algorithm. The following right congruence is based on the one given by Steffen et al. [29].

Definition 4. Two strings $u, u' \in A^*$ are equal with respect to some function $\chi : A^+ \to B$ if and only if χ maps them to the same output for all continuations:

$$u \equiv_\chi u' \iff \forall v \in A^+, \chi(u \cdot v) = \chi(u' \cdot v).$$

Let $[u]_\chi$ denote the equivalence class of u with respect to $\equiv_\chi$. We write $|\chi|$ for the index of $\equiv_\chi$.

Note the correspondence with the definition of minimality: the equivalence classes cover all input strings (reachability), and are distinguished by differences in the outputs for the continuations of those input strings (observability). It should follow that a Mealy machine having these equivalence classes as its states will be minimal, if it actually realizes the underlying function. This machine is also defined by Steffen et al. [29].

Definition 5. The Mealy machine $M_\chi = (Q, q_0, \delta, \gamma)$, based on some function $\chi : A^+ \to B$, where $|\chi|$ is finite, is defined by

- $Q = \{[u]_\chi \mid u \in A^*\}$,
- $q_0 = [\varepsilon]_\chi$,
- $\delta([u]_\chi, a) = [u \cdot a]_\chi$ for all $u \in A^*$ and $a \in A$, and
- $\gamma([u]_\chi, a) = \chi(u \cdot a)$ for all $u \in A^*$ and $a \in A$.

To see that this is a well-defined Mealy machine, note that obviously $\varepsilon \in A^*$, so q_0 is well-defined. For all $u \in A^*$ and $a \in A$, $[u \cdot a]_\chi$ is a valid state, since $A^+ \subseteq A^*$. Furthermore, if $u' \in A^*$, then $[u]_\chi = [u']_\chi$ implies $[u \cdot a]_\chi = [u' \cdot a]_\chi$ by the definition of $\equiv_\chi$. Therefore, δ is well-defined. For the above case we also have $\chi(u \cdot a) = \chi(u' \cdot a)$, so γ is well-defined. The finite index of $\equiv_\chi$ guarantees that Q is a finite set.

Lemma 2. *Any input string leads M_χ to the state representing the corresponding equivalence class. That is, $\delta^*_{M_\chi}(u) = [u]_\chi$ for any $u \in A^*$.*

Proof. Let $M_\chi = (Q, q_0, \delta, \gamma)$. We prove this lemma by induction on the length of u. For $u = \varepsilon$ we have

$$\begin{aligned} \delta^*_{M_\chi}(\varepsilon) &= q_0, && \text{by the definition of } \delta^*_{M_\chi}, \\ &= [\varepsilon]_\chi, && \text{by the definition of } q_0. \end{aligned}$$

Assume that $\delta^*_{M_\chi}(v) = [v]_\chi$ for some $v \in A^*$. For $u = v \cdot a$ with $a \in A$ we then find

$$\begin{aligned} \delta^*_{M_\chi}(v \cdot a) &= \delta(\delta^*_{M_\chi}(v), a), && \text{by the definition of } \delta^*_{M_\chi}, \\ &= \delta([v]_\chi, a), && \text{by the induction hypothesis}, \\ &= [v \cdot a]_\chi, && \text{by the definition of } \delta. \end{aligned}$$

Hence $\delta^*_{M_\chi}(u) = [u]_\chi$ for all $u \in A^*$. □

Lemma 3. *M_χ is a realization of χ; that is, $\gamma^+_{M_\chi} = \chi$.*

Proof. We will have to show that $\gamma^+_{M_\chi}(u) = \chi(u)$ for any $u \in A^+$. Note that u can be decomposed such that $u = v \cdot a$ with $v \in A^*$ and $a \in A$. Using $M_\chi = (Q, q_0, \delta, \gamma)$, we have

$$\begin{aligned} \gamma^+_{M_\chi}(v \cdot a) &= \gamma(\delta^*_{M_\chi}(v), a), && \text{by the definition of } \gamma^+_{M_\chi}, \\ &= \gamma([v]_\chi, a), && \text{by Lemma 2}, \\ &= \chi(v \cdot a), && \text{by the definition of } \gamma. \end{aligned}$$

Thus M_χ is a realization of χ. □

Theorem 2. *Given a rational function $\chi : A^+ \to B$, the Mealy machine M_χ is the unique minimal realization of χ.*

Proof. First we must show that M_χ is well-defined by proving that $|\chi|$ is finite. Since χ is a rational function, there must be some Mealy machine $M = (Q, q_0, \delta, \gamma)$ with $\gamma^+_M = \chi$. We will show that $|\gamma^+_M|$ is bounded by $|Q|$. Suppose $|\gamma^+_M| > |Q|$. Then there are $u, u' \in A^*$ with $\delta^*_M(u) = \delta^*_M(u')$ and $\gamma^+_M(u \cdot v) \neq \gamma^+_M(u' \cdot v)$ for some $v \in A^+$. Using Lemma 1 we find that $\Gamma_M(\delta^*_M(u))(v) \neq \Gamma_M(\delta^*_M(u'))(v)$, which implies $\delta^*_M(u) \neq \delta^*_M(u')$, revealing a contradiction. Thus $|\gamma^+_M| \leq |Q|$ must hold. Since Q is a finite set it follows that $|\gamma^+_M| = |\chi|$ must be finite. Hence M_χ is well-defined.

Lemma 2 directly implies the reachability of M_χ. It remains to show that M_χ is observable. For any $u \in A^*$ and $v \in A^+$, we have

$$\begin{aligned}
\Gamma_{M_\chi}([u]_\chi)(v) &= \Gamma_{M_\chi}(\delta^*_{M_\chi}(u))(v), && \text{by Lemma 2,} \\
&= \gamma^+_{M_\chi}(u \cdot v), && \text{by Lemma 1,} \\
&= \chi(u \cdot v), && \text{by Lemma 3.}
\end{aligned}$$

Together with the definition of $\equiv_\chi$ this proves the observability of M_χ. It follows from Theorem 1 and Lemma 3 that M_χ is the unique minimal realization of χ. □

4 Angluin's Algorithm for Mealy Machines

Angluin's algorithm [5] learns an automaton realizing the observable behavior of a target system by assuming the ability to ask two types of queries. The first query determines the output of the target machine on a specific input string. For regular languages this is called a membership query, but for rational functions we will rename it to an *output query*, which we denote by $\mathrm{OUT}(u)$ for any $u \in A^+$. Since its description corresponds to that of a rational function, we will treat $\mathrm{OUT} : A^+ \to B$ as such. The other query is an *equivalence query*, denoted by $\mathrm{EQ}(M)$. It determines whether a given Mealy machine M is a realization of OUT. If so, it will return *yes*. Otherwise it yields a *counterexample* $z \in A^+$ such that $\gamma^+_M(z) \neq \mathrm{OUT}(z)$.

The algorithm keeps track of the following data structure, which was introduced for regular languages by Angluin [5]. Our version is equivalent to the one given by Steffen et al. [29].

Definition 6. An *observation table* is a tuple (S, E, T), where

- $S \subseteq A^*$ is a set of *access strings*,
- $E \subseteq A^+$ is a set of *distinguishing suffixes*, and
- $T : (S \cup S \cdot A) \to B^E$ records known outputs of the target machine. That is, $T(s)(e) = \mathrm{OUT}(s \cdot e)$ for any $s \in (S \cup S \cdot A)$ and $e \in E$.

The observation table can be visualized as a table that has the strings from $(S \cup S \cdot A)$ as its row labels, the distinguishing suffixes as its column labels, and the values of T in the corresponding cells. By the upper part of the observation table we refer to the rows labelled by the access strings, while the other rows are called the lower part.

For example, if $A = \{0, 1\}$ and $B = \{\mathbf{0}, \mathbf{1}\}$, the observation table with $S = \{\varepsilon, 0, 01\}$, $E = \{1, 11\}$, and, for any $u \in A^+$, $\mathrm{OUT}(u) = \mathbf{0}$ if $|u| \leq 2$ and $\mathrm{OUT}(u) = \mathbf{1}$ otherwise would be given as follows.

		1	11
S	ε	**0**	**0**
	0	**0**	**1**
	01	**1**	**1**
$S \cdot A \setminus S$	1	**0**	**1**
	00	**1**	**1**
	010	**1**	**1**
	011	**1**	**1**

Intuitively, the upper part of T is an approximation of the equivalence classes of $\equiv_{\mathrm{OUT}}$, where S contains prefixes that represent those classes and E provides suffixes to distinguish them. This approximation is refined, by augmenting S and E, until it completely spans the actual equivalence classes.

The algorithm will repeatedly turn the observation table into a Mealy machine. Naturally, given the interpretation of T, this machine resembles the Mealy machine M_{OUT}. The initial state of M_{OUT} is defined as $[\varepsilon]_{\mathrm{OUT}}$, so here we will use $T(\varepsilon)$ and therefore initialize $S = \{\varepsilon\}$. With just one access string there is nothing to distinguish yet, so we set $E = \varnothing$.

Although our initialization of E naturally follows from the interpretation of T as an approximation of the right congruence, existing descriptions of the algorithm initialize $E = \{\varepsilon\}$ for regular languages and $E = A$ for rational functions. The idea behind this is that, as will soon become apparent, it allows the automaton to be completely specified from just the observation table. However, Steffen et al. [29] note that this extended initialization, for Mealy machines, adds significantly more data to the observation table than required. This would impair the output query complexity of the algorithm, and therefore they suggest to initialize $E = \varnothing$, although they do not incorporate this into their own algorithm.

Now recall that the well-definedness of the transition function of M_{OUT} depends on the fact that for any $u \in A^*$ and $a \in A$, $[u \cdot a]_{\mathrm{OUT}}$ is a valid equivalence class, given that $[u]_{\mathrm{OUT}}$ is one. For the observation table this translates into the following definition, which was introduced by Angluin [5] for regular languages and by Pena and Oliveira [25] for rational functions.

Definition 7 (Closed table)**.** The observation table is called *closed* if for each $s \in S \cdot A$ there exists an access string $s' \in S$ with $T(s) = T(s')$.

Furthermore, for any $u, u' \in A^*$, if $[u]_{\mathrm{OUT}} = [u']_{\mathrm{OUT}}$ then we must have $[u \cdot a]_{\mathrm{OUT}} = [u' \cdot a]_{\mathrm{OUT}}$ and for the well-definedness of the output function also $\mathrm{OUT}(u \cdot a) = \mathrm{OUT}(u' \cdot a)$. In analogy to this, we obtain the following properties for the observation table, which were first expressed, more abstractly, by Berg et al. [9].

Definition 8 (Consistent table)**.** The observation table is called *transition-consistent* if whenever $s, s' \in S$ are such that $T(s) = T(s')$, we also have $T(s \cdot a) = T(s' \cdot a)$ for all $a \in A$. The observation table is *output-consistent* if for any $s, s' \in S$ with $T(s) = T(s')$ we also have $\mathrm{OUT}(s \cdot a) = \mathrm{OUT}(s' \cdot a)$ for all $a \in A$. A *consistent* observation table is both transition-consistent and output-consistent.

Note that algorithms that use the extended initialization of E do not need this additional notion of output-consistency, since the definition of T implies that it is then automatically satisfied by transition-consistency.

Our version of the algorithm initially always has a closed and consistent observation table. This means that we can start our main loop by constructing a hypothesis automaton. Except for the output function, the following definition corresponds to the definition given by Angluin [5]. Our output function is assigned through output queries to accomodate the initially empty E.

Definition 9 (Mealy machine associated with a closed and consistent table)**.** The hypothesis Mealy machine $H = (Q, q_0, \delta, \gamma)$ is constructed from the observation table using

- $Q = \{T(s) \mid s \in S\}$,
- $q_0 = T(\varepsilon)$,
- $\delta(T(s), a) = T(s \cdot a)$ for all $s \in S$ and $a \in A$, and
- $\gamma(T(s), a) = \mathrm{OUT}(s \cdot a)$ for all $s \in S$ and $a \in A$.

To see that this is a well-defined Mealy machine, note that S initially and thus always contains ε, so q_0 is well-defined. Since the observation table is closed and transition-consistent, δ is also well-defined. Finally, γ is well-defined considering that the observation table is output-consistent.

After constructing this hypothesis, the algorithm queries its equivalence with the target system. If $\mathrm{EQ}(H)$ says *yes*, the algorithm will terminate returning the hypothesis. Otherwise we will have $\mathrm{EQ}(H) = z$ such that $\gamma_H^+(z) \neq \mathrm{OUT}(z)$. Angluin processes this counterexample by adding z and all of its prefixes to S.

Processing the counterexample may cause defects in the closedness or consistency of the observation table. We must fix these in order to be able to construct a new hypothesis. If the observation table is not closed, Angluin's

Algorithm 1: Angluin's algorithm [5] for Mealy machines.

$S \leftarrow \{\varepsilon\}, E \leftarrow \varnothing$
repeat
  construct H
  $z \leftarrow \mathrm{EQ}(H)$
  if $z \neq yes$ **then**
    $S \leftarrow S \cup \{u \in A^* \mid u \cdot v = z, v \in A^*\}$
    while (S, E, T) is not closed or not consistent **do**
      if (S, E, T) is not closed **then**
        find $s \in S \cdot A$ with $T(s) \neq T(s')$ for all $s' \in S$
        $S \leftarrow S \cup \{s\}$
      if (S, E, T) is not transition-consistent **then**
        find $s, s' \in S$, $a \in A$ and $e \in E$ with
          $T(s) = T(s')$ and $T(s \cdot a)(e) \neq T(s' \cdot a)(e)$
        $E \leftarrow E \cup \{a \cdot e\}$
      if (S, E, T) is not output-consistent **then**
        find $s, s' \in S$ and $a \in A$ with
          $T(s) = T(s')$ and $\mathrm{OUT}(s \cdot a) \neq \mathrm{OUT}(s' \cdot a)$
        $E \leftarrow E \cup \{a\}$
until $z = yes$
return H

algorithm will find $s \in S \cdot A$ such that for all $s' \in S$, $T(s) \neq T(s')$. It will then add this s to S. When the observation table is not transition-consistent, the algorithm will find $s, s' \in S$, $a \in A$ and $e \in E$ such that $T(s) = T(s')$, but $T(s \cdot a)(e) \neq T(s' \cdot a)(e)$. In this case it will add $a \cdot e$ to E, such that $T(s)$ will be distinguished from $T(s')$. If the observation table is not output-consistent, we can find $s, s' \in S$ and $a \in A$ such that $T(s) = T(s')$, but $\mathrm{OUT}(s \cdot a) \neq \mathrm{OUT}(s' \cdot a)$. Then we add a to E, also distinguishing $T(s)$ from $T(s')$. After preparing the observation table, the algorithm restarts its main loop by constructing a new hypothesis. This continues until counterexamples cease to arise.

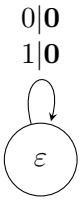
Pseudocode is shown in Algorithm 1. We demonstrate the algorithm with the following example.

Example 1. For $A = \{0, 1\}$ and $B = \{\mathbf{0}, \mathbf{1}, \mathbf{2}\}$, consider the target function given, for any $u \in A^+$, by

$$\mathrm{OUT}(u) = \begin{cases} \mathbf{2} & \text{if } u = 11 \\ \mathbf{1} & \text{if 01 is a suffix of } u \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

To construct the initial hypothesis, note that there is only one state, $T(\varepsilon)$, and that $\gamma(T(\varepsilon), a) = \mathrm{OUT}(a) = \mathbf{0}$ for any $a \in A$. Thus we arrive at

the Mealy machine depicted below. We label each node of the hypothesis diagram by one of the shortest access strings such that the corresponding state is the row of that access string.



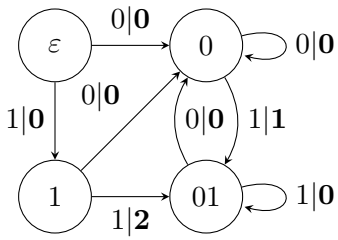
An equivalence query tells us that $\gamma_H^+(01) \neq \mathrm{OUT}(01)$, so we add 0 and 01 to S. Note that there are no distinguishing suffixes yet, and that therefore the table is still trivially closed and transition-consistent. However, it is not output-consistent. More specifically, we find that $T(\varepsilon) = T(0)$, whereas $\mathrm{OUT}(\varepsilon \cdot 1) = \mathbf{0} \neq \mathbf{1} = \mathrm{OUT}(0 \cdot 1)$, so we set $E \leftarrow E \cup \{1\}$. The resulting table is shown below on the left.

	1
ε	**0**
0	**1**
01	**0**
1	**2**
00	**1**
010	**1**
011	**0**

	1
ε	**0**
0	**1**
01	**0**
1	**2**
00	**1**
010	**1**
011	**0**
10	**1**
11	**0**

Now the observation table is not closed: there is no $s \in S$ with $T(s) = T(1)$. Thus we assign $S \leftarrow S \cup \{1\}$. The next table, shown above on the right, is closed, but not transition-consistent. We find that $T(\varepsilon) = T(01)$, but $T(\varepsilon \cdot 1)(1) \neq T(01 \cdot 1)(1)$. Then we set $E \leftarrow E \cup \{1 \cdot 1\}$. The resulting table is given below.

	1	11
ε	**0**	**2**
0	**1**	**0**
01	**0**	**0**
1	**2**	**0**
00	**1**	**0**
010	**1**	**0**
011	**0**	**0**
10	**1**	**0**
11	**0**	**0**



The observation table is now closed and consistent, so we conjecture the corresponding hypothesis (illustrated above) and find $\mathrm{EQ}(H) = \mathit{yes}$.

4.1 Minimality of the Hypotheses

Before discussing the correctness of the algorithm, we will first explore the minimality of its hypotheses. In terms of the observation table, reachability can be achieved by requiring that the prefixes of all access strings are also access strings. This guarantees that a row that does not represent the initial state is transitioned to from a row labelled by an access string that is one symbol smaller, eventually leading back to the row representing the initial state. For observability we can require all non-empty suffixes of the distinguishing suffixes to be distinguishing suffixes as well. This will make sure that transition pairs preserve the inequality of two states, unless the transition pair caused that inequality. Following Angluin [5], we arrive at the following formal definition.

Definition 10. We call S *prefix-closed* if $u \cdot a \in S$ implies $u \in S$ for all $u \in A^*$ and $a \in A$. Similarly, E is said to be *suffix-closed* if $a \cdot v \in E$ implies $v \in E$ for any $a \in A$ and $v \in A^+$.

Note that the loop that fixes the closedness and consistency of the observation table respects these properties. This also holds for the counterexample processing method of Angluin. However, we will later discuss a variation on the algorithm that does not keep E suffix-closed, so hereafter we will only implicitly assume the prefix-closedness of S. According to our informal reasoning above, reachability should now always be satisfied. The following proof is a simple adaptation of the proofs given by Angluin [5, Lemma 2] and Niese [23, Lemma 8.14].

Lemma 4. *Every hypothesis is reachable. More specifically,* $T(s) = \delta_H^*(s)$ *for every* $s \in S$.

Proof. We prove this lemma by induction on the length of s, which is possible because of the prefix-closedness of S. If $s = \varepsilon$, we have

$$\begin{aligned} T(\varepsilon) &= q_0, && \text{by the definition of } q_0, \\ &= \delta_H^*(\varepsilon), && \text{by the definition of } \delta_H^*. \end{aligned}$$

For $s = t \cdot a$ with $t \in S$ and $a \in A$ we assume $\delta_H^*(t) = T(t)$. Then

$$\begin{aligned} T(t \cdot a) &= \delta(T(t), a), && \text{by the definition of } \delta, \\ &= \delta(\delta_H^*(t), a), && \text{by the induction hypothesis,} \\ &= \delta_H^*(t \cdot a), && \text{by the definition of } \delta_H^*. \end{aligned}$$

Thus all states of H can be reached. □

Since the states of the hypothesis are determined by the distinct rows of the observation table, which represent the equivalence classes of our right congruence, the hypothesis should be observable if it is consistent with its observation table. We define and prove this formally.

Definition 11. The hypothesis is said to be *consistent with its observation table* if for any $s \in S$ and $e \in E$, we have $\gamma_H^+(s \cdot e) = T(s)(e)$.

Lemma 5. *Every hypothesis consistent with its observation table is minimal.*

Proof. Assume H is consistent with its observation table. Then for any $s \in S$ and $e \in E$,

$$\begin{aligned} \Gamma_H(T(s))(e) &= \Gamma_H(\delta_H^*(s))(e), && \text{by Lemma 4,} \\ &= \gamma_H^+(s \cdot e), && \text{by Lemma 1,} \\ &= T(s)(e), && \text{by Definition 11.} \end{aligned}$$

Since e ranges over the domain of the function $T(s) : E \to B$, it follows directly that Γ_H is injective. This proves the observability of H, which is also reachable as a result of Lemma 4. Hence the hypothesis is minimal if it is consistent with its observation table. □

Next we will show that the suffix-closedness maintained by Angluin translates to consistency with the observation table, such that all her hypotheses are minimal. This is also proven by Angluin [5, Lemma 3] and Niese [23, Lemma 8.15].

Lemma 6. *All hypotheses are minimal if E is kept suffix-closed.*

Proof. As shown above, we are done if we can prove $\Gamma_H(T(s))(e) = T(s)(e)$ for any $s \in S$ and $e \in E$. This can be achieved by induction on the length of e because of the suffix-closedness of E. If $e = a \in A$, we have

$$\begin{aligned} \Gamma_H(T(s))(a) &= \gamma(T(s), a), && \text{by the definition of } \Gamma_H, \\ &= \mathrm{OUT}(s \cdot a), && \text{by the definition of } \gamma, \\ &= T(s)(a), && \text{by the definition of } T. \end{aligned}$$

For $e = a \cdot e'$ with $a \in A$ and $e' \in E$, we assume $\Gamma_H(T(s))(e') = T(s)(e')$ for any $s \in S$. Let $s' \in S$ be such that $T(s') = T(s \cdot a)$, which is possible because the observation table is closed. Then

$$\begin{aligned} \Gamma_H(T(s))(a \cdot e') &= \Gamma_H(\delta(T(s), a))(e'), && \text{by the definition of } \Gamma_H, \\ &= \Gamma_H(T(s \cdot a))(e'), && \text{by the definition of } \delta, \\ &= \Gamma_H(T(s'))(e') \\ &= T(s')(e'), && \text{by the induction hypothesis,} \\ &= T(s \cdot a)(e') \\ &= \mathrm{OUT}(s \cdot a \cdot e'), && \text{by the definition of } T, \\ &= T(s)(a \cdot e'), && \text{by the definition of } T. \end{aligned}$$

Thus H is consistent with its observation table if E is suffix-closed. □

4.2 Correctness and Termination

The correctness of the algorithm follows directly from its termination. Similar to the reasoning given by Angluin [5] and Niese [23], we will limit the progress the algorithm can make after which we show that it terminates whenever this limit is reached. This progress amounts to an increase in the number of distinct rows of (the upper part of) the observation table. Note that in this sense negative progress is not possible, as access strings and distinguishing suffixes are never removed.

Lemma 7. *The number of distinct rows of the observation table can never exceed* $|\mathrm{OUT}|$.

Proof. For any $s, s' \in S$ with $T(s) \neq T(s')$ there is some $e \in E$ such that $T(s)(e) \neq T(s')(e)$, which implies $\mathrm{OUT}(s \cdot e) \neq \mathrm{OUT}(s' \cdot e)$ by the definition of T. Thus $[s]_{\mathrm{OUT}} \neq [s']_{\mathrm{OUT}}$. Then the number of distinct rows of the observation table can never exceed $|\mathrm{OUT}|$. □

Next we prove that every iteration of the loop that enforces the closedness and consistency of the observation table makes progress, which shows that this loop will terminate.

Lemma 8. *Processing a defect in either the closedness or consistency of the observation table directly results in an increase in the number of distinct rows of the observation table.*

Proof. If the observation table is not closed then there will be a row in the lower part of the table which differs from all rows in the upper part. This row is added to the upper part of the table, so the number of distinct rows increases.

If the observation table is not consistent, then a suffix will be added that distinguishes two equal rows in the observation table, thus also increasing the number of distinct rows. □

Regarding the termination of the main loop of the algorithm we will now prove that the specific method of processing counterexamples used by Angluin also makes progress.

Lemma 9. *Processing a counterexample will result in an increase in the number of distinct rows of the observation table.*

Proof. We can decompose the counterexample $z = s \cdot a$ with $s \in A^*$ and $a \in A$. Adding all the prefixes of z to S will also add s to S. Let (S', E', T') be the observation table of the next hypothesis $H' = (Q', q'_0, \delta', \gamma')$. Then

$$\begin{aligned}
\gamma^+_{H'}(s \cdot a) &= \Gamma_{H'}(\delta^*_{H'}(s))(a), && \text{by Lemma 1,} \\
&= \gamma'(\delta^*_{H'}(s), a), && \text{by the definition of } \Gamma_{H'}, \\
&= \gamma'(T'(s), a), && \text{by Lemma 4,} \\
&= \mathrm{OUT}(s \cdot a), && \text{by the definition of } \gamma'.
\end{aligned}$$

In other words, the counterexample will now be classified correctly, revealing that $\gamma^+_{H'} \neq \gamma^+_H$.

From Lemma 6 we know that H and H' are minimal, and we also know that H' must have at least as many states as H. Suppose H and H' have equally many states. Then $Q' = \{T'(s) \mid s \in S\}$ because $S \subseteq S'$ and $E \subseteq E'$. We will show that this would entail $\gamma^+_H = \gamma^+_{H'}$ by proving H and H' isomorphic. Define $\phi : Q \to Q'$ by $\phi(T(s)) = T'(s)$ for any $s \in S$. This must be well-defined, as $E \subseteq E'$ implies that if two rows would have become unequal, nothing could have compensated this in order to maintain $|Q| = |Q'|$. That ϕ is injective also follows directly from the fact that $E \subseteq E'$. Then ϕ is also surjective, since $|Q| = |Q'|$.

It remains to show that ϕ is a Mealy machine homomorphism. The preservation of the initial state is immediate:

$$\begin{aligned}
\phi(q_0) &= \phi(T(\varepsilon)), && \text{by the definition of } q_0,\\
&= T'(\varepsilon), && \text{by the definition of } \phi,\\
&= q'_0, && \text{by the definition of } q'_0.
\end{aligned}$$

Regarding the transition function, for any $s \in S$ and $a \in A$, the closedness of the observation table lets us find $s' \in S$ such that $T(s \cdot a) = T(s')$. Similarly, we can find $s'' \in S$ such that $T'(s \cdot a) = T'(s'')$. For all $e \in E$ we then have

$$\begin{aligned}
T(s')(e) &= T(s \cdot a)(e)\\
&= \mathrm{OUT}(s \cdot a \cdot e), && \text{by the definition of } T,\\
&= T'(s \cdot a)(e), && \text{by the definition of } T',\\
&= T'(s'')(e)\\
&= \mathrm{OUT}(s'' \cdot e), && \text{by the definition of } T',\\
&= T(s'')(e), && \text{by the definition of } T.
\end{aligned}$$

Hence $T(s') = T(s'')$, so using ϕ we find that $T'(s') = T'(s'') = T'(s \cdot a)$. Then

$$\begin{aligned}
\phi(\delta(T(s), a)) &= \phi(T(s \cdot a)), && \text{by the definition of } \delta,\\
&= \phi(T(s'))\\
&= T'(s'), && \text{by the definition of } \phi,\\
&= T'(s \cdot a), && \text{as shown above,}\\
&= \delta'(T'(s), a), && \text{by the definition of } \delta',\\
&= \delta'(\phi(T(s)), a), && \text{by the definition of } \phi.
\end{aligned}$$

Finally, for the output function we simply have

$$\begin{aligned}
\gamma(T(s), a) &= \mathrm{OUT}(s \cdot a), && \text{by the definition of } \gamma,\\
&= \gamma'(T'(s), a), && \text{by the definition of } \gamma',\\
&= \gamma'(\phi(T(s)), a), && \text{by the definition of } \phi.
\end{aligned}$$

This concludes the proof showing that H and H' are isomorphic, which contradicts our other result stating that $\gamma^+_{H'} \neq \gamma^+_H$. Thus $|Q| = |Q'|$ is not possible, and therefore the number of distinct rows of the observation table must have increased. □

Lemma 10. *The hypothesis is correct and minimal if the number of distinct rows of the observation table equals* |OUT|.

Proof. Assume the number of distinct rows equals |OUT| and suppose the hypothesis is not correct. Then there must be a counterexample such that by Lemma 9 the number of distinct rows of the observation table can be increased. However, Lemma 7 states that this is impossible. Therefore, the hypothesis must be correct.

Now suppose the hypothesis H is not consistent with its observation table. Then there would exist $s \in S$ and $e \in E$ such that

$$\mathrm{OUT}(s \cdot e) = \gamma^+_H(s \cdot e) \neq T(s)(e) = \mathrm{OUT}(s \cdot e).$$

Thus H is consistent with its observation table and by Lemma 5 therefore minimal. □

Theorem 3. *The algorithm will terminate and output a correct and minimal Mealy machine.*

Proof. Since OUT is a rational function, it follows from Theorem 2 that M_{OUT} is well-defined and thus that |OUT|, which by the definition of M_{OUT} equals the number of states of this machine, is finite. Because of Lemma 7 and Lemma 8, the loop that makes the observation table ready to be converted into a Mealy machine will terminate. Together with Lemma 9 this means that the main loop will terminate as well. More specifically, it will terminate when the number of distinct rows of the observation table equals |OUT|. Lemma 10 now shows that the algorithm will output a correct and minimal Mealy machine. □

Note that the only dependency on the specific method of Angluin [5] is made by Lemma 9, so if we prove this lemma for another method we will directly have proven Theorem 3 for it.

4.3 Reduced Observation Tables

Rivest and Schapire [26] introduced an optimization to Angluin's algorithm that improves its output query complexity by reducing the size of the observation table. They maintain the property that $T(s) \neq T(s')$ for all $s, s' \in S$ with $s \neq s'$. Note that this is initially satisfied and can never be impaired by the enforcement of closedness of the observation table. Furthermore, it automatically satisfies consistency so that these checks can be skipped.

Rivest and Schapire define the states of their hypotheses to be the access strings of the observation table. For our setting we will assume a function $\sigma : A^* \to S$ such that $\sigma(u)$ for any $u \in A^*$ is the unique $s \in S$ with $\delta_H^*(s) = \delta_H^*(u)$.

The main difference with the original algorithm lies in the processing of counterexamples. We will explain the method by proving that Lemma 9 holds for it. We do so by elaborating on the reasoning given by Rivest and Schapire [26] and Steffen et al. [29, Theorem 2].

Proof of Lemma 9. The key observation is that for a counterexample z there will be $u \in A^*$, $a \in A$ and $v \in A^+$ such that $z = u \cdot a \cdot v$ and

$$\mathrm{OUT}(\sigma(u) \cdot a \cdot v) \neq \mathrm{OUT}(\sigma(u \cdot a) \cdot v). \tag{2}$$

To see that this decomposition exists, choose u to be the largest prefix of z such that for some $v' \in A^*$ with $u \cdot v' = z$ we have $\mathrm{OUT}(\sigma(u) \cdot v') = \mathrm{OUT}(z)$. This must be possible, since it holds if we choose $u = \varepsilon$ and $v' = z$. It remains to show that $|v'| \geq 2$. If we choose $u = z$ and $v' = \varepsilon$, we find $\mathrm{OUT}(\sigma(u) \cdot v') = \mathrm{OUT}(\sigma(z)) \neq \mathrm{OUT}(z)$, as z is a counterexample. This contradicts the property $\mathrm{OUT}(\sigma(u) \cdot v') = \mathrm{OUT}(z)$.

Now suppose $v' = a' \in A$. Then

$$\begin{aligned}
\gamma_H^+(\sigma(u) \cdot a') &= \gamma(\delta_H^*(\sigma(u)), a'), && \text{by the definition of } \gamma_H^+,\\
&= \gamma(T(\sigma(u)), a'), && \text{by Lemma 4,}\\
&= \mathrm{OUT}(\sigma(u) \cdot a'), && \text{by the definition of } \gamma,\\
&= \mathrm{OUT}(z)\\
&\neq \gamma_H^+(z), && \text{as } z \text{ is a counterexample,}\\
&= \gamma_H^+(u \cdot a')\\
&= \Gamma_H(\delta_H^*(u))(a'), && \text{by Lemma 1,}\\
&= \Gamma_H(\delta_H^*(\sigma(u)))(a'), && \text{by the definition of } \sigma,\\
&= \gamma_H^+(\sigma(u) \cdot a'), && \text{by Lemma 1.}
\end{aligned}$$

Thus we can have $v' = a \cdot v$, and it follows that the decomposition constrained by (2) exists, using the maximization property of u for the inequality.

In general we have

$$\begin{aligned}
T(\sigma(u) \cdot a) &= \delta(T(\sigma(u)), a), && \text{by the definition of } \delta,\\
&= \delta(\delta_H^*(\sigma(u)), a), && \text{by Lemma 4,}\\
&= \delta(\delta_H^*(u), a), && \text{by the definition of } \sigma,\\
&= \delta_H^*(u \cdot a), && \text{by the definition of } \delta_H^*,\\
&= \delta_H^*(\sigma(u \cdot a)), && \text{by the definition of } \sigma,\\
&= T(\sigma(u \cdot a)), && \text{by Lemma 4.}
\end{aligned}$$

However, (2) forces $\sigma(u) \cdot a \neq \sigma(u \cdot a)$. It follows directly that adding v to E will distinguish $T(\sigma(u) \cdot a)$ from all $T(s)$ for $s \in S$, such that it will be added to the upper part of the observation table during the next enforcement of closedness thereof. □

Instead of adding the counterexample z and its prefixes to S, Rivest and Schapire use a binary search algorithm to find the decomposition given by (2) and then they simply add v to E. It follows that the algorithm will not violate the property $T(s) \neq T(s')$ for any $s, s' \in S$ with $s \neq s'$.

Algorithm 2: Counterexample processing by binary search [26].

Input: a counterexample z
Output: a suffix distinguishing two currently equal rows
Function `find_suffix(`z`)`
  $u \leftarrow \varepsilon, a \leftarrow z, v \leftarrow \varepsilon$
  while $|a| > 1$ **do**
    divide $a = x \cdot y$ such that $|x| - |y| \in \{0, 1\}$
    if $\mathrm{OUT}(\sigma(u \cdot x) \cdot y \cdot v) = \mathrm{OUT}(z)$ **then**
      $u \leftarrow u \cdot x$
      $a \leftarrow y$
    else
      $a \leftarrow x$
      $v \leftarrow y \cdot v$
  return v

Pseudocode for the binary search is shown in Algorithm 2. The function will maintain the properties $u \cdot a \cdot v = z$ and $\mathrm{OUT}(\sigma(u) \cdot a \cdot v) = \mathrm{OUT}(z)$ for each iteration. Appropriately moving half of a to either u or v every time, it will find a decomposition satisfying (2) when $|a| = 1$, since u will be maximized (although possibly only locally) as in the above proof. We present the full algorithm in Algorithm 3.

4.3.1 Minimality of Intermediate Hypotheses

Lee and Yannakakis [19] observed that the algorithm of Rivest and Schapire [26] may make conjectures for different automata having the same number of states. Related properties are that a hypothesis may not classify the previous counterexample correctly [7] and that intermediate hypotheses are not guaranteed to be minimal [29]. After all, the suffix-closedness of E is sacrificed. This is, however, not a necessary condition for minimality,[1] so we will illustrate this last issue by means of an example.

[1] For example, if OUT is a constant function, consider any observation table with $\varepsilon \in S$ that does not have a suffix-closed E. The corresponding hypothesis will always have one state. This same example shows that S does not have to be prefix-closed either.

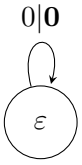
Algorithm 3: Rivest and Schapire's algorithm [26] for Mealy machines.

$S \leftarrow \{\varepsilon\}, E \leftarrow \varnothing$
repeat
  construct H
  $z \leftarrow \mathrm{EQ}(H)$
  if $z \neq yes$ **then**
    $E \leftarrow E \cup \{\texttt{find\_suffix}(z)\}$
    while (S, E, T) is not closed **do**
      find $s \in S \cdot A$ with $T(s) \neq T(s')$ for all $s' \in S$
      $S \leftarrow S \cup \{s\}$
until $z = yes$
return H

Example 2. Consider the input alphabet $A = \{0\}$, the output alphabet $B = \{\mathbf{0}, \mathbf{1}\}$, and the target function defined, for any $u \in A^+$, by

$$\mathrm{OUT}(u) = \begin{cases} \mathbf{1} & \text{if } u = 0000 \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

The only information needed to construct the initial hypothesis is the output for the only transition, which is given by $\gamma(T(\varepsilon), 0) = \mathrm{OUT}(\varepsilon \cdot 0) = \mathbf{0}$. The result is as follows.



An equivalence query tells us this is not the automaton we are looking for by providing us with the counterexample 0000. Following Algorithm 2, we initialize the execution of `find_suffix(0000)` by assigning $u \leftarrow \varepsilon$, $a \leftarrow 0000$, and $v \leftarrow \varepsilon$. Then we divide $a = x \cdot y$ by setting $x \leftarrow 00$ and $y \leftarrow 00$, and we find

$$\mathrm{OUT}(\sigma(\varepsilon \cdot 00) \cdot 00 \cdot \varepsilon) = \mathrm{OUT}(00) \neq \mathrm{OUT}(0000).$$

Therefore, we assign $a \leftarrow x = 00$ and $v \leftarrow y \cdot v = 00$. Now we divide a by setting $x \leftarrow 0$ and $y \leftarrow 0$. Then we have

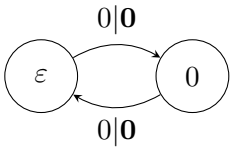
$$\mathrm{OUT}(\sigma(\varepsilon \cdot 0) \cdot 0 \cdot 00) = \mathrm{OUT}(000) \neq \mathrm{OUT}(0000).$$

Again we assign $a \leftarrow x = 0$ and $v \leftarrow y \cdot v = 0 \cdot 00 = 000$. Since $|a| = 1$, we conclude that the suffix $v = 000$ should be added to E. The resulting observation table, shown below on the left, is not closed, because we have $T(0) \neq T(\varepsilon)$, while $T(\varepsilon)$ is the only row in the upper part of the table. Thus

we add 0 to S, which results in the observation table shown below in the middle.

	000
ε	**0**
0	**1**

	000
ε	**0**
0	**1**
00	**0**



This last table is closed, so we construct the next hypothesis, which is illustrated above. The point is that this hypothesis is not observable. More specifically, we have $T(\varepsilon) \neq T(0)$, but $\Gamma_H(T(\varepsilon)) = \Gamma_H(T(0))$ because every transition generates the output **0**. This concludes the example.

Because of Theorem 3 we do not have to worry about the final hypothesis, but the loss of minimality of the intermediate hypotheses may give problems for specific implementations of equivalence queries that assume such a minimal model [e.g., 12].

A couple of authors have tried to overcome this problem. Shahbaz and Groz [27] presented a minor optimization to a method introduced by Maler and Pnueli [20], who processed a counterexample z by adding z and all of its suffixes to E. This will include the suffix found by Rivest and Schapire, which is a suffix of z. Obviously this keeps E suffix-closed, which guarantees minimal hypotheses as shown in Lemma 6, but it also increases the complexity of E.

Steffen et al. [29] were able to preserve the query complexity of the algorithm of Rivest and Schapire, by keeping E *semantically suffix-closed*. Translated to our setting, this means that for any two access strings $s, s' \in S$ and any decomposition $v_1 \cdot v_2 \in E$ with $T(s)(v_1 \cdot v_2) \neq T(s')(v_1 \cdot v_2)$, there is some $e' \in E$ such that $T(\sigma(s \cdot v_1))(e') \neq T(\sigma(s' \cdot v_1))(e')$. The intuition behind this is that the "duty" of v_2 to distinguish $T(\sigma(s \cdot v_1))$ from $T(\sigma(s' \cdot v_1))$ is delegated to e'. Their main theorem states that every hypothesis constructed from an observation table with semantically suffix-closed E is minimal. That this is incorrect follows from the fact that the delegations may be circular. For instance, it is possible that $e' = v_1 \cdot v_2$ and $\{\sigma(s \cdot v_1), \sigma(s' \cdot v_1)\} = \{s, s'\}$ for all of the decompositions.

This is the case in Example 2. The final observation table given by this example shows that we may consider only $s = \varepsilon$ and $s' = 0$ in order to have $T(s) \neq T(s')$. There is one suffix, so we must take $v_1 \cdot v_2 = e' = 000$. Table 1 enumerates the four values v_1 can take and shows that E is semantically suffix-closed. However, we concluded in Example 2 that the corresponding hypothesis is not minimal, which contradicts the theorem of Steffen et al. Note that we have chosen this example such that it carries over to their setting, which has the extended initialization of E.

Returning to the original method of Rivest and Schapire, one obvious solution is to minimize the hypotheses before making an equivalence query

v_1	$\sigma(s \cdot v_1)$	$\sigma(s' \cdot v_1)$	$T(\sigma(s \cdot v_1))(e')$	$T(\sigma(s' \cdot v_1))(e')$
ε	ε	0	**0**	**1**
0	0	ε	**1**	**0**
00	ε	0	**0**	**1**
000	0	ε	**1**	**0**

Table 1: Semantic suffix-closedness of the last hypothesis in Example 2.

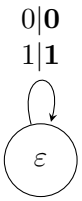
on them [29]. However, we will see that this may result in an unnecessary increase in the number of equivalence queries.

Balcázar et al. [7] note that the same counterexample can potentially be used to answer several equivalence queries. An interesting thought is that reusing a counterexample for as long as it is applicable might be sufficient to guarantee a minimal hypothesis when an equivalence query is finally required. We use an example to demonstrate that this is not the case.

Example 3. For $A = \{0, 1\}$ and $B = \{\mathbf{0}, \mathbf{1}, \mathbf{2}\}$, consider the target function given, for any $u \in A^+$, by

$$\mathrm{OUT}(u) = \begin{cases} \mathbf{2} & \text{if } u = v \cdot 1 \cdot w \cdot 1 \text{ for any } v, w \in A^* \\ \mathbf{1} & \text{if } u = 1 \text{ or } u = 01 \\ \mathbf{0} & \text{otherwise.} \end{cases}$$

Initially, we construct the following hypothesis.



An equivalence query yields the counterexample 101. Therefore, we start the execution of `find_suffix(101)` by setting $u \leftarrow \varepsilon$, $a \leftarrow 101$, and $v \leftarrow \varepsilon$. For the first iteration we divide $a = x \cdot y$ with $x \leftarrow 10$ and $y \leftarrow 1$. Then we have

$$\mathrm{OUT}(\sigma(\varepsilon \cdot 10) \cdot 1 \cdot \varepsilon) = \mathrm{OUT}(1) = \mathbf{1} \neq \mathbf{2} = \mathrm{OUT}(101).$$

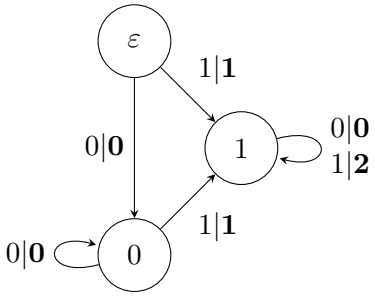
Thus we assign $a \leftarrow x = 10$ and $v \leftarrow y \cdot v = 1$. Next we decompose $a = x \cdot y$ by setting $x \leftarrow 1$ and $y \leftarrow 0$. Now

$$\mathrm{OUT}(\sigma(\varepsilon \cdot 1) \cdot 0 \cdot 1) = \mathrm{OUT}(01) = \mathbf{1} \neq \mathbf{2} = \mathrm{OUT}(101).$$

Hence we assign $a \leftarrow x = 1$ and $v \leftarrow y \cdot v = 01$. Since $|a| = 1$, we add 01 to E. The observation tables, before and after adding 0 and 1 to S to achieve closedness, are shown below. Next to them we present the resulting hypothesis.

	01
ε	**1**
0	**0**
1	**2**

	01
ε	**1**
0	**0**
1	**2**
00	**0**
01	**2**
10	**2**
11	**2**



One can easily verify that the counterexample is resolved, i.e., we now have $\gamma_H^+(101) = \mathbf{2} = \mathrm{OUT}(101)$. However, the transition sets going out of the states labelled by ε and 0 are equal in terms of their output symbols and target states. It follows that $\Gamma_H(T(\varepsilon)) = \Gamma_H(T(0))$, which concludes this example.

It turns out that these minimality defects are related to other counterexamples that can always be found in the observation table. Note that Lemma 5 entails that the hypothesis is not consistent with its observation table if it is not minimal. If this is the case, then there must be $s \in S$ and $e \in E$ such that

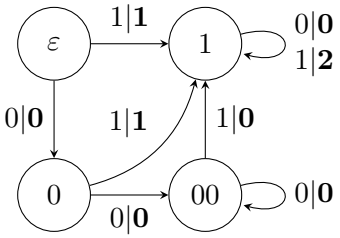
$$\gamma_H^+(s \cdot e) \neq T(s)(e) = \mathrm{OUT}(s \cdot e).$$

In other words, $s \cdot e$ is a counterexample. Hence we can adjust the algorithm to make equivalence queries only for minimal hypotheses, otherwise finding a counterexample in the observation table. This solution is actually a minor optimization, essentially similar to the one used by Balle [8]. Pseudocode for our final algorithm is shown in Algorithm 4.

Example 4. We continue Example 3 using this method. Note that the progress made so far is compatible with the new method because the first hypothesis was trivially consistent with its observation table. Instead of making an equivalence query, we find that the new hypothesis is not consistent with its observation table. More specifically, $\gamma_H^+(0 \cdot 01) \neq T(0)(01)$. Thus we process the counterexample 001, by adding `find_suffix(001)` $= 1$ to E. Below we have expanded and closed the observation table and constructed a hypothesis out of it.

	01	1
ε	**1**	**1**
0	**0**	**1**
1	**2**	**2**
00	**0**	**0**
01	**2**	**2**
10	**2**	**2**
11	**2**	**2**

	01	1
ε	**1**	**1**
0	**0**	**1**
1	**2**	**2**
00	**0**	**0**
01	**2**	**2**
10	**2**	**2**
11	**2**	**2**
000	**0**	**0**
001	**2**	**2**



Algorithm 4: "Safe" adaptation of the algorithm of Rivest and Schapire.

$S \leftarrow \{\varepsilon\}, E \leftarrow \varnothing$
repeat
  construct H
  if H is consistent with (S, E, T) **then**
    $z \leftarrow \mathrm{EQ}(H)$
  else
    find $s \in S$ and $e \in E$ with $\gamma_H^+(s \cdot e) \neq T(s)(e)$
    $z \leftarrow s \cdot e$
  if $z \neq yes$ **then**
    $E \leftarrow E \cup \{\texttt{find\_suffix}(z)\}$
    while (S, E, T) is not closed **do**
      find $s \in S \cdot A$ with $T(s) \neq T(s')$ for all $s' \in S$
      $S \leftarrow S \cup \{s\}$
until $z = yes$
return H

This hypothesis is consistent with its observation table, so we make an equivalence query and find $\mathrm{EQ}(H) = yes$.

4.4 Query Complexity

Variants of Angluin's algorithm are usually assessed by the complexity of their equivalence and output query usages, as these queries are considered significantly more expensive than regular computation steps. We will determine the worst case equivalence and output query complexities of our adaptations of the algorithms given by Angluin [5], Maler and Pnueli [20], and Rivest and Schapire [26]. The other algorithms are variants of those that make no significant changes, as far as our setting is concerned. Let m be the length of the longest counterexample and define $n = |\mathrm{OUT}|$ and $k = |A|$.

We have already seen that the main loop is executed at most n times, so the equivalence query complexity is $O(n)$ for all considered algorithms. For the output query complexity we assume that the results of these queries, at least the ones covered by the observation table, are cached. That is, such queries will never be made twice for the same input string. This is realistic, considering the expensiveness of the queries. A typical implementation actually stores T as a table, which is extended immediately when S or E is augmented.

In general, the total output complexity of the algorithms is bounded by the size of the final observation table and the additional queries made during the processing of counterexamples. Let S and E be the sets of access

strings and distinguishing suffixes when the algorithm terminates. The final observation table has $|S| * (k+1)$ rows and $|E|$ columns. If p represents the output queries that are made during the processing of a single counterexample and for which the result is not stored in the observation table, then the total output query complexity is given by $O(|S| * k * |E| + np)$.

Note that $|S|$ and $|E|$ will always have to be at least $O(n)$, since $|S| \geq n$ and in the worst case n suffixes are required to distinguish the different rows. Therefore, the number of queries needed for the enforcement of output-consistency and the construction of the output functions is insignificant, being in $O(n * |S| * k)$ because the hypothesis preparation loop is executed no more than n times in total as a result of Lemma 8 and Lemma 9.

For each algorithm we must express $|S|$, $|E|$ and p in n, m and k. Consider Angluin's method first. Adding the counterexample and all of its prefixes to S puts the complexity of $|S|$ at $O(nm)$. E will only be augmented when a distinction can be made, so $|E| \leq n$. The counterexample processing method does not make any output queries for which the result is not stored in the observation table, so p is in $O(1)$. Hence the output query complexity of Angluin's algorithm, adapted to our setting, is $O(n^2mk)$. Balcázar et al. [7] arrive at the same result for the algorithm for regular languages.

For the method of Maler and Pnueli [20] we also find an output query complexity of $O(n^2mk)$, after swapping $|S|$ and $|E|$. Note that Maler and Pnueli did not intend to improve on the complexity of Angluin, but merely introduced their change in order to eliminate the notion of consistency. However, Shahbaz and Groz [27] did find an improved complexity for Mealy machines when comparing their derivative of this method to Angluin's method. This improvement is a result of their initialization of $E = A$. Their method is invariant to this difference in initialization, but compared to our setting they worsened the complexity of Angluin's method.

Rivest and Schapire [26] keep the size of the observation table minimal. Their counterexample processing method adds one distinguishing suffix to E, and an improvement of the closedness of the observation table consists in adding one string to S. Using the results of Section 4.2, we find that $|S|$ and $|E|$ are in this case both at most n. The binary search on the counterexample has p in $O(\log m)$, so, in our setting, their output query complexity is $O(n^2k + n \log m)$. Rivest and Schapire also obtain this result for their algorithm for regular languages.

5 Discussion

We have presented a reconstruction of the algorithms by Angluin [5] and Rivest and Schapire [26] adapted to Mealy machines, along with a full formal correctness proof. Furthermore, we have shown that equivalence queries are not required for non-minimal hypotheses, since a counterexample can be

found in the observation table whenever the hypothesis is not minimal. In particular, this overcomes problems induced by the counterexample processing method of Rivest and Schapire [26]. Our solution enables the direct use of standard conformance testing methods for equivalence query approximations while retaining the efficiency of the algorithm.

Mealy machine adaptations of Angluin's algorithm have appeared many times before. Most of them are very brief, such as the description given by Pena and Oliveira [25]. More details were provided by Shahbaz and Groz [27], but this thesis is influenced most significantly by the work of Steffen et al. [29], who elaborated on the underlying theory and presented an adaptation of the optimized algorithm of Rivest and Schapire. However, these are all missing formal proofs. Adaptations with formal proofs are given for automata that assign output strings to transitions by Niese [23] and for automata that allow output strings both on states and on transitions by Vilar [30], but these authors consider only the counterexample processing method of Angluin. Our proofs explicitly handle the methods of both Angluin and Rivest and Schapire.

We did not consider the algorithm presented by Kearns and Vazirani [18]. Note that the approximated right congruence can more efficiently be represented by a tree that has the distinguishing suffixes as its nodes and the access strings as its leafs. This is what Kearns and Vazirani propose. A string is classified by *sifting* it through the tree, where taking the right branch from a node means that the concatenation of the string with the suffix of that node is accepted by the target language and taking the left branch means that it is not. It might be interesting to adapt this algorithm formally to Mealy machines and compare the results with our findings. Balcázar et al. [7], who provide a unified view on the algorithms for regular languages, claim that this algorithm can be optimized such that its query complexity matches that of the algorithm by Rivest and Schapire.

Our reconstruction of Angluin's algorithm in Section 4 is strongly connected to the theory due to Nerode [22] presented in Section 3.2. In the case of Mealy machines this may not be very exciting, but this approach has been used to learn more complicated types of automata [11, 15], and it may inspire even more interesting future results.

Instead of using the right congruence, the minimal automaton can equivalently be constructed from the *total response* of the target machine [6], which in our setting is given by $\Gamma_M \circ \delta_M^*$ for a Mealy machine M. From this point of view, Lemma 1 actually expresses a property that is essential for the applicability of the algorithm: the output query function, being a rational function, completely determines the total response of the target machine. This approach is also closer to the more abstract reformulation of Jacobs and Silva [16]. Although lacking a correctness proof, they demonstrate that the algorithm can be applied to automata in other categories. Further research in this direction may lead to many more elegant generalizations.

References

[1] Fides Aarts, Bengt Jonsson, and Johan Uijen. Generating models of infinite-state communication protocols using regular inference with abstraction. In *Testing Software and Systems*, pages 188–204. Springer, 2010.

[2] Fides Aarts, Julien Schmaltz, and Frits Vaandrager. Inference and abstraction of the biometric passport. In *Leveraging Applications of Formal Methods, Verification, and Validation*, pages 673–686. Springer, 2010.

[3] Fides Aarts, Joeri de Ruiter, and Erik Poll. Formal models of bank cards for free. In *Proceedings of the 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation Workshops*, pages 461–468. IEEE Computer Society, 2013.

[4] Glenn Ammons, Rastislav Bodík, and James R. Larus. Mining specifications. In *ACM Sigplan Notices*, volume 37, pages 4–16. ACM, 2002.

[5] Dana Angluin. Learning regular sets from queries and counterexamples. *Information and computation*, 75(2):87–106, 1987.

[6] Michael A. Arbib and Ernest G. Manes. Adjoint machines, state-behavior machines, and duality. *Journal of Pure and Applied Algebra*, 6(3):313–344, 1975.

[7] José L. Balcázar, Josep Díaz, Ricard Gavaldà, and Osamu Watanabe. Algorithms for learning finite automata from queries: A unified view. *Advances in Algorithms, Languages, and Complexity*, page 53, 1997.

[8] Borja Balle. Implementing Kearns-Vazirani algorithm for learning DFA only with membership queries. In *ZULU workshop organised during ICGI*, pages 12–19, 2010.

[9] Therese Berg, Olga Grinchtein, Bengt Jonsson, Martin Leucker, Harald Raffelt, and Bernhard Steffen. On the correspondence between conformance testing and regular inference. In *Fundamental Approaches to Software Engineering*, pages 175–189. Springer, 2005.

[10] Therese Berg, Bengt Jonsson, and Harald Raffelt. Regular inference for state machines using domains with equality tests. In *Fundamental Approaches to Software Engineering*, pages 317–331. Springer, 2008.

[11] Benedikt Bollig, Peter Habermehl, Carsten Kern, and Martin Leucker. Angluin-style learning of NFA. In *Proceedings of the 21st international jont conference on Artifical intelligence*, pages 1004–1009. Morgan Kaufmann Publishers Inc., 2009.

[12] Tsun S. Chow. Testing software design modeled by finite-state machines. *IEEE Trans. Software Eng.*, 4(3):178–187, 1978.

[13] Edmund M. Clarke, Jr., Orna Grumberg, and Doron A. Peled. *Model Checking*. MIT press, 1999.

[14] Alex Groce, Doron Peled, and Mihalis Yannakakis. Adaptive model checking. In *Tools and Algorithms for the Construction and Analysis of Systems*, pages 357–370. Springer, 2002.

[15] Falk Howar, Bernhard Steffen, Bengt Jonsson, and Sofia Cassel. Inferring canonical register automata. In *Verification, Model Checking, and Abstract Interpretation*, pages 251–266. Springer, 2012.

[16] Bart Jacobs and Alexandra Silva. Automata learning: A categorical perspective. In *Horizons of the Mind. A Tribute to Prakash Panangaden*, volume 8464 of *Lecture Notes in Computer Science*, pages 384–406. Springer, 2014.

[17] R. E. Kalman. On the general theory of control systems. *IRE Transactions on Automatic Control*, 4(3):110–110, 1959.

[18] Michael J. Kearns and Umesh V. Vazirani. *An Introduction to Computational Learning Theory*. MIT press, 1994.

[19] David Lee and Mihalis Yannakakis. Principles and methods of testing finite state machines—a survey. *Proceedings of the IEEE*, 84(8):1090–1123, 1996.

[20] Oded Maler and Amir Pnueli. On the learnability of infinitary regular sets. *Information and Computation*, 118(2):316–326, 1995.

[21] George H. Mealy. A method for synthesizing sequential circuits. *Bell System Technical Journal*, 34(5):1045–1079, 1955.

[22] A. Nerode. Linear automaton transformations. *Proceedings of the American Mathematical Society*, 9(4):541–544, 1958.

[23] Oliver Niese. *An Integrated Approach to Testing Complex Systems*. PhD thesis, Universität Dortmund, 2003.

[24] Doron Peled, Moshe Y. Vardi, and Mihalis Yannakakis. Black box checking. In *Formal Methods for Protocol Engineering and Distributed Systems*, pages 225–240. Springer, 1999.

[25] Jorge M. Pena and Arlindo L. Oliveira. A new algorithm for the reduction of incompletely specified finite state machines. In *Proceedings of the 1998 IEEE/ACM international conference on Computer-aided design*, pages 482–489. ACM, 1998.

[26] Ronald L. Rivest and Robert E. Schapire. Inference of finite automata using homing sequences. *Information and Computation*, 103(2):299–347, 1993.

[27] Muzammil Shahbaz and Roland Groz. Inferring Mealy machines. In *FM 2009: Formal Methods*, pages 207–222. Springer, 2009.

[28] Guoqiang Shu and David Lee. Testing security properties of protocol implementations – a machine learning based approach. In *Proceedings of the 27th International Conference on Distributed Computing Systems*, page 25. IEEE Computer Society, 2007.

[29] Bernhard Steffen, Falk Howar, and Maik Merten. Introduction to active automata learning from a practical perspective. In *Formal Methods for Eternal Networked Software Systems*, pages 256–296. Springer, 2011.

[30] Juan Miguel Vilar. Query learning of subsequential transducers. In *Grammatical Interference: Learning Syntax from Sentences*, pages 72–83. Springer, 1996.