Angluin-Style Learning of NFA

Benedikt Bollig, Peter Habermehl,
Carsten Kern, and Martin Leucker

Research Report LSV-08-28

October 2008

Angluin-Style Learning of NFA[⋆]

Benedikt Bollig[1], Peter Habermehl[1], Carsten Kern[2], and Martin Leucker[3]

[1] LSV, CNRS UMR 8643 & ENS de Cachan, France
[2] Software Modeling and Verification Group, RWTH Aachen University, Germany
[3] Institut für Informatik, TU München, Germany

Abstract. This paper introduces NL*, a learning algorithm for inferring non-deterministic finite-state automata using membership and equivalence queries. More specifically, residual finite-state automata (RFSA) are learned similar as in Angluin's popular L* algorithm, which however learns deterministic finite-state automata (DFA). As RFSA can be exponentially more succinct than DFA, RFSA are the preferable choice for many learning applications. The implementation of our algorithms is applied to a collection of examples and confirms the expected advantage of NL* over L*.

1 Introduction

In recent years, learning techniques have become popular especially in the area of automatic verification. They have been used for *minimizing* (partially) specified systems [17] and for model checking *black-box systems* [19, 10]. They proved extremely helpful in deriving assumptions in *compositional model checking* [5, 2], have been employed in *learning interfaces* [1], and for conformance testing of boolean programs [12]. Learning algorithms are at the heart of several practical approaches to *regular model checking* [11, 21, 22] and to the automatic verification of *networks of processes* [9]. To put it bluntly, automata learning is en vogue [14].

The general goal of learning algorithms employed in verification is to identify a *machine*, usually of *minimal* size, that *conforms* with an *a priori fixed* set of strings or a given machine. Nearly all algorithms learn *deterministic finite automata* (DFA) or deterministic finite-state machines (Mealy-/Moore machines), as the class of DFA has preferable properties in the setting of learning. For every regular language, there is a unique minimal DFA accepting it, which moreover can be characterized thanks to Nerode's right congruence. It is this characterization which builds the foundations of most learning algorithms.

In general, two types of learning algorithms for DFA can be distinguished, so-called *online* and *offline* algorithms. Offline algorithms get a fixed set of *positive* and *negative* examples, comprising strings that should be *accepted* and, respectively, strings that should be *rejected* by the automaton in question. The learning algorithm now has to provide a (minimal) automaton that accepts the positive examples and rejects the negative ones. For deriving minimal automata, major achievements are due to Biermann [4], Trakhtenbrot and Barzdin [20]. Efficient algorithms inferring a *not necessarily minimal* DFA are presented in [13] or in [18] under the name *RPNI*.

Online algorithms have the possibility to ask further *queries*, whether some string is in the language of the automaton to learn or not. In this way, an online algorithm can enlarge the set of examples as needed.

A popular setup for an online approach is that of Angluin's L* algorithm [3]: a minimal DFA is learned based on *membership* and *equivalence queries*. Using a pictorial language, we have a *learner* whose job is to come up with the automaton to learn, a *teacher* who may answer whether a given string is in the language as well as an *oracle* answering whether the automaton hypothesis currently proposed by the learner is correct or not.

Nevertheless, DFA have a serious drawback, especially for applications in verification. In general, a DFA might be exponentially bigger than a *non-deterministic finite-state automaton* (NFA). For many applications, it would be a tremendous improvement to work with an exponentially more

[⋆] Work partially supported by DAAD/ÉGIDE (Procope 2008) within the project *Synthesis of Design Models from Scenarios by Learning*

succinct NFA rather than the corresponding DFA. As such, learning algorithms for NFA are of need. However, the class of NFA lacks important properties that are essential for current learning algorithms: There is no unique minimal NFA for a given regular language, say it is not clear which automaton to learn, and, there is no characterization of NFA in terms of right-congruence classes.

In a seminal paper, Denis et al. [6] introduce the class of *residual finite-state automata* (RFSA). It is a sub-class of NFA, which shares important properties with the class of DFA: For every regular language, there is a unique minimal canonical RFSA accepting it. The states of this automaton correspond to right-congruence classes, or, equivalently, to *residuals* of the accepted language. At the same time, the RFSA can be exponentially more succinct than the corresponding DFA. As such, RFSA are the preferable class for learning regular languages. Denis et al. provided in [7] an offline algorithm, called DeLeTe2, which works in the spirit of RPNI. Alternatives and extensions to this algorithms have then been presented, most recently in [8], which also gives a nice overview on offline algorithms for learning NFA.

In this paper, we introduce NL* as the first online learning algorithm for RFSA, patterned after Angluin's L* algorithm. Thus, using membership and equivalence queries, our algorithm infers a (minimal) canonical RFSA for the language in question, which is always smaller than or equal to the corresponding DFA.

Moreover, we show that we need at most $O(n^2)$ equivalence queries and roughly mn^3 membership queries (more than L*) if the number of states of the minimal deterministic automaton for the language to be learned is n and the size of the biggest counterexample given by the equivalence oracle is m. We have implemented our algorithm and studied its efficiency on a collection of regular languages described by regular expressions. It turns out that for most examples, the resulting RFSA is much smaller than the corresponding DFA. Perhaps even more important, the resulting RFSA is typically obtained after far less membership and equivalence queries than L* in contrast to our theoretical bound. Especially equivalence queries are often quite expensive in practice, especially in the context of conformance testing. Altogether, this paper provides a new learning algorithm for regular languages which is expected to enhance associated verification tasks considerably.

2 Preliminaries

For the rest of the paper, we fix a finite set Σ of letters, called *alphabet*. Finite sequences of letters are elements of Σ^* and are called *words*. Sets of words are termed *languages* and are thus subsets of Σ^*. For a word $w \in \Sigma^*$ we denote by $\mathit{Pref}(w)$ (resp. $\mathit{Suff}(w)$) the set of all its prefixes (resp. suffixes) including w itself and the empty word ϵ. We denote the powerset of set X by 2^X.

In the following subsections we introduce the basic automata models and the learning algorithm we are going to employ and extend, respectively.

2.1 Finite-State Automata

A prominent concept for representing regular languages are finite-state automata. Typically, one distinguishes *deterministic* and *non-deterministic* versions thereof.

Definition 1 (Finite-State Automaton). *A* non-deterministic finite-state automaton *(NFA)* $\mathcal{A} = (Q, Q_0, F, \delta)$ *consists of a finite set of* states Q*, a set of* initial states $Q_0 \subseteq Q$*, a set of* final states $F \subseteq Q$*, and a* transition function $\delta : Q \times \Sigma \to 2^Q$.

We call $\mathcal{A}$ a *deterministic finite-state automaton* (DFA) if $|Q_0| = 1$ and $|\delta(q, a)| = 1$ for all $q \in Q$ and $a \in \Sigma$. The transition function δ of an NFA is extended as usual to $\bar{\delta} : Q \times \Sigma^* \to 2^Q$ by $\bar{\delta}(q, \epsilon) = \{q\}$ and $\bar{\delta}(q, aw) = \bigcup_{q' \in \delta(q,a)} \bar{\delta}(q', w)$, and subsequently to sets of states $Q' \subseteq Q$ by $\hat{\delta}(Q', w) = \bigcup_{q \in Q'} \bar{\delta}(q, w)$. To simplify notation, we use δ to denote both $\bar{\delta}$ and $\hat{\delta}$.

Given $\mathcal{A}$, the *language* of a state $q \in Q$, denoted by L_q, is the set of words $w \in \Sigma^*$ such that $\delta(q, w) \cap F \neq \emptyset$. The *language* $L(\mathcal{A})$ *accepted* by $\mathcal{A}$ is the union of languages of its initial states: $L(\mathcal{A}) = \bigcup_{q \in Q_0} L_q$. Two automata are called *equivalent* if they accept the same language.

It is folklore that the languages of finite-state automata are the *regular* ones.

An NFA is called *minimal* if there is no equivalent NFA with strictly less states. Likewise, we call a DFA *minimal* if there is no equivalent DFA with strictly less states. In contrast to NFA, a DFA has always a unique minimal representative:

Theorem 1 (Myhill-Nerode). *For each regular language L, there exists a unique (up to isomorphism) minimal DFA $\mathcal{A}$ with $L(\mathcal{A}) = L$.*

2.2 Residual Finite-State Automata

Here we recall the notion of *residual finite-state automata* (RFSA) introduced and studied in the seminal work [6]. RFSA are a subclass of NFA inheriting some desirable features of DFA. Most important for learning, every regular language is accepted by a unique (canonical) RFSA with a minimal number of states. As this property does not hold for arbitrary NFA, it seems difficult to come up with learning algorithms for the whole class of NFA. At the same time, like for NFA, RFSA can be exponentially more succinct than DFA, making it the preferable automaton model to work with in practical learning applications.

Technically, RFSA and DFA have the property that the states of the automata correspond to so-called *residual languages* defined below. This is not true for all NFA.

Definition 2 (Residual Language). *For a language $L \subseteq \Sigma^*$ and $u \in \Sigma^*$, we denote by $u^{-1}L$ the set $\{v \in \Sigma^* \mid uv \in L\}$. A language $L' \subseteq \Sigma^*$ is a* residual language *of L if there is a $u \in \Sigma^*$ with $L' = u^{-1}L$. We denote by $Res(L)$ the set of residual languages of L.*

For simplicity, we also talk of a *residual* rather than of a residual language. The Myhill-Nerode theorem states that the number of residual languages of a language is finite iff this language is regular [16]. Moreover, for a minimal DFA $\mathcal{A}$, there is a natural bijection between its states and the residual languages of $L(\mathcal{A})$.

We are now prepared to introduce residual automata:

Definition 3 (Residual Finite-State Automaton). *A* residual finite-state automaton *(RFSA) over Σ is an NFA $\mathcal{R} = (Q, Q_0, F, \delta)$ such that for each $q \in Q$, $L_q \in Res(L(\mathcal{R}))$.*

In other words, each state accepts a residual language of $L(\mathcal{R})$, but not every residual language must be accepted by a *single* state. Intuitively, the states of an RFSA are a subset of the states of the corresponding minimal DFA. Yet, using non-determinism, certain states of a minimal DFA are not needed as they correspond to the union of languages of other states. To this end, we distinguish *prime* and *composed* residuals: A residual is called *composed*, if it is the non-trivial union of other residuals. Otherwise, it is called *prime*.

Definition 4 (Prime and Composed Residuals). *Let $L \subseteq \Sigma^*$ be a language. A residual $L' \in Res(L)$ is called* composed *if there are $L_1, \ldots, L_n \in Res(L) \setminus \{L'\}$ such that $L' = L_1 \cup \ldots \cup L_n$. Otherwise, it is called* prime*. The set of prime residuals of L is denoted by $Primes(L)$.*

We can now define the *canonical* RFSA of a regular language. The idea is that its set of states corresponds exactly to its prime residuals. Moreover, the transition function should be *saturated* in the sense that a transition to a state should exist if it does not change the accepted language. Formally, we define:

Definition 5 (Canonical RFSA). *Let L be a regular language. The canonical RFSA of L, denoted by $\mathcal{R}(L)$, is the tuple (Q, Q_0, F, δ) where $Q = Primes(L)$, $Q_0 = \{L' \in Q \mid L' \subseteq L\}$, $F = \{L' \in Q \mid \epsilon \in L'\}$, and $\delta(L_1, a) = \{L_2 \in Q \mid L_2 \subseteq a^{-1}L_1\}$.*

Note that the canonical RFSA of a regular language is well-defined as the set of prime residuals for a regular language is finite and, for each $a \in \Sigma$ and $L' \in Res(L)$, we have $a^{-1}L' \in Res(L)$. Moreover, we actually have $L(\mathcal{R}(L)) = L$. By definition, there is a single and thus unique RFSA for every regular language. We say that an RFSA $\mathcal{R}$ is *canonical* if it is the canonical RFSA of $L(\mathcal{R})$.

2.3 Angluin's Learning Algorithm L*

Angluin's algorithm L* [3] learns or infers a minimal DFA for a given regular language. In the algorithm, a so-called *Learner*, who initially knows nothing about the given language L, is trying to learn a DFA $\mathcal{A}$ such that $L(\mathcal{A}) = L$. To this end, it asks repeatedly queries to a *Teacher* and an *Oracle*, who both know L. There are two kinds of queries (cf. Figure 1):

– A *membership query* consists in asking the *Teacher* if a string $w \in \Sigma^*$ is in L.
– An *equivalence query* consists in asking the *Oracle* whether a *hypothesized* DFA $\mathcal{H}$ is correct, i.e., whether $L(\mathcal{H}) = L$. The *Oracle* answers *yes* if $\mathcal{H}$ is correct, or supplies a counterexample w, drawn from the symmetric difference of L and $L(\mathcal{H})$.

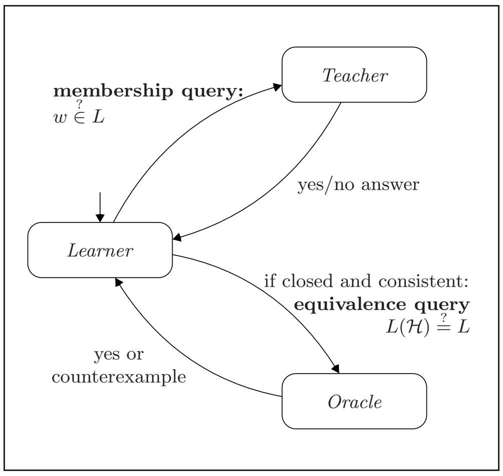

Fig. 1. Components of L* and their interaction

The *Learner* maintains a prefix-closed set $U \subseteq \Sigma^*$ of words that are candidates for identifying states, and a suffix-closed set $V \subseteq \Sigma^*$ of words that are used to distinguish such states. Words from U are usually called *access strings* and words from V *experiments*. The sets U and V are increased when needed. The *Learner* makes membership queries for all words in $(U \cup U\Sigma)V$, and organizes the results into a *table* $\mathcal{T} = (T, U, V)$ where function T maps each $w \in (U \cup U\Sigma)V$ to an element from $\{+, -\}$ where *parity* $+$ represents *accepted* and $-$ *not accepted*. To a string $u \in U \cup U\Sigma$, we assign a function $row(u) : V \to \{+, -\}$ given by $row(u)(v) = T(uv)$. Any such function is called a *row* of $\mathcal{T}$ and the set of all rows of a table is denoted by $Rows(\mathcal{T})$. We let $Rows_{\text{upp}}(\mathcal{T}) = \{row(u) \mid u \in U\}$ denote the set of rows that represent the "upper" part of the table. Likewise, the rows from $Rows_{\text{low}}(\mathcal{T}) = \{row(u) \mid u \in U\Sigma\}$ occur in its "lower" part. Table $\mathcal{T}$ is

– *closed*, if for all $u \in U$ and $a \in \Sigma$ there is $u' \in U$ such that $row(ua) = row(u')$, and
– *consistent*, if for all $u,u' \in U$ and $a \in \Sigma$, $row(u) = row(u')$ implies $row(ua) = row(u'a)$.

If $\mathcal{T}$ is not closed, we find $u' \in U\Sigma$ such that $row(u) \neq row(u')$ for all $u \in U$. We move u' to U and ask membership queries for every $u'av$ where $a \in \Sigma$ and $v \in V$. Likewise, if $\mathcal{T}$ is not consistent, we find $u, u' \in U$, $a \in \Sigma$, and $v \in V$ such that $row(u) = row(u')$ and $row(ua)(v) \neq row(u'a)(v)$. Then we add av to V and ask membership queries for every $u''av$ where $u'' \in U \cup U\Sigma$. When $\mathcal{T}$ is closed and consistent the *Learner* constructs a hypothesized DFA $\mathcal{H} = (Q, q_0, \delta, F)$, where $Q = \{row(u) \mid u \in U\} = Rows_{\text{upp}}(\mathcal{T})$, q_0 is the row $row(\epsilon)$, δ is defined by $\delta(row(u), a) = row(ua)$, and $F = \{r \in Q \mid r(\epsilon) = +\}$. Then, the *Learner* submits $\mathcal{H}$ to an equivalence query (asking whether $L(\mathcal{H}) = L$). If the answer is *yes*, the learning procedure is completed. Otherwise the returned counterexample u is processed by adding every prefix of u (including u) to U, extending

$U\Sigma$ accordingly, and subsequent membership queries are performed in order to make the table closed and consistent, after which a new hypothesized DFA is constructed, etc. (cf. Figure 1).

Remark 1. L* can be modified by changing the treatment of counterexamples. Instead of adding the counterexample and its prefixes to U one can add the counterexample and all its suffixes to V. This ensures that the table is *always* consistent [15].

3 Learning of Residual Finite-State Automata

Here we explain how to modify Angluin's learning algorithm L*, which infers DFA, towards learning of non-deterministic finite-state automata in terms of RFSA.

3.1 From Tables to RFSA

To simplify our presentation, we closely follow Angluin's notions and notation. We also use tables $\mathcal{T} = (T, U, V)$ with a prefix-closed set of words U, a suffix-closed set V, and a mapping $T : (U \cup U\Sigma)V \to \{+, -\}$. As above, we associate with any word $u \in U \cup U\Sigma$ a mapping $row(u) : V \to \{+, -\}$. Again, members of U are used to reach states and members of V to distinguish states. We adopt notations introduced before such as $Rows(\mathcal{T})$, $Rows_{\text{upp}}(\mathcal{T})$, and $Rows_{\text{low}}(\mathcal{T})$.

The main difference in the new approach is that *not all rows* of the table will correspond to states of the hypothesized RFSA, but only so-called *prime* rows. Essentially, we have to define for rows what corresponds to *union*, *composed*, *prime*, and *subset* previously introduced for languages.

Definition 6 (Join Operator). *Let $\mathcal{T} = (T, U, V)$ be a table. The* join $(r_1 \sqcup r_2) : V \to \{+, -\}$ *of two rows $r_1, r_2 \in Rows(\mathcal{T})$ is defined component-wise for each $v \in V$: $(r_1 \sqcup r_2)(v) = r_1(v) \sqcup r_2(v)$ where $- \sqcup - = -$ and $+ \sqcup + = + \sqcup - = - \sqcup + = +$.*

Note that the join operator is associative, commutative, and idempotent, yet that the join of two rows is *not* necessarily a row of table $\mathcal{T}$.

Definition 7 (Composed and Prime Rows). *Let $\mathcal{T} = (T, U, V)$ be a table. A row $r \in Rows(\mathcal{T})$ is called* composed *if there are rows $r_1, \ldots, r_n \in Rows(\mathcal{T}) \setminus \{r\}$ such that $r = r_1 \sqcup \ldots \sqcup r_n$. Otherwise, r is called* prime*. The set of prime rows in $\mathcal{T}$ is denoted by $Primes(\mathcal{T})$. Moreover, we let $Primes_{\text{upp}}(\mathcal{T}) = Primes(\mathcal{T}) \cap Rows_{\text{upp}}(\mathcal{T})$.*

Definition 8 (Covering Relation). *Let $\mathcal{T} = (T, U, V)$ be a table. A row $r \in Rows(\mathcal{T})$ is* covered *by row $r' \in Rows(\mathcal{T})$, denoted by $r \sqsubseteq r'$, if for all $v \in V$, $r(v) = +$ implies $r'(v) = +$. If moreover $r' \neq r$, then r is* strictly covered *by r', denoted by $r \sqsubset r'$.*

Note that r may be covered by r' and both r and r' are prime. A composed row covers all the primes it is composed of.

As in Angluin's learning algorithm, we now introduce concepts comparable to closedness and consistency and call them RFSA-closedness and RFSA-consistency.

For DFA, closedness ensures that every row in the lower part also occurs in the upper part. For RFSA, this translates to the idea that each row of the lower part of the table is composed of (prime) rows from the upper part. Formally:

Definition 9 (RFSA-Closedness). *A table $\mathcal{T} = (T, U, V)$ is called* RFSA-closed *if, for each $r \in Rows_{\text{low}}(\mathcal{T})$, $r = \bigsqcup\{r' \in Primes_{\text{upp}}(\mathcal{T}) \mid r' \sqsubseteq r\}$.*

Note that a table is RFSA-closed iff any prime row of the lower part is a prime row of the upper part of the table.

The idea of consistency in case of DFA is as follows: Assume that two words u and u' of the table have the same row. This suggests that both words lead to the same state of the automaton to learn as they cannot be distinguished by words $v \in V$. Hence, they induce the same residuals. Then, however, ua and $u'a$ have to induce equal residuals as well, for any $a \in \Sigma$. In other words, if there is some $a \in \Sigma$ and $v \in V$ such that $T(uav) \neq T(u'av)$, then the residuals induced by u and u' cannot be the same and must be distinguishable by the suffix av to be added to V.

For RFSA, if there are u and u' with $row(u) \sqsubseteq row(u')$, then this suggests that the residual induced by u is a subset of the residual induced by u'. If indeed so, then the same relation must hold for the successors ua and $u'a$. This is formally expressed as:

Definition 10 (RFSA-Consistency). *A table $\mathcal{T} = (T, U, V)$ is called* RFSA-consistent *if, for all $u, u' \in U$ and $a \in \Sigma$, $row(u') \sqsubseteq row(u)$ implies $row(u'a) \sqsubseteq row(ua)$.*

With a table that is RFSA-closed and RFSA-consistent, we can associate an NFA. Later we will show that this NFA corresponds to a canonical RFSA after our learning algorithm terminates.

Definition 11 (NFA of a Table). *For a table $\mathcal{T} = (T, U, V)$ that is RFSA-closed and RFSA-consistent, we define an NFA $\mathcal{R}_{\mathcal{T}} = (Q, Q_0, F, \delta)$ by*

- $Q = Primes_{\text{upp}}(\mathcal{T})$,
- $Q_0 = \{r \in Q \mid r \sqsubseteq row(\epsilon)\}$,
- $F = \{r \in Q \mid r(\epsilon) = +\}$, *and*
- $\delta(row(u), a) = \{r \in Q \mid r \sqsubseteq row(ua)\}$ *for $u \in U$ with $row(u) \in Q$ and $a \in \Sigma$.*

Note that $Primes_{\text{upp}}(\mathcal{T}) = Primes(\mathcal{T})$, as $\mathcal{T}$ is closed. Furthermore, $row(\epsilon)$ is not in Q_0 iff it is composed. Moreover, δ is well-defined: Take u, u' with $row(u) = row(u')$. Then, $row(u) \sqsubseteq row(u')$ and $row(u') \sqsubseteq row(u)$. Consistency implies that $row(ua) \sqsubseteq row(u'a)$ and $row(u'a) \sqsubseteq row(ua)$ so that both resulting rows are the same.

For the rest of this subsection, we fix a table $\mathcal{T} = (T, U, V)$ that is RFSA-closed and RFSA-consistent. We prove some important properties of the automaton $\mathcal{R}_{\mathcal{T}}$ constructed from the table.

Lemma 1. *Let $\mathcal{R}_{\mathcal{T}} = (Q, Q_0, F, \delta)$. For all $u \in U$ and $r \in \delta(Q_0, u)$, we have $r \sqsubseteq row(u)$.*

Proof: We prove this lemma by induction on the length of u. If $u = \epsilon$, then we have $\delta(Q_0, \epsilon) = Q_0$ and by definition of $\mathcal{R}_{\mathcal{T}}$ we have $\forall r \in Q_0 . r \sqsubseteq row(\epsilon)$. If $u = u'a$, then $\delta(Q_0, u'a) = \delta(\delta(Q_0, u'), a)$. Take an $r \in \delta(\delta(Q_0, u'), a)$. Because of the definition of δ we have that there exist $u'' \in U$ and $r' \in \delta(Q_0, u')$ with $r' = row(u'')$ and $r \sqsubseteq row(u''a)$. By induction hypothesis we have $r' \sqsubseteq row(u')$. Therefore $row(u'') \sqsubseteq row(u')$ and, by RFSA-consistency, $row(u''a) \sqsubseteq row(u'a)$. This implies $r \sqsubseteq row(u'a)$. □

The next lemma says that each *state* of $\mathcal{R}_{\mathcal{T}}$ correctly classifies strings of V.

Lemma 2. *Let $\mathcal{R}_{\mathcal{T}} = (Q, Q_0, F, \delta)$. For each $r \in Q$ and $v \in V$, the following hold:*

1. $r(v) = -$ *iff* $v \notin L_r$ *and*
2. $row(\epsilon)(v) = -$ *iff* $v \notin L(\mathcal{R}_{\mathcal{T}})$.

Proof: 1. We prove $\forall v \in V . (r(v) = -$ iff $v \notin L_r)$ by induction on the length of v.

Suppose $v = \epsilon$. Then, $r(\epsilon) = -$ iff $r \in F$ by definition of $\mathcal{R}_{\mathcal{T}}$ and therefore $r(\epsilon) = -$ iff $\epsilon \notin L_r$. Suppose now $v = av'$. Let $r = row(u)$ for some $r \in Q$ and $u \in U$.

"only if": As V is suffix-closed, $r(av') = row(u)(av') = -$ implies $row(ua)(v') = -$. Since the table is RFSA-closed, we have $row(ua) = \bigsqcup\{r' \in Q \mid r' \sqsubseteq row(ua)\}$. Thus by the definition of $\sqsubseteq$, we have that for all $r' \in Q$ with $r' \sqsubseteq row(ua)$, $r'(v') = -$ as well. Due to the induction hypothesis, this implies $v' \notin L_{r'}$ for all $r' \in Q$ with $r' \sqsubseteq row(ua)$. Thus, $av' \notin L_r$, as the states reached from r by a are exactly the $r' \in Q$ with $r' \sqsubseteq row(ua)$ by definition.

"if": Now let $r(av') = row(u)(av') = +$. This implies that $row(ua)(v') = +$ as V is suffix-closed. Since the table is RFSA-closed there exists $r' \in Q$ with $r' \sqsubseteq row(ua)$ and $r'(v') = +$. Then, by induction hypothesis, $v' \in L_{r'}$. Therefore, $av' \in L_r$, since by definition of the transition relation, r' can be reached from r by a.

2. Now, $\forall v \in V . (row(\epsilon)(v) = -$ iff $v \notin L(\mathcal{R}_{\mathcal{T}}))$ follows easily if $row(\epsilon) \in Q$. If not, then we have $row(\epsilon) = \bigsqcup\{r' \in Q \mid r' \sqsubseteq row(\epsilon)\} = Q_0$, and the "only if"-direction follows from the first part of the lemma applied on the r'. The if-direction follows from the fact that $v \notin L(\mathcal{R}_{\mathcal{T}})$ implies that for all $r' \in Q$ with $r' \sqsubseteq row(\epsilon)$ we have $v \notin L_{r'}$ and applying the first part of the lemma. □

The following lemma states an essential property for states in the covering relation and the languages accepted by these states:

Lemma 3. *Let $\mathcal{R}_\mathcal{T} = (Q, Q_0, F, \delta)$. For every $r_1, r_2 \in Q$, $r_1 \sqsubseteq r_2$ iff $L_{r_1} \subseteq L_{r_2}$.*

Proof: Let $r_1, r_2 \in Q$ and assume $u_1, u_2 \in U$ with $row(u_1) = r_1$ and $row(u_2) = r_2$.

"only if": Suppose that $r_1 \sqsubseteq r_2$ and $w \in L_{r_1}$. We distinguish two cases.

Assume first $w = \epsilon$. Then, $row(u_1)(\epsilon) = +$ and, due to $r_1 \sqsubseteq r_2$, $row(u_2)(\epsilon) = +$. Thus, $r_2 \in F$ so that $\epsilon \in L_{r_2}$. Now let $w = aw'$ with $a \in \Sigma$. We have $\delta(r_1, aw') \cap F \neq \emptyset$. Thus, there is $r \in \delta(r_1, a)$ such that $\delta(r, w') \cap F \neq \emptyset$. From $r_1 \sqsubseteq r_2$, we obtain, by RFSA-consistency, $row(u_1a) \sqsubseteq row(u_2a)$. By the definition of δ, $r \sqsubseteq row(u_1a)$, which implies $r \sqsubseteq row(u_2a)$. Thus, $r \in \delta(r_2, a)$ and we have $aw' \in L_{r_2}$.

"if": Assume $r_1 \not\sqsubseteq r_2$. We will show that $L_{r_1} \not\subseteq L_{r_2}$. By definition of the $\sqsubseteq$-relation, there has to be $v \in V$ such that $row(u_1)(v) = +$ but $row(u_2)(v) = -$. By Lemma 2, $v \in L_{r_1}$ and $v \notin L_{r_2}$. Therefore, $L_{r_1} \not\subseteq L_{r_2}$. □

The automaton $\mathcal{R}_\mathcal{T}$ constructed from the RFSA-closed and RFSA-consistent table $\mathcal{T}$ is not necessarily an RFSA (see Appendix E). But we can show that $\mathcal{R}_\mathcal{T}$ is a canonical RFSA *if it is consistent wrt. the table*, i.e., the automaton correctly classifies all words of $\mathcal{T}$.

Definition 12. *We say that $\mathcal{R}_\mathcal{T}$ is* consistent with the table $\mathcal{T}$ *if, for all $w \in (U \cup U\Sigma)V$, we have $T(w) = +$ iff $w \in L(\mathcal{R}_\mathcal{T})$.*

The next lemma is a stronger version of Lemma 1, if we have additionally that $\mathcal{R}_\mathcal{T}$ is consistent with $\mathcal{T}$.

Lemma 4. *Suppose $\mathcal{R}_\mathcal{T} = (Q, Q_0, F, \delta)$ is consistent with $\mathcal{T}$. Then, for all $u \in U$ with $row(u) \in Q$, we have $row(u) \in \delta(Q_0, u)$.*

Proof: If $row(u)(v) = -$ for all $v \in V$, this is easy to see, by the definition of δ. If not, we suppose that $row(u) \notin \delta(Q_0, u)$ and get a contradiction. With Lemma 1, we have $\forall r \in \delta(Q_0, u).r \sqsubseteq row(u)$. Then, Lemma 3 implies $\forall r \in \delta(Q_0, u).L_r \subseteq L_{row(u)}$. As $row(u) \in Q$ and $row(u) \notin \delta(Q_0, u)$ there exists $v \in V$ such that $row(u)(v) = +$ and for all $r \in \delta(Q_0, u)$, $r(v) = -$. This with Lemma 2 implies that for all $r \in \delta(Q_0, u).v \notin L_r$. But then $uv \notin L(\mathcal{R}_\mathcal{T})$ which is a contradiction to the fact that $\mathcal{R}_\mathcal{T}$ is consistent with $\mathcal{T}$. □

Theorem 2. *Let $\mathcal{T}$ be a table that is RFSA-closed and RFSA-consistent and let $\mathcal{R}_\mathcal{T}$ be consistent with $\mathcal{T}$. Then, $\mathcal{R}_\mathcal{T}$ is a canonical RFSA.*

Proof: Let $\mathcal{T} = (T, U, V)$ and assume $\mathcal{R}_\mathcal{T} = (Q, q_0, F, \delta)$. Let L denote $L(\mathcal{R}_\mathcal{T})$. We first show that $\mathcal{R}_\mathcal{T}$ is an RFSA (i.e., all states accept residuals). Let $u \in U$ with $row(u) \in Q$. We show that $L_{row(u)} = u^{-1}L$. Due to Lemma 4 we have $row(u) \in \delta(Q_0, u)$. This implies $L_{row(u)} \subseteq u^{-1}L$. We have furthermore with Lemma 1 that $\forall r \in \delta(Q_0, u).r \sqsubseteq row(u)$. This implies with Lemma 3 $\forall r \in \delta(Q_0, u).L_r \subseteq L_{row(u)}$. This gives $u^{-1}L \subseteq L_{row(u)}$. Together with $L_{row(u)} \subseteq u^{-1}L$ we have $L_{row(u)} = u^{-1}L$.

It remains to show that $L_{row(u)}$ is prime. But this is due to Lemma 3, which states that the relation $\sqsubseteq$ over rows corresponds exactly to the subset relation over languages. This precise correspondence is also the reason why the transition function δ is saturated, as required in the definition of a canonical RFSA. □

3.2 The Algorithm

We now describe NL*. Its pseudo code is given in Table 3.2. After initializing the table $\mathcal{T}$, the current table is repeatedly checked for RFSA-closedness and RFSA-consistency. If the algorithm detects a the violation of the RFSA-closedness condition (cf. Definition 9), i.e., some $row(ua)$ with $u \in U$ and $a \in \Sigma$ is prime and is not contained in $Primes_{\text{upp}}(\mathcal{T})$, then ua is added to U. This involves additional membership queries. On the other hand, whenever the algorithm perceives

```
NL$^*$($\Sigma$, DFA: $\mathcal{A}$):
1  initialize $\mathcal{T} = (T, U, V)$ by $U = V = \{\epsilon\}$ and $T(w)$ for all $w \in (U \cup U\Sigma)V$
2  repeat
3          while $\mathcal{T}$ is not (RFSA-closed and RFSA-consistent)
4              do
5                  if $\mathcal{T}$ is not RFSA-closed then
6                      find $u \in U$ and $a \in \Sigma$ such that $row(ua) \in Primes(\mathcal{T}) \setminus Primes_{\text{upp}}(\mathcal{T})$
7                      extend table to $\mathcal{T} := (T', U \cup \{ua\}, V)$ by membership queries
8                  if $\mathcal{T}$ is not RFSA-consistent then
9                      find $u \in U, a \in \Sigma$, and $v \in V$ such that the following holds:
10                         $T(uav) = -$ and
11                         $T(u'av) = +$ for some $u' \in U$ such that $row(u') \sqsubseteq row(u)$,
12                     extend table to $\mathcal{T} := (T', U, V \cup \{av\})$
13         /* $\mathcal{T}$ is both RFSA-closed and RFSA-consistent */
14         from $\mathcal{T}$, construct the hypothesized NFA $\mathcal{R}_{\mathcal{T}}$ (cf. Definition 11)
15         /* perform equivalence test */
16         if $(L(\mathcal{A}) = L(\mathcal{R}_{\mathcal{T}}))$
17            then  equivalence test succeeds
18            else  get counterexample $w \in (L(\mathcal{A}) \setminus L(\mathcal{R}_{\mathcal{T}})) \cup (L(\mathcal{R}_{\mathcal{T}}) \setminus L(\mathcal{A}))$
19                  extend table to $\mathcal{T} := (T', U, V \cup Suff(w))$ by membership queries
20 until equivalence test succeeds
21 return $\mathcal{R}_{\mathcal{T}}$
```

Table 1. NL*: the NFA version of Angluin's algorithm L*

an RFSA-consistency violation (cf. Definition 10) a suffix av can be determined which makes two existing rows distinct or incomparable. In this case, a column is added to V invoking supplemental queries. This procedure is repeated until $\mathcal{T}$ is RFSA-closed and RFSA-consistent. If both properties are fulfilled a conjecture $\mathcal{R}_{\mathcal{T}}$ can be derived from $\mathcal{T}$ (cf. Definition 11) and either a counterexample u from the symmetric difference of $L(\mathcal{A})$ and $L(\mathcal{R}_{\mathcal{T}})$ is provided and $Suff(u)$ added to V reinvoking NL* or the learning procedure terminates successfully. Notice that the algorithm makes sure that V is always suffix-closed and U prefix-closed.

Remark 2. We chose to treat the counterexamples as in the variant of L* described in Remark 1. Indeed, treating the counterexamples as in the original L* does not lead to a terminating algorithm (see Appendix F). The treatment of the counterexample ensures that each row can appear at most once in the upper part of the table, because we only add rows when the table is not RFSA-closed.

Proving the termination of Angluin's learning algorithm L* is quite straightforward. In our setting this is not so easy anymore since, as we show in Appendix D, after an equivalence query or a violation of RFSA-consistency the number of states of the hypothesized automaton does not necessarily increase (as it is the case for L*). To show termination of NL*, we need first a simple lemma.

Lemma 5. *If the minimal DFA $\mathcal{A}^*$ for a given regular language L has n states, then the tables constructed in the runs of* NL* *with input $\mathcal{A}^*$ cannot have more than n different rows.*

Proof: Having more than n different rows in a table implies that L has more than n different residuals which is impossible, as the minimal DFA $\mathcal{A}^*$ for L has n states. □

Now we are prepared to give the main contribution of the paper. For any given regular language L, the algorithm NL* infers the canonical RFSA $\mathcal{R}$ accepting L.

Theorem 3. *Let n be the number of states of the minimal DFA $\mathcal{A}^*$ for a given regular language $L \subseteq \Sigma^*$. Let m be the length of the biggest counterexample returned by the equivalence test (or* 1 *if the equivalence test always succeeds). Then,* NL* *returns after at most $O(n^2)$ equivalence queries and $O(m|\Sigma|n^3)$ membership queries the canonical RFSA $\mathcal{R}(L)$.*

Proof: First of all, if the algorithm terminates, then it outputs the canonical RFSA for L due to Theorem 2, because passing the equivalence test implies that the constructed automaton must be consistent with the table.

We show in the following that the algorithm terminates after at most $O(n^2)$ equivalence queries. We define first a measure M associating a tuple of positive natural numbers to tables. For a given table $\mathcal{T}$, let $M(\mathcal{T}) = (l_{up}, l, p, i)$, where $l_{up} = |\mathit{Rows}_{\mathrm{upp}}(\mathcal{T})|$ is the number of rows in the upper part of the table, $l = |\mathit{Rows}(\mathcal{T})|$ the number of different rows in the whole table, $p = |\mathit{Primes}(\mathcal{T})|$ the number of prime rows in the table, and i the number of strict coverings of pairs of different rows of the table, i.e., $i = |\{(r, r') \mid r, r' \in \mathit{Rows}(\mathcal{T}) \text{ and } r \sqsubset r'\}|$. It is crucial to consider rows and not members of U. Initially, $l_{up} = 1$ and ($l = 1$ or $l = 2$) and ($p = 1$ or $p = 2$) and ($i = 0$ or $i = 1$).

Let us examine how the measure (l_{up}, l, p, i) evolves during a run of NL*. An example of this evolution is given in Appendix G, where Table 13 depicts an NL* run and the measure associated with any of its tables. It is clear that l_{up} and l can never decrease since two different rows stay different by extending the table.

If the table is not RFSA-closed, then, after extending the table, l_{up} increases by 1. Simultaneously, l might increase by $k > 0$ (the number of new rows added). At the same time, i might increase by at most k times the old value of l (the largest possible number of strict covering relations between new rows and old rows) plus $k(k-1)/2$ (the largest possible number of strict covering relations between new rows).

If the table is not RFSA-consistent, then, after extending the table, l_{up} stays unchanged. However, l might increase by $k > 0$. At the same time, as before, i might increase by at most k times the old value of l plus $k(k-1)/2$. If l does not increase, then this means that, for two strings $u, u' \in U \cup U\Sigma$ with $\mathit{row}(u) = \mathit{row}(u')$ in the table before the extension, we have again $\mathit{row}(u) = \mathit{row}(u')$ after the extension. Therefore, no strict covering relation can be added in the extended table. But since we add a word to V making two rows r and r' in the original table with $r \sqsubset r'$ uncomparable, i is *decreased* by at least 1.

If the table is RFSA-closed and RFSA-consistent, then an equivalence query is performed. Let us fix an RFSA-closed and RFSA-consistent table $\mathcal{T} = (T, U, V)$ before the equivalence test. If the test fails, we obtain a counterexample w and a new table $\mathcal{T}' = (T', U, V \cup \mathit{Suff}(w))$. Notice that $\mathcal{T}$ must be extended (otherwise, we have $w \in V$, which implies with Lemma 2 that w is correctly classified by $\mathcal{R}_{\mathcal{T}}$). Either l increases or not. If l increases by $k > 0$, then, as before, i might increase by at most k times the old value of l plus $k(k-1)/2$. If l does not increase, then i cannot increase (see explanation for the case that the table is not RFSA-consistent). We will furthermore show that p increases or i decreases. Suppose that this is not the case, i.e., p and i remain unchanged. Then, the automata $\mathcal{R}_{\mathcal{T}}$ and $\mathcal{R}_{\mathcal{T}'}$ constructed from $\mathcal{T}$ and, respectively, $\mathcal{T}'$ must be the same: all primes of $\mathcal{T}$ must still be primes in $\mathcal{T}'$ (as p stays the same, no primes are added), the initial and final states stay the same, and the transition relation is defined using the covering relation which does not change. This is because l does not change and therefore no new strict covering relation can be added like for the corresponding case above, where the table is not RFSA-consistent. Furthermore, since i does not change, no strict covering relation is removed. But the two automata being the same is a contradiction since $\mathcal{R}_{\mathcal{T}'}$ classifies w correctly according to Lemma 2 (w is in V), whereas $\mathcal{R}_{\mathcal{T}}$ does not. Therefore, p increases or i decreases (notice that p might be decreased by other steps). Consider Table 13 in Appendix G. We have $M(\mathcal{T}_8) = (7, 7, 6, 9)$ and $M(\mathcal{T}_9) = (7, 7, 7, 9)$. I.e., after adding the counterexample, both l and i remain unchanged. However, p is indeed increased.

Putting the different cases together, we notice that after each extension of the table either (1) l_{up} is increased or (2) l is increased by $k > 0$ and simultaneously i is increased by at most $kl + k(k-1)/2$ or (3) l stays the same and we have that i decreases or p increases. Due to Lemma 5, l_{up}, l, and p cannot increase beyond n. Hence, the algorithm must (1) always reach an equivalence query and (2) terminate after at most $O(n^2)$ equivalence queries. Concerning the number of membership queries, we notice that their maximal number corresponds to the size of the table which has at most $n + n|\Sigma|$ (n rows in the upper part + their successors) rows and $O(mn^2)$ columns since at each extension at most m suffixes are added to V. □

Language L is == word with in the 3rd position from end :
exemple L = {aba, aaa, abb, bbabb, ...}
!L = {baa, bab, bbabbb...}

$\mathcal{T}_1$	ϵ
ϵ	$-$
a	$-$
b	$-$

T1 is closed & consist,
counter-ex : aaa

$\overset{1)}{\Rightarrow}$

$\mathcal{T}_2$	ϵ
ϵ	$-$
a	$-$
aa	$-$
aaa	$+$
b	$-$
ab	$-$
aab	$+$
$aaaa$	$+$
$aaab$	$+$

Not consistent :
row(a) == row(aa)
but tow(aa) != row(aaa)

$\overset{2)}{\Rightarrow}$

$\mathcal{T}_3$	ϵ	a
ϵ	$-$	$-$
a	$-$	$-$
aa	$-$	$+$
aaa	$+$	$+$
b	$-$	$-$
ab	$-$	$+$
aab	$+$	$+$
$aaaa$	$+$	$+$
$aaab$	$+$	$+$

Not consistent :
row(e) == row(a)
but tow(a) != row(aa)

$\overset{3)}{\Rightarrow}$

$\mathcal{T}_4$	ϵ	a	ba
ϵ	$-$	$-$	$-$
a	$-$	$-$	$+$
aa	$-$	$+$	$+$
aaa	$+$	$+$	$+$
b	$-$	$-$	$-$
ab	$-$	$+$	$-$
aab	$+$	$+$	$-$
$aaaa$	$+$	$+$	$+$
$aaab$	$+$	$+$	$-$

Not closed :
row(aab) not
in S

$\overset{4)}{\Rightarrow}$

$\mathcal{T}_5$	ϵ	a	ba
ϵ	$-$	$-$	$-$
a	$-$	$-$	$+$
aa	$-$	$+$	$+$
aaa	$+$	$+$	$+$
ab	$-$	$+$	$-$
aab	$+$	$+$	$-$
b	$-$	$-$	$-$
$aaaa$	$+$	$+$	$+$
$aaab$	$+$	$+$	$-$
aba	$+$	$-$	$+$
abb	$+$	$-$	$-$
$aaba$	$+$	$-$	$+$
$aabb$	$+$	$-$	$-$

$\overset{5)}{\Rightarrow}$

$\mathcal{T}_6$	ϵ	a	ba
ϵ	$-$	$-$	$-$
a	$-$	$-$	$+$
aa	$-$	$+$	$+$
aaa	$+$	$+$	$+$
ab	$-$	$+$	$-$
aab	$+$	$+$	$-$
$aaba$	$+$	$-$	$+$
$aabb$	$+$	$-$	$-$
b	$-$	$-$	$-$
$aaaa$	$+$	$+$	$+$
$aaab$	$+$	$+$	$-$
aba	$+$	$-$	$+$
abb	$+$	$-$	$-$
$aabaa$	$-$	$+$	$+$
$aabab$	$-$	$+$	$-$
$aabba$	$-$	$-$	$+$
$aabbb$	$-$	$-$	$-$

Table 2. Learning L_2 with L*

The theoretical complexity we obtain for NL* in terms of equivalence queries is higher compared to L* where at most n equivalence queries are needed. The complexity in terms of membership queries is higher for NL* as well (L* needs roughly $|\Sigma|mn^2$ queries). But, we observe that in practice *less* equivalence and membership queries are needed (cf. Section 5).

4 NL* by means of an Example

Suppose $\Sigma = \{a, b\}$ and let $L_n \subseteq \Sigma^*$ be the language of words over Σ containing an a at the $(n{+}1)$-last position according to the regular expression $\Sigma^*\, a\, \Sigma^n$. Then, L_n is accepted by a minimal DFA $\mathcal{A}_n^*$ with 2^{n+1} states. Nevertheless, there are NFA (cf. Figure 2) with only $n+2$ states accepting L_n. It is easy to see that there is even a canonical RFSA $\mathcal{R}_n$ of size $n+2$ accepting L_n. In other words, $\mathcal{R}_n$ is exponentially more succinct than $\mathcal{A}_n^*$.

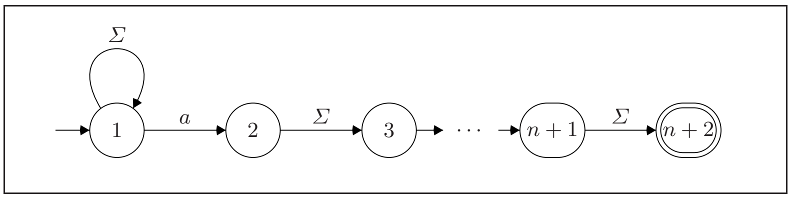

Fig. 2. An NFA over $\Sigma = \{a, b\}$ accepting the regular language L_n with $n + 2$ states

Now we show how L_2 (whose minimal DFA $\mathcal{A}_2^*$ is given in Figure 3) is learned by Angluin's L* algorithm and by our algorithm NL*. We start with a run of L*, which is illustrated in Table 2. Table $\mathcal{T}_1$ is closed and consistent but does not represent the intended automaton because the word aaa is not accepted. Hence, we add $\mathit{Pref}(aaa)$ to U and $\mathit{Pref}(aaa)\Sigma$ to $U\Sigma$. The result (after performing the necessary membership queries) is $\mathcal{T}_2$. This table is closed but not consistent ($row(a) = row(aa)$ but not $row(aa) = row(aaa)$). Thus, we add the column a and obtain $\mathcal{T}_3$, which is still not consistent leading to $\mathcal{T}_4$. After making the table closed, we obtain $\mathcal{T}_6$, which is consistent as well and whose corresponding automaton (Figure 3) accepts L_2.

Now we show a run of NL* on L_2. It is depicted in Table 3. The rows with a preceding $*$ are the prime rows. The table $\mathcal{T}_1$ is RFSA-closed and RFSA-consistent but does not represent the intended

	$\mathcal{T}_1$	ϵ
$*$	ϵ	$-$
$*$	b	$-$
$*$	a	$-$

$\Rightarrow^{1.}_{ce}$

	$\mathcal{T}_2$	ϵ	aaa	aa	a
$*$	ϵ	$-$	$+$	$-$	$-$
$*$	b	$-$	$+$	$-$	$-$
$*$	a	$-$	$+$	$+$	$-$

$\Rightarrow^{2.}_{ncl}$

	$\mathcal{T}_3$	ϵ	aaa	aa	a
$*$	ϵ	$-$	$+$	$-$	$-$
$*$	a	$-$	$+$	$+$	$-$
$*$	b	$-$	$+$	$-$	$-$
$*$	ab	$-$	$+$	$-$	$+$
	aa	$-$	$+$	$+$	$+$

$\Rightarrow^{3.}_{ncl}$

	$\mathcal{T}_4$	ϵ	aaa	aa	a
$*$	ϵ	$-$	$+$	$-$	$-$
$*$	a	$-$	$+$	$+$	$-$
$*$	ab	$-$	$+$	$-$	$+$
$*$	b	$-$	$+$	$-$	$-$
	aa	$-$	$+$	$+$	$+$
$*$	abb	$+$	$+$	$-$	$-$
	aba	$+$	$+$	$+$	$-$

$\Rightarrow^{4.}_{ncl}$

	$\mathcal{T}_5$	ϵ	aaa	aa	a
$*$	ϵ	$-$	$+$	$-$	$-$
$*$	a	$-$	$+$	$+$	$-$
$*$	ab	$-$	$+$	$-$	$+$
$*$	abb	$+$	$+$	$-$	$-$
$*$	b	$-$	$+$	$-$	$-$
	aa	$-$	$+$	$+$	$+$
	aba	$+$	$+$	$+$	$-$
$*$	$abbb$	$-$	$+$	$-$	$-$
$*$	$abba$	$-$	$+$	$+$	$-$

Table 3. Learning L_2 with NL*

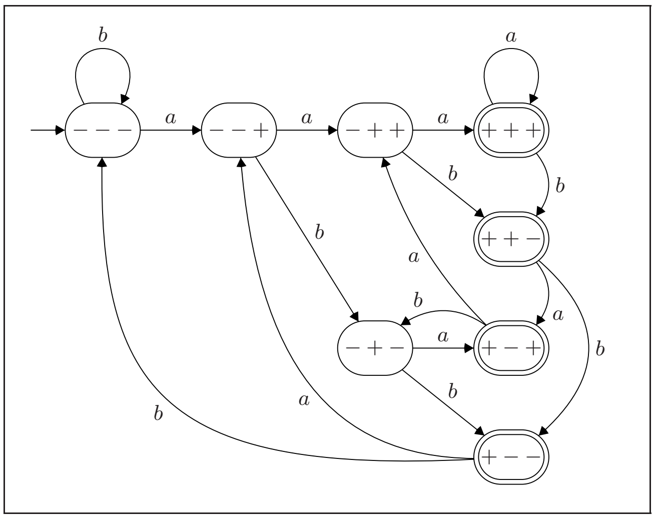

Fig. 3. Minimal DFA $\mathcal{A}_2^*$ accepting language L_2 with 8 ($= 2^{2+1}$) states

automaton because the word *aaa* is not accepted. We add *aaa* and all its suffixes to V, perform membership queries, and then obtain table $\mathcal{T}_2$, which is not closed. We add a to U and continue. After solving two more closedness violations, we obtain finally table $\mathcal{T}_5$ which is RFSA-closed and RFSA-consistent, and its corresponding automaton given in Figure 4 is the canonical RFSA for L_2. Notice that table $\mathcal{T}_5$ is not closed in the Angluin sense and L* would continue adding strings to the upper part of the table.

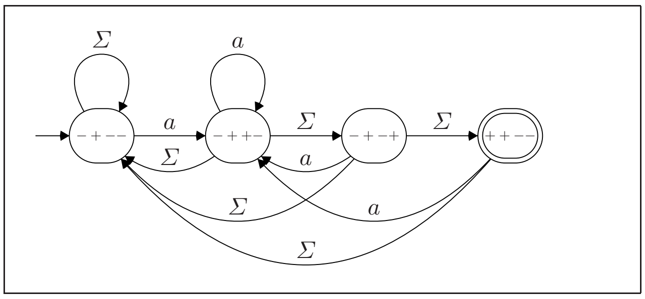

Fig. 4. Canonical RFSA $\mathcal{R}_2$ accepting L_2 with $4 = 2 + 2$ states

5 Experiments

To evaluate the practical performance of NL*, we compare our learning algorithm NL* with Angluin's L* algorithm and its modification according to Remark 1, called L$^*_{\text{col}}$. As NL* is arguably similar in spirit to L$^*_{\text{col}}$, a comparison with this algorithm seems more fair. To this end, all algorithms have been implemented in Java. We have tested the algorithms on a wide range of examples. Following [7], we randomly generate large sets of regular expressions over different sizes of alphabets. A detailed description of this as well as a full description of the outcome can be found in Appendix A. Here, we only present a characteristic selection of the results for an alphabet of size three.

Results For the diagrams in this section, we generated a set of 3180 regular expressions, resulting in minimal DFA with sizes ranging between 1 and 200 states. These DFA were given to the learning algorithms, i.e. membership and equivalence queries were answered according to these automata. To evaluate the algorithms' performances, we measured, for each algorithm and input regular expression, the *number of states of the final automaton* (RSFA or DFA) and the *number of membership (resp. equivalence) queries* to infer it. As Figure 5 shows, the number of states of the automata learned by NL* is considerably smaller than that of L* and L$^*_{\text{col}}$ confirming the results of [7]. More importantly, in practice, the actual sizes of RFSA compared to DFA seem to follow an exponential gap.

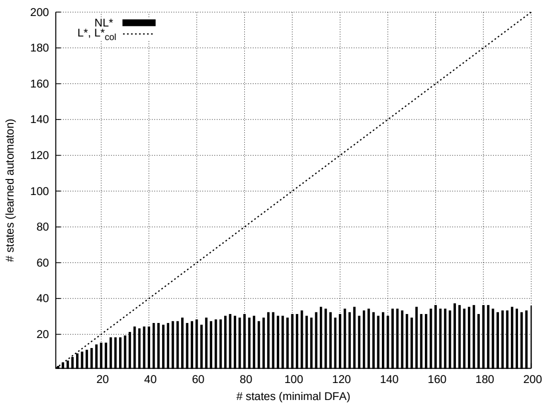

Fig. 5. Number of states

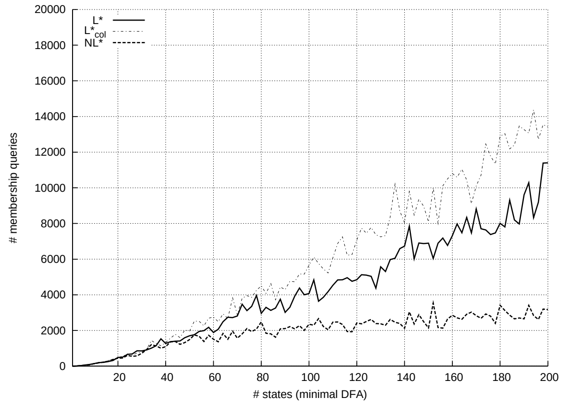

Fig. 6. Number of membership queries

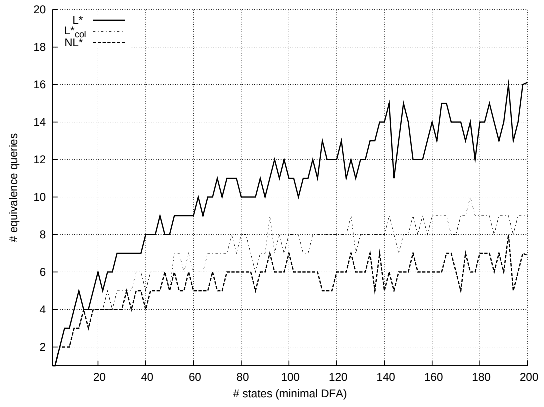

Fig. 7. Number of equivalence queries

In Figure 6, the number of membership queries is depicted. As in the first case, NL* behaves much better than the other learning algorithms. While the difference between the curves is rather small for automata with less than 40 states, it increases significantly for larger automata. The same is the case for the number of equivalence queries depicted in Figure 7. This is in contrast with the theoretical result we obtained in Theorem 3. The experiments we performed point out the clear predominance of NL* over L* and $\mathrm{L}^*_{\mathrm{col}}$ as long as the user is not dependent on a deterministic model.

6 Conclusion

We presented NL*, an algorithm for learning the canonical RFSA of a regular language, using membership and equivalence queries. Theoretically, an RFSA can be at most as big as a DFA accepting the same language, but it might be exponentially more succinct. Our experiments carried out with our implementation shows that this theoretical benefit is indeed the standard case in practice. Moreover, NL* typically needs less membership and equivalence queries, especially for large automata, than the corresponding algorithm L* for learning DFA. Thus, NL* clearly outperforms L*. There is room for further improvement by adapting the various variants of L* developed recently.

Our motivation for studying learning techniques is their application in the area of verification. Most often, non-deterministic automata rather than deterministic automata are sufficient for verification tasks and thus the preferable class of automata to work with. In the future, we plan to show that using our NL* algorithm, the limits of learning-based verification techniques can be pushed ahead considerably.

References

1. R. Alur, P. Cerný, P. Madhusudan, and W. Nam. Synthesis of interface specifications for java classes. In J. Palsberg and M. Abadi, editors, *POPL*, pages 98–109. ACM, 2005.
2. R. Alur, P. Madhusudan, and W. Nam. Symbolic compositional verification by learning assumptions. In *17th International Conference on Computer Aided Verification*, volume 3576 of *Lecture Notes in Computer Science*, pages 548–562. Springer-Verlag, 2005.
3. D. Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2):87–106, 1987.
4. A. W. Biermann and J. A. Feldman. On the synthesis of finite-state machines from samples of their behaviour. *IEEE Transactions on Computers*, 21:592–597, 1972.
5. J. M. Cobleigh, D. Giannakopoulou, and C. S. Pasareanu. Learning assumptions for compositional verification. In H. Garavel and J. Hatcliff, editors, *TACAS*, volume 2619 of *Lecture Notes in Computer Science*, pages 331–346. Springer, 2003.

6. F. Denis, A. Lemay, and A. Terlutte. Residual finite state automata. *Fundamenta Informaticae*, 51(4):339–368, 2002.
7. F. Denis, A. Lemay, and A. Terlutte. Learning regular languages using RFSAs. *Theoretical Comput. Sci.*, 313(2):267–294, 2004.
8. P. García, M. V. de Parga, G. Alvarez, and J. Ruiz. Learning regular languages using nondeterministic finite automata. In *CIAA*, volume 5148 of *LNCS*, pages 92–101. Springer, 2008.
9. O. Grinchtein, M. Leucker, and N. Piterman. Inferring network invariants automatically. In *Proceedings of the 3rd International Joint Conference on Automated Reasoning (IJCAR'06)*, volume 4130 of *Lecture Notes in Artificial Itelligence*, Sept. 2006.
10. A. Groce, D. Peled, and M. Yannakakis. Adaptive model checking. In *Tools and Algorithms for the Construction and Analysis of Systems (TACAS'02)*, volume 2280 of *LNCS*, 2002.
11. P. Habermehl and T. Vojnar. Regular model checking using inference of regular languages. *Electr. Notes Theor. Comput. Sci.*, 138(3):21–36, 2005.
12. V. Kumar, P. Madhusudan, and M. Viswanathan. Minimization, learning, and conformance testing of boolean programs. In C. Baier and H. Hermanns, editors, *CONCUR*, volume 4137 of *Lecture Notes in Computer Science*, pages 203–217. Springer, 2006.
13. K. J. Lang. Random dfa's can be approximately learned from sparse uniform examples. In *COLT*, pages 45–52, 1992.
14. M. Leucker. Learning meets verification. In F. S. de Boer, M. M. Bonsangue, S. Graf, and W. P. de Roever, editors, *FMCO*, volume 4709 of *LNCS*, pages 127–151. Springer, 2006.
15. O. Maler and A. Pnueli. On the learnability of infinitary regular sets. *Inf. Comput.*, 118(2):316–326, 1995.
16. A. Nerode. Linear Automata Transformation. *American Math. Society*, 9:541–544, 1958.
17. A. L. Oliveira and J. P. M. Silva. Efficient algorithms for the inference of minimum size dfas. *Machine Learning*, 44(1/2):93–119, 2001.
18. J. Oncina and P. Garcia. Inferring regular languages in polynomial update time. In *Pattern Recognition and Image Analysis*, volume 1 of *Series in Machine Perception and Artificial Intelligence*, pages 49–61. World Scientific, Singapore, 1992.
19. D. Peled, M. Vardi, and M. Yannakakis. Black box checking. In *Proc. FORTE/PSTV*, pages 225–240. Kluwer, 1999.
20. B. Trakhtenbrot and J. Barzdin. *Finite aut.: behaviour and synthesis*. North-Holland, 1973.
21. A. Vardhan, K. Sen, M. Viswanathan, and G. Agha. Learning to verify safety properties. In *ICFEM*, volume 3308 of *LNCS*, pages 274–289. Springer, 2004.
22. A. Vardhan, K. Sen, M. Viswanathan, and G. Agha. Using language inference to verify omega-regular properties. In *TACAS*, volume 3440 of *LNCS*, pages 45–60. Springer, 2005.

A Additional Experiments

In this section we detail the experiments we performed. We compare our learning algorithm NL* with two other learning algorithms, namely Angluin's L* algorithm and its modification L$^*_{\text{col}}$. In L$^*_{\text{col}}$ counterexamples are treated as in NL*, i.e., instead of adding the counterexample and all its prefixes to the set of rows U, the counterexample w and all its suffixes are added to the set of columns V. Due to this modification NL* and L$^*_{\text{col}}$ become easier to compare.

To obtain maximally significant statistical data, we implemented all three learning algorithms and let them execute the same set of samples. The derivation of the sample sets used in our experiments is described in the following paragraph.

Derivation of Sample Sets As proposed in [7], we randomly generate large sets of regular expressions over different sizes of alphabets. By means of a context-free grammar for inducing regular expressions, we iteratively construct derivations of randomly drawn length by choosing production rules according to fixed probabilities for the three operators *concatenation* $(\cdot, 0.556)$, *choice* $(+, 0.278)$ and *Kleene star* $(^*, 0.166)$. As result, we get an object consisting only of non-terminals and linked by regular expression operators. To obtain the final regular expression, all non-terminals symbols of the derivation are assigned a letter from the corresponding alphabet according to a uniform distribution.

For the statistics enclosed in this paper, we generated several sets of regular expressions for different alphabet sizes ($|\Sigma| \in \{2, 3\}$). All regular expressions had equivalent minimal DFA between 1 and 200 states. These sets were fed to the three learning algorithms and the results stored. To evaluate the algorithms' performances, we recorded the following characteristics for each algorithm and each input regular expression: the *number of states of final automaton*, the *number of membership queries* and the *number of equivalence queries* to infer the final automaton model.

In the next paragraph, we summarize the results obtained from our experiments on 2- and 3-letter alphabets. The results are based on approximately 3200 sample regular expressions, each.

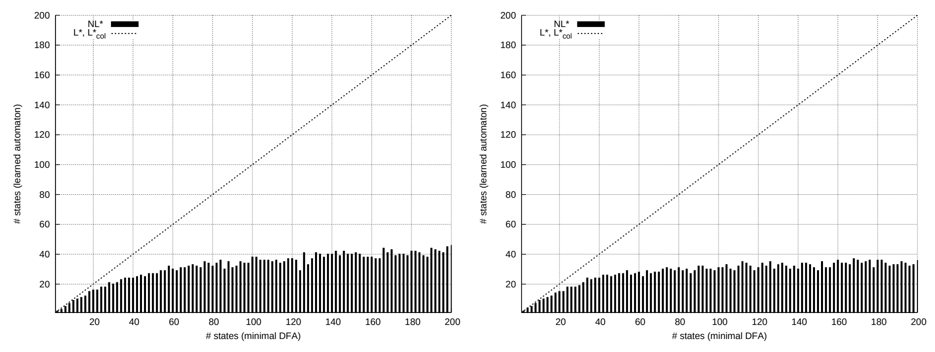

Fig. 8. Number of states of minimal DFA and canonical RFSA (2- and 3-letter alphabet)

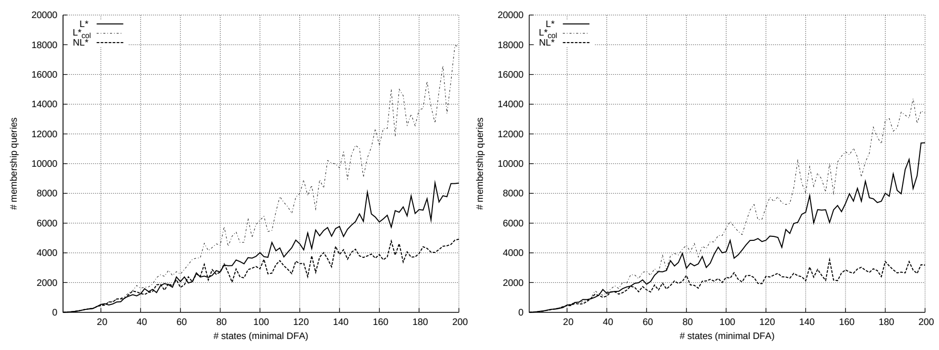

Fig. 9. Number of membership queries (2- and 3-letter alphabet)

Results Figure 8 compares the number of states of the automata learned by L^* (or equivalently, L^*_{col}) (i.e., minimal DFA) and NL^* (i.e., canonical RFSA) for 2- and 3-letter alphabets. The exponential gap between the models learned by L^* and inferred by NL^* is obvious. Figure 9 juxtaposes the number of membership queries of the three algorithms. Note, that for automata of size larger than 40 states, NL^* seems to need increasingly less membership queries than the other two algorithms for inferring deterministic automata. Moreover, we see that in the case of membership queries the basic version of Angluin performs much better than its extended version. This, however, is not the case if one considers the number of equivalence queries necessary to infer the automata. Though NL^* still performs far better than both, L^* and L^*_{col}, the extended version of Angluin now behaves nicer. In almost all cases it needs less equivalence queries than L^*. In many application areas equivalence queries are extremely costly. Hence, for inferring deterministic

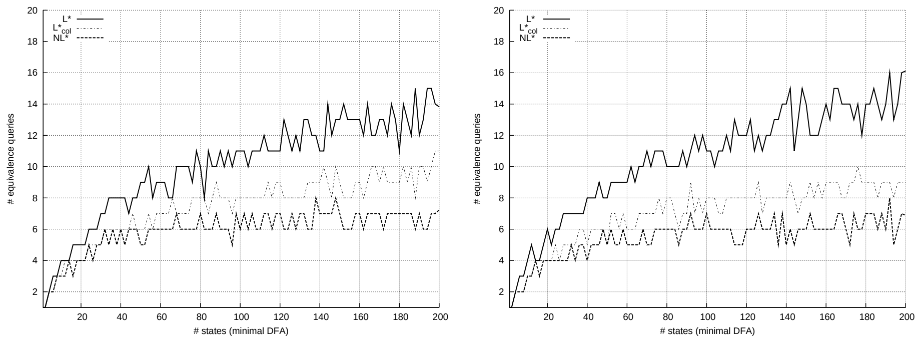

Fig. 10. Number of equivalence queries (2- and 3-letter alphabet)

automata, L^*_{col} might in many cases be of high interest and even preferable to the basic version L^*.

To get a numerical impression of "which algorithm is superior to which", consider the Tables 4 and 5. They emphasize the results mentioned above but also show that, in all cases, there is a significant number of wins of NL^* over the other two algorithms.

NL^* and L^*	Won	Lost	Tie
States	95.78%	0.0%	4.22%
Membership queries	77.04%	22.01%	0.95%
Equivalence queries	89.64%	2.24%	8.12%

NL^* and L^*_{col}	Won	Lost	Tie
States	95.78%	0.0%	4.22%
Membership queries	88.85%	7.87%	3.28%
Equivalence queries	65.10%	13.32%	21.58%

Table 4. Comparing NL^* to L^* and L^*_{col} (2-letter alphabet)

NL^* and L^*	Won	Lost	Tie
States	95.91%	0.0%	4.09%
Membership queries	81.92%	16.95%	1.13%
Equivalence queries	90.16%	2.20%	7.64%

NL^* and L^*_{col}	Won	Lost	Tie
States	95.91%	0.0%	4.09%
Membership queries	89.71%	7.08%	3.21%
Equivalence queries	64.34%	14.06%	21.60%

Table 5. Comparing NL^* to L^* and L^*_{col} (3-letter alphabet)

The tables' entries have to be understood as follows: the column "Won" describes the number of times in our experiments where NL^* was superior to L^* or L^*_{col}, respectively, i.e., whenever the difference between number of states of the automaton derived by L^* and by NL^* was positive. Similarly, column "Lost" describes when this number is negative. In case the difference is equal to 0 we have a "Tie". The same numbers were calculated for the membership and equivalence queries and depicted in lines 2 and 3 of the corresponding tables.

B An example where NL^* needs more membership queries than L^*

In this section, we see an example where NL^* needs more membership queries than its deterministic version L^*. Moreover, the resulting automata have got the same number of states.

Let a parameterized minimal DFA over $\Sigma = \{a, b\}$ be $\mathcal{A}^*_n = \langle Q, q_0, F, \delta \rangle$ (for $n > 2$), let $\mathcal{R}_n$ be the corresponding canonical RFSA, and let $L_n = L(\mathcal{A}^*_n) = L(\mathcal{R}_n)$. Hereby, $\mathcal{A}^*_n$ is given by:

- $Q = \{q_i \mid 0 \leq i \leq n-1\}$
- $F = Q \setminus \{q_{n-1}\}$
- $\delta(q_i, a) = \delta(q_i, b) = q_{i+1}$ for $0 \leq i \leq n-3$, $\delta(q_{n-2}, a) = q_{n-1}$, $\delta(q_{n-2}, b) = q_0$, $\delta(q_{n-1}, a) = \delta(q_{n-1}, b) = q_{n-1}$.

Figure 11 shows the minimal DFA $\mathcal{A}^*_5$ and Figure 12 the corresponding canonical RFSA $\mathcal{R}_5$.

Lemma 6 (Learning RFSA might need more membership queries than learning DFA). *In comparison to learning the minimal DFA $\mathcal{A}_n^*$ using L^*, learning the canonical RFSA $\mathcal{R}_n$ requires $n-2$ additional resolutions of inconsistencies (but no additional equivalence queries).*

Proof: The proof can be done via induction on the number of states and is easily traceable following the example below. Note that NL^* needs indeed $(2n+1)\cdot(n-2)$ more membership queries than L^*. $\square$

Learning the DFA from Figure 11 and its canonical RFSA (Figure 12) is depicted in Tables 6 and 7, respectively.

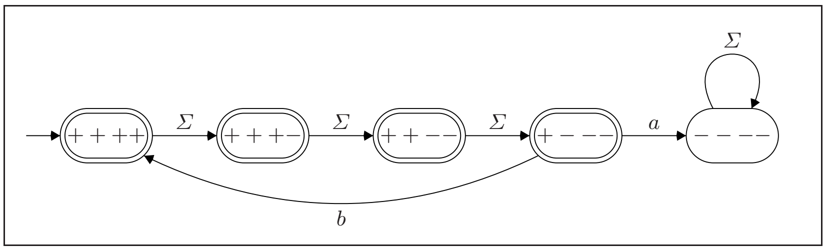

Fig. 11. The minimal DFA $\mathcal{A}_5^*$ over $\Sigma = \{a, b\}$ for which NL^* needs more membership queries than L^*

$\mathcal{T}_1$	ε
ε	$+$
a	$+$
b	$+$

$\overset{1)}{\Rightarrow}$

$\mathcal{T}_2$	ε
ε	$+$
a	$+$
aa	$+$
aaa	$+$
$aaaa$	$-$
b	$+$
ab	$+$
aab	$+$
$aaab$	$+$
$aaaaa$	$-$
$aaaab$	$-$

$\overset{2)}{\Rightarrow}$

$\mathcal{T}_3$	ε	a
ε	$+$	$+$
a	$+$	$+$
aa	$+$	$+$
aaa	$+$	$-$
$aaaa$	$-$	$-$
b	$+$	$+$
ab	$+$	$+$
aab	$+$	$-$
$aaab$	$+$	$+$
$aaaaa$	$-$	$-$
$aaaab$	$-$	$-$

$\overset{3)}{\Rightarrow}$

$\mathcal{T}_4$	ε	a	aa
ε	$+$	$+$	$+$
a	$+$	$+$	$+$
aa	$+$	$+$	$-$
aaa	$+$	$-$	$-$
$aaaa$	$-$	$-$	$-$
b	$+$	$+$	$+$
ab	$+$	$+$	$-$
aab	$+$	$-$	$-$
$aaab$	$+$	$+$	$+$
$aaaaa$	$-$	$-$	$-$
$aaaab$	$-$	$-$	$-$

$\overset{4)}{\Rightarrow}$

$\mathcal{T}_5$	ε	a	aa	aaa
ε	$+$	$+$	$+$	$+$
a	$+$	$+$	$+$	$-$
aa	$+$	$+$	$-$	$-$
aaa	$+$	$-$	$-$	$-$
$aaaa$	$-$	$-$	$-$	$-$
b	$+$	$+$	$+$	$-$
ab	$+$	$+$	$-$	$-$
aab	$+$	$-$	$-$	$-$
$aaab$	$+$	$+$	$+$	$+$
$aaaaa$	$-$	$-$	$-$	$-$
$aaaab$	$-$	$-$	$-$	$-$

1) $\mathcal{T}_1$ is closed and consistent but a counterexample can be obtained because $aaaa \in L(\mathcal{A}_{\mathcal{T}_1})$ and $aaaa \notin L$. Hence, add $\mathit{Pref}(aaaa)$ to U.
2) $\mathcal{T}_2$ is not consistent: $T(aa) = T(aaa)$ but $T(aaa) \neq T(aaaa)$. Hence, add a to V.
3) $\mathcal{T}_3$ is not consistent: $T(a) = T(aa)$ but $T(aa) \neq T(aaa)$. Hence, add aa to V.
4) $\mathcal{T}_4$ is not consistent: $T(\varepsilon) = T(a)$ but $T(a) \neq T(aa)$. Hence, add aaa to V.
5) $\mathcal{T}_5$ is then closed and consistent and the minimal DFA $\mathcal{A}_5^*$ from Figure 11 can be derived.

Table 6. Learning $\mathcal{A}_5^*$ with L^*

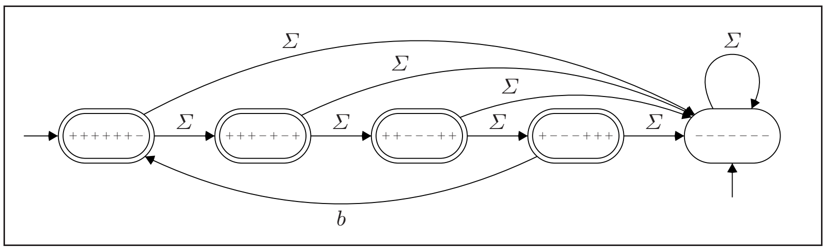

Fig. 12. RFSA $\mathcal{R}_5$ recognizing the language of the minimal DFA $\mathcal{A}_5^*$ from Figure 11

	$\mathcal{T}_0$	ε
$*$	ε	+
$*$	b	+
$*$	a	+

$\Rightarrow^{1.}_{ce}$

	$\mathcal{T}_1$	ε	$aaaa$	aaa	aa	a
$*$	ε	+	−	+	+	+
$*$	b	+	−	−	+	+
$*$	a	+	−	−	+	+

$\Rightarrow^{2.}_{ncl}$

	$\mathcal{T}_2$	ε	$aaaa$	aaa	aa	a
$*$	ε	+	−	+	+	+
$*$	b	+	−	−	+	+
$*$	a	+	−	−	+	+
$*$	bb	+	−	−	−	+
$*$	ba	+	−	−	−	+

$\Rightarrow^{3.}_{ncl}$

	$\mathcal{T}_3$	ε	$aaaa$	aaa	aa	a
$*$	ε	+	−	+	+	+
$*$	b	+	−	−	+	+
$*$	bb	+	−	−	−	+
$*$	a	+	−	−	+	+
$*$	ba	+	−	−	−	+
$*$	bbb	+	−	−	−	−
$*$	bba	+	−	−	−	−

$\Rightarrow^{4.}_{ncl}$

	$\mathcal{T}_4$	ε	$aaaa$	aaa	aa	a
$*$	ε	+	−	+	+	+
$*$	b	+	−	−	+	+
$*$	bb	+	−	−	−	+
$*$	bbb	+	−	−	−	−
$*$	a	+	−	−	+	+
$*$	ba	+	−	−	−	+
$*$	bba	+	−	−	−	−
$*$	$bbbb$	+	−	+	+	+
$*$	$bbba$	−	−	−	−	−

$\Rightarrow^{5.}_{ncl}$

	$\mathcal{T}_5$	ε	$aaaa$	aaa	aa	a
$*$	ε	+	−	+	+	+
$*$	b	+	−	−	+	+
$*$	bb	+	−	−	−	+
$*$	bbb	+	−	−	−	−
$*$	$bbba$	−	−	−	−	−
$*$	a	+	−	−	+	+
$*$	ba	+	−	−	−	+
$*$	bba	+	−	−	−	−
$*$	$bbbb$	+	−	+	+	+
$*$	$bbbab$	−	−	−	−	−
$*$	$bbbaa$	−	−	−	−	−

$\Rightarrow^{6.}_{ncs}$

	$\mathcal{T}_6$	ε	$aaaa$	aaa	aa	a	$baaa$
$*$	ε	+	−	+	+	+	−
$*$	b	+	−	−	+	+	−
$*$	bb	+	−	−	−	+	−
$*$	bbb	+	−	−	−	−	+
$*$	$bbba$	−	−	−	−	−	−
$*$	a	+	−	−	+	+	−
$*$	ba	+	−	−	−	+	−
$*$	bba	+	−	−	−	−	+
$*$	$bbbb$	+	−	+	+	+	−
$*$	$bbbab$	−	−	−	−	−	−
$*$	$bbbaa$	−	−	−	−	−	−

$\Rightarrow^{7.}_{ncs}$

	$\mathcal{T}_7$	ε	$aaaa$	aaa	aa	a	$baaa$	$bbaaa$
$*$	ε	+	−	+	+	+	−	−
$*$	b	+	−	−	+	+	−	−
$*$	bb	+	−	−	−	+	−	+
$*$	bbb	+	−	−	−	−	+	−
$*$	$bbba$	−	−	−	−	−	−	−
$*$	a	+	−	−	+	+	−	−
$*$	ba	+	−	−	−	+	−	+
$*$	bba	+	−	−	−	−	+	−
$*$	$bbbb$	+	−	+	+	+	−	−
$*$	$bbbab$	−	−	−	−	−	−	−
$*$	$bbbaa$	−	−	−	−	−	−	−

$\Rightarrow^{8.}_{ncs}$

	$\mathcal{T}_8$	ε	$aaaa$	aaa	aa	a	$baaa$	$bbaaa$	$bbbaaa$
$*$	ε	+	−	+	+	+	−	−	−
$*$	b	+	−	−	+	+	−	−	+
$*$	bb	+	−	−	−	+	−	+	−
$*$	bbb	+	−	−	−	−	+	−	−
$*$	$bbba$	−	−	−	−	−	−	−	−
$*$	a	+	−	−	+	+	−	−	+
$*$	ba	+	−	−	−	+	−	+	−
$*$	bba	+	−	−	−	−	+	−	−
$*$	$bbbb$	+	−	+	+	+	−	−	−
$*$	$bbbab$	−	−	−	−	−	−	−	−
$*$	$bbbaa$	−	−	−	−	−	−	−	−

1) Table $\mathcal{T}_0$ is closed and consistent but a counterexample can be found: $aaaa$ is accepted by the hypothesis but is not in the language we want to infer.
2) Table $\mathcal{T}_0$ violates the closedness property. Hence, we try to make $\mathcal{T}_1$ closed by moving row b to U.
3) – 5) We get three more closedness violations and resolve them by moving rows bb, bbb and $bbbb$ to the upper part of the table.
6) Table $\mathcal{T}_5$ violates the consistency property because $bbb \sqsubseteq bb$ but $bbbb \not\sqsubseteq bbb$. We try to obtain consistency by adding suffix $baaa$ to V.
7) – 8) But still, two more inconsistencies occur. Hence we add suffix $bbaaa$ and $bbbaaa$ to V.
9) Table $\mathcal{T}_8$ is closed and consistent and the final model can be calculated (cf. Figure 12).

Table 7. Learning $\mathcal{R}_5$ with NL*

C An example where NL* needs more equivalence queries than L*

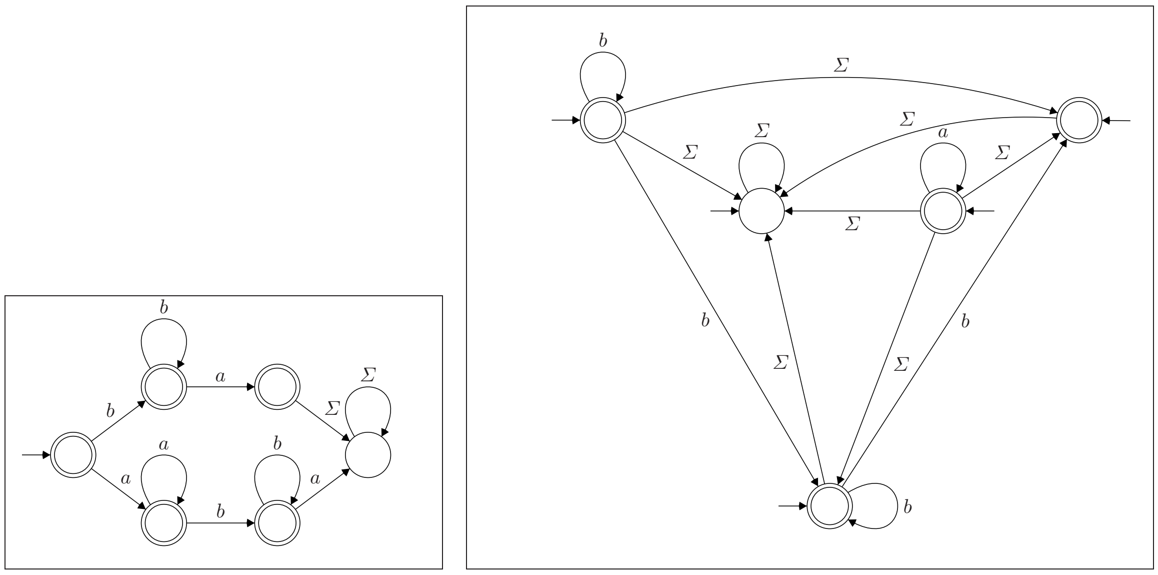

Fig. 13. Minimal DFA to infer (left) and its canonical RFSA (right)

$\mathcal{T}_0$	ε
ε	$+$
b	$+$
a	$+$

$\Rightarrow^{1.}_{ce}$

$\mathcal{T}_1$	ε
ε	$+$
a	$+$
aba	$-$
ab	$+$
b	$+$
aa	$+$
$abab$	$-$
$abaa$	$-$
abb	$+$

$\Rightarrow^{2.}_{ncs}$

$\mathcal{T}_2$	ε	a
ε	$+$	$+$
a	$+$	$+$
aba	$-$	$-$
ab	$+$	$-$
b	$+$	$+$
aa	$+$	$+$
$abab$	$-$	$-$
$abaa$	$-$	$-$
abb	$+$	$-$

$\Rightarrow^{3.}_{ncs}$

$\mathcal{T}_3$	ε	a	ba
ε	$+$	$+$	$+$
a	$+$	$+$	$-$
aba	$-$	$-$	$-$
ab	$+$	$-$	$-$
b	$+$	$+$	$+$
aa	$+$	$+$	$-$
$abab$	$-$	$-$	$-$
$abaa$	$-$	$-$	$-$
abb	$+$	$-$	$-$

$\Rightarrow^{4.}_{ce}$

$\mathcal{T}_4$	ε	a	ba
ε	$+$	$+$	$+$
b	$+$	$+$	$+$
a	$+$	$+$	$-$
aba	$-$	$-$	$-$
ab	$+$	$-$	$-$
baa	$-$	$-$	$-$
ba	$+$	$-$	$-$
aa	$+$	$+$	$-$
$abab$	$-$	$-$	$-$
$abaa$	$-$	$-$	$-$
abb	$+$	$-$	$-$
bb	$+$	$+$	$+$
$baab$	$-$	$-$	$-$
$baaa$	$-$	$-$	$-$
bab	$-$	$-$	$-$

$\Rightarrow^{5.}_{ncs}$

$\mathcal{T}_5$	ε	a	ba	aa
ε	$+$	$+$	$+$	$+$
b	$+$	$+$	$+$	$-$
a	$+$	$+$	$-$	$+$
aba	$-$	$-$	$-$	$-$
ab	$+$	$-$	$-$	$-$
baa	$-$	$-$	$-$	$-$
ba	$+$	$-$	$-$	$-$
aa	$+$	$+$	$-$	$+$
$abab$	$-$	$-$	$-$	$-$
$abaa$	$-$	$-$	$-$	$-$
abb	$+$	$-$	$-$	$-$
bb	$+$	$+$	$+$	$-$
$baab$	$-$	$-$	$-$	$-$
$baaa$	$-$	$-$	$-$	$-$
bab	$-$	$-$	$-$	$-$

$\Rightarrow^{6.}_{ncs}$

$\mathcal{T}_6$	ε	a	ba	aa	b
ε	$+$	$+$	$+$	$+$	$+$
b	$+$	$+$	$+$	$-$	$+$
a	$+$	$+$	$-$	$+$	$+$
aba	$-$	$-$	$-$	$-$	$-$
ab	$+$	$-$	$-$	$-$	$+$
baa	$-$	$-$	$-$	$-$	$-$
ba	$+$	$-$	$-$	$-$	$-$
aa	$+$	$+$	$-$	$+$	$+$
$abab$	$-$	$-$	$-$	$-$	$-$
$abaa$	$-$	$-$	$-$	$-$	$-$
abb	$+$	$-$	$-$	$-$	$+$
bb	$+$	$+$	$+$	$-$	$+$
$baab$	$-$	$-$	$-$	$-$	$-$
$baaa$	$-$	$-$	$-$	$-$	$-$
bab	$-$	$-$	$-$	$-$	$-$

1) found counterexample: aba for current model $\mathcal{A}_0$ (based on $\mathcal{T}_0$)
2) consistency violation: Trying to obtain consistency (1) for $\mathcal{T}_1$ by adding suffix: a
3) consistency violation: Trying to obtain consistency (1) for $\mathcal{T}_2$ by adding suffix: ba
4) found counterexample: baa for current model $\mathcal{A}_3$ (based on $\mathcal{T}_3$)
5) consistency violation: Trying to obtain consistency (1) for $\mathcal{T}_4$ by adding suffix: aa
6) consistency violation: Trying to obtain consistency (1) for $\mathcal{T}_5$ by adding suffix: b
7) Table $\mathcal{T}_6$ is closed and consistent. Final model calculated (cf. Figure 13).

Table 8. An example of an L* run that needs less equivalence queries than NL*

	$\mathcal{T}_0$	ε
$*$	ε	$+$
$*$	b	$+$
$*$	a	$+$

$\Rightarrow^{1.}_{ce}$

	$\mathcal{T}_1$	ε	aba	ba	a
$*$	ε	$+$	$-$	$+$	$+$
$*$	b	$+$	$-$	$+$	$+$
$*$	a	$+$	$-$	$-$	$+$

$\Rightarrow^{2.}_{ncl}$

	$\mathcal{T}_2$	ε	aba	ba	a
$*$	ε	$+$	$-$	$+$	$+$
$*$	a	$+$	$-$	$-$	$+$
$*$	b	$+$	$-$	$+$	$+$
$*$	ab	$+$	$-$	$-$	$-$
$*$	aa	$+$	$-$	$-$	$+$

$\Rightarrow^{3.}_{ncl}$

	$\mathcal{T}_3$	ε	aba	ba	a
$*$	ε	$+$	$-$	$+$	$+$
$*$	a	$+$	$-$	$-$	$+$
$*$	ab	$+$	$-$	$-$	$-$
$*$	b	$+$	$-$	$+$	$+$
$*$	aa	$+$	$-$	$-$	$+$
$*$	abb	$+$	$-$	$-$	$-$
$*$	aba	$-$	$-$	$-$	$-$

$\Rightarrow^{4.}_{ncl}$

	$\mathcal{T}_4$	ε	aba	ba	a
$*$	ε	$+$	$-$	$+$	$+$
$*$	a	$+$	$-$	$-$	$+$
$*$	ab	$+$	$-$	$-$	$-$
$*$	aba	$-$	$-$	$-$	$-$
$*$	b	$+$	$-$	$+$	$+$
$*$	aa	$+$	$-$	$-$	$+$
$*$	abb	$+$	$-$	$-$	$-$
$*$	$abab$	$-$	$-$	$-$	$-$
$*$	$abaa$	$-$	$-$	$-$	$-$

$\Rightarrow^{5.}_{ce}$

	$\mathcal{T}_5$	ε	aba	ba	a	baa	aa
	ε	$+$	$-$	$+$	$+$	$-$	$+$
$*$	a	$+$	$-$	$-$	$+$	$-$	$+$
$*$	ab	$+$	$-$	$-$	$-$	$-$	$-$
$*$	aba	$-$	$-$	$-$	$-$	$-$	$-$
$*$	b	$+$	$-$	$+$	$+$	$-$	$-$
$*$	aa	$+$	$-$	$-$	$+$	$-$	$+$
$*$	abb	$+$	$-$	$-$	$-$	$-$	$-$
$*$	$abab$	$-$	$-$	$-$	$-$	$-$	$-$
$*$	$abaa$	$-$	$-$	$-$	$-$	$-$	$-$

$\Rightarrow^{6.}_{ncl}$

	$\mathcal{T}_6$	ε	aba	ba	a	baa	aa
	ε	$+$	$-$	$+$	$+$	$-$	$+$
$*$	b	$+$	$-$	$+$	$+$	$-$	$-$
$*$	a	$+$	$-$	$-$	$+$	$-$	$+$
$*$	ab	$+$	$-$	$-$	$-$	$-$	$-$
$*$	aba	$-$	$-$	$-$	$-$	$-$	$-$
$*$	aa	$+$	$-$	$-$	$+$	$-$	$+$
$*$	abb	$+$	$-$	$-$	$-$	$-$	$-$
$*$	$abab$	$-$	$-$	$-$	$-$	$-$	$-$
$*$	$abaa$	$-$	$-$	$-$	$-$	$-$	$-$
$*$	bb	$+$	$-$	$+$	$+$	$-$	$-$
$*$	ba	$+$	$-$	$-$	$-$	$-$	$-$

$\Rightarrow^{7.}_{ce}$

	$\mathcal{T}_7$	ε	aba	ba	a	baa	aa	bab	ab	b
	ε	$+$	$-$	$+$	$+$	$-$	$+$	$-$	$+$	$+$
$*$	b	$+$	$-$	$+$	$+$	$-$	$-$	$-$	$-$	$+$
$*$	a	$+$	$-$	$-$	$+$	$-$	$+$	$-$	$+$	$+$
$*$	ab	$+$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$+$
$*$	aba	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
$*$	aa	$+$	$-$	$-$	$+$	$-$	$+$	$-$	$+$	$+$
$*$	abb	$+$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$+$
$*$	$abab$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
$*$	$abaa$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
$*$	bb	$+$	$-$	$+$	$+$	$-$	$-$	$-$	$-$	$+$
$*$	ba	$+$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$

$\Rightarrow^{8.}_{ncl}$

	$\mathcal{T}_8$	ε	aba	ba	a	baa	aa	bab	ab	b
	ε	$+$	$-$	$+$	$+$	$-$	$+$	$-$	$+$	$+$
$*$	b	$+$	$-$	$+$	$+$	$-$	$-$	$-$	$-$	$+$
$*$	a	$+$	$-$	$-$	$+$	$-$	$+$	$-$	$+$	$+$
$*$	ab	$+$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$+$
$*$	aba	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
$*$	ba	$+$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
$*$	aa	$+$	$-$	$-$	$+$	$-$	$+$	$-$	$+$	$+$
$*$	abb	$+$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$+$
$*$	$abab$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
$*$	$abaa$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
$*$	bb	$+$	$-$	$+$	$+$	$-$	$-$	$-$	$-$	$+$
$*$	bab	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
$*$	baa	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$

1) found counterexample: *aba* for current model $\mathcal{A}_0$ (based on $\mathcal{T}_0$)
2) closedness violation: Trying to make $\mathcal{T}_1$ closed with row *a*
3) closedness violation: Trying to make $\mathcal{T}_2$ closed with row *ab*
4) closedness violation: Trying to make $\mathcal{T}_3$ closed with row *aba*
5) found counterexample: *baa* for current model $\mathcal{A}_4$ (based on $\mathcal{T}_4$)
6) closedness violation: Trying to make $\mathcal{T}_5$ closed with row *b*
7) found counterexample: *bab* for current model $\mathcal{A}_6$ (based on $\mathcal{T}_6$)
8) closedness violation: Trying to make $\mathcal{T}_7$ closed with row *ba*
9) Table $\mathcal{T}_8$ is closed and consistent. Final model calculated (cf. Figure 13).

Table 9. An example of an NL* run that needs more equivalence queries than L*

D An example where the number of states does not increase after inserting a counterexample

In contrast to Angluin's learning algorithm L*, in NL* it might be the case that the resolution of a consistency violation does not introduce a new state. As shown in Theorem 3, the termination of the algorithm can still be assured.

Let $\Sigma = \{a, b, c\}$. We want to learn the RFSA $\mathcal{R}_2$ from Figure 14. The minimal DFA is depicted in Figure 15.

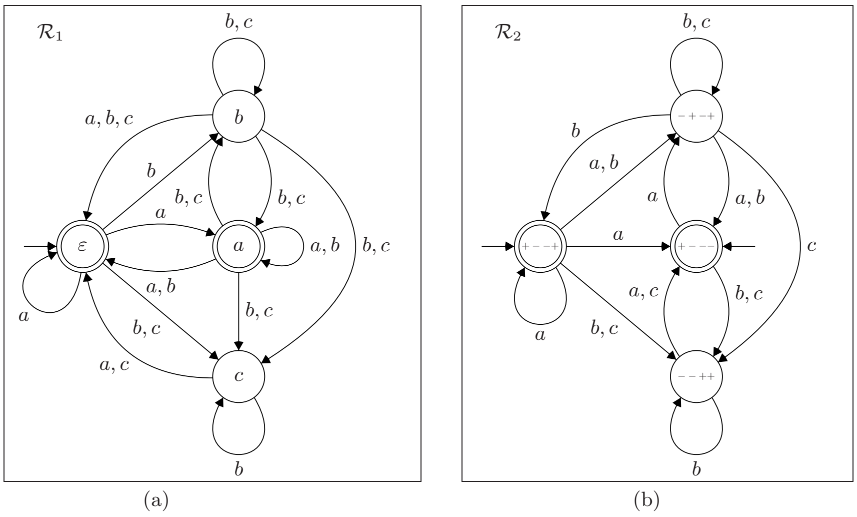

Fig. 14. (a) RFSA $\mathcal{R}_1$ derived from table $\mathcal{T}_7$, (b) RFSA $\mathcal{R}_2$ derived from table $\mathcal{T}_8$

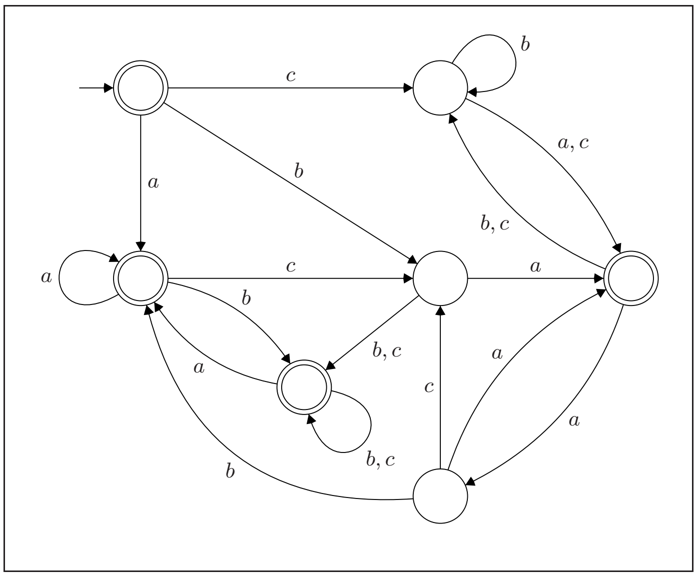

Fig. 15. The minimal DFA corresponding to RFSA $\mathcal{R}_2$ from Figure 14

	$\mathcal{T}_0$	ε
$*$	ε	$+$
$*$	b	$-$
$*$	c	$-$
$*$	a	$+$

$\Rightarrow^{1.}_{ncl}$

	$\mathcal{T}_1$	ε
$*$	ε	$+$
$*$	b	$-$
$*$	c	$-$
$*$	a	$+$
$*$	bb	$+$
$*$	bc	$+$
$*$	ba	$+$

$\Rightarrow^{2.}_{ncs}$

	$\mathcal{T}_2$	ε	b
$*$	ε	$+$	$-$
$*$	b	$-$	$+$
$*$	c	$-$	$-$
	a	$+$	$+$
	bb	$+$	$+$
	bc	$+$	$+$
$*$	ba	$+$	$-$

$\Rightarrow^{3.}_{ncl}$

	$\mathcal{T}_3$	ε	b
$*$	ε	$+$	$-$
$*$	b	$-$	$+$
$*$	c	$-$	$-$
	a	$+$	$+$
	bb	$+$	$+$
	bc	$+$	$+$
$*$	ba	$+$	$-$
$*$	cb	$-$	$-$
$*$	cc	$+$	$-$
$*$	ca	$+$	$-$

$\Rightarrow^{4.}_{ncs}$

	$\mathcal{T}_4$	ε	b	c
$*$	ε	$+$	$-$	$-$
$*$	b	$-$	$+$	$+$
$*$	c	$-$	$-$	$+$
$*$	a	$+$	$+$	$-$
	bb	$+$	$+$	$+$
	bc	$+$	$+$	$+$
$*$	ba	$+$	$-$	$-$
$*$	cb	$-$	$-$	$+$
$*$	cc	$+$	$-$	$-$
$*$	ca	$+$	$-$	$-$

$\Rightarrow^{5.}_{ncl}$

	$\mathcal{T}_5$	ε	b	c
$*$	ε	$+$	$-$	$-$
$*$	b	$-$	$+$	$+$
$*$	c	$-$	$-$	$+$
$*$	a	$+$	$+$	$-$
	bb	$+$	$+$	$+$
	bc	$+$	$+$	$+$
$*$	ba	$+$	$-$	$-$
$*$	cb	$-$	$-$	$+$
$*$	cc	$+$	$-$	$-$
$*$	ca	$+$	$-$	$-$
	ab	$+$	$+$	$+$
$*$	ac	$-$	$+$	$+$
$*$	aa	$+$	$+$	$-$

$\Rightarrow^{6.}_{ce}$

	$\mathcal{T}_6$	ε	b	c	baa	aa	a
$*$	ε	$+$	$-$	$-$	$-$	$+$	$+$
$*$	b	$-$	$+$	$+$	$+$	$-$	$+$
$*$	c	$-$	$-$	$+$	$-$	$-$	$+$
$*$	a	$+$	$+$	$-$	$+$	$+$	$+$
	bb	$+$	$+$	$+$	$+$	$+$	$+$
	bc	$+$	$+$	$+$	$+$	$+$	$+$
$*$	ba	$+$	$-$	$-$	$-$	$+$	$-$
$*$	cb	$-$	$-$	$+$	$-$	$-$	$+$
$*$	cc	$+$	$-$	$-$	$-$	$+$	$-$
$*$	ca	$+$	$-$	$-$	$-$	$+$	$-$
	ab	$+$	$+$	$+$	$+$	$+$	$+$
$*$	ac	$-$	$+$	$+$	$+$	$-$	$+$
$*$	aa	$+$	$+$	$-$	$+$	$+$	$+$

$\Rightarrow^{7.}_{ncl}$

	$\mathcal{T}_7$	ε	b	c	baa	aa	a
$*$	ε	$+$	$-$	$-$	$-$	$+$	$+$
	b	$-$	$+$	$+$	$+$	$-$	$+$
$*$	c	$-$	$-$	$+$	$-$	$-$	$+$
	a	$+$	$+$	$-$	$+$	$+$	$+$
$*$	ba	$+$	$-$	$-$	$-$	$+$	$-$
	bb	$+$	$+$	$+$	$+$	$+$	$+$
	bc	$+$	$+$	$+$	$+$	$+$	$+$
$*$	cb	$-$	$-$	$+$	$-$	$-$	$+$
$*$	cc	$+$	$-$	$-$	$-$	$+$	$-$
$*$	ca	$+$	$-$	$-$	$-$	$+$	$-$
	ab	$+$	$+$	$+$	$+$	$+$	$+$
	ac	$-$	$+$	$+$	$+$	$-$	$+$
	aa	$+$	$+$	$-$	$+$	$+$	$+$
$*$	bab	$-$	$-$	$+$	$-$	$-$	$+$
$*$	bac	$-$	$-$	$+$	$-$	$-$	$+$
$*$	baa	$-$	$+$	$-$	$+$	$-$	$+$

$\Rightarrow^{8.}_{ncl}$

	$\mathcal{T}_8$	ε	b	c	baa	aa	a
$*$	ε	$+$	$-$	$-$	$-$	$+$	$+$
	b	$-$	$+$	$+$	$+$	$-$	$+$
$*$	c	$-$	$-$	$+$	$-$	$-$	$+$
	a	$+$	$+$	$-$	$+$	$+$	$+$
$*$	ba	$+$	$-$	$-$	$-$	$+$	$-$
$*$	baa	$-$	$+$	$-$	$+$	$-$	$+$
	bb	$+$	$+$	$+$	$+$	$+$	$+$
	bc	$+$	$+$	$+$	$+$	$+$	$+$
$*$	cb	$-$	$-$	$+$	$-$	$-$	$+$
$*$	cc	$+$	$-$	$-$	$-$	$+$	$-$
$*$	ca	$+$	$-$	$-$	$-$	$+$	$-$
	ab	$+$	$+$	$+$	$+$	$+$	$+$
	ac	$-$	$+$	$+$	$+$	$-$	$+$
	aa	$+$	$+$	$-$	$+$	$+$	$+$
$*$	bab	$-$	$-$	$+$	$-$	$-$	$+$
$*$	bac	$-$	$-$	$+$	$-$	$-$	$+$
	$baab$	$+$	$+$	$-$	$+$	$+$	$+$
	$baac$	$-$	$+$	$+$	$+$	$-$	$+$
$*$	$baaa$	$+$	$-$	$-$	$-$	$+$	$-$

1) $\mathcal{T}_0$ is not closed. Hence, we try to make it closed by moving row b to the upper table.
2) $\mathcal{T}_1$ violates the consistency property, as $b \sqsubseteq \epsilon$ but $bb \not\sqsubseteq b$. We try to obtain consistency by adding suffix b to V.
3) After resolving the inconsistency the table is not closed. Thus, we add row c to U.
4) Table $\mathcal{T}_3$ is inconsistent again ($c \sqsubseteq \epsilon$ but $cc \not\sqsubseteq c$) and we insert suffix c to V.
5) A closedness violationforces us to move row a to U.
6) As table $\mathcal{T}_5$ is closed and consistent, a hypothesis can be derived (cf. Figure 14 (a)) and a counterexample is found. The word baa is accepted by our hypothesis but not contained in the language to learn. Hence, we add $\mathit{Suff}(baa)$ to V.

7) − 9) Resolving two more closedness violations by moving rows ba and baa to the upper table, table $\mathcal{T}_8$ is closed and consistent, again. The final model is calculated and depicted in Figure 14 (b).

Table 10. An example of an NL* run where the number of states does not increase

E An example where an intermediate hypothesis is not an RFSA

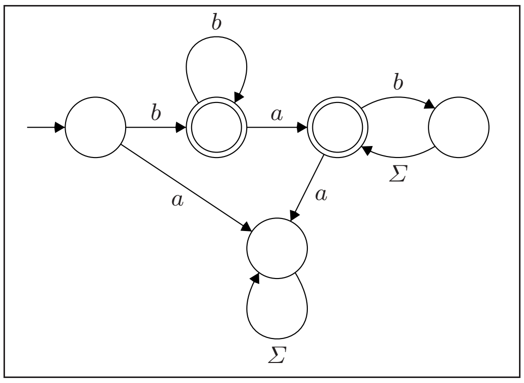

Fig. 16. Minimal DFA recognizing the language to infer

	$\mathcal{T}_0$	ε
$*$	ε	$-$
$*$	b	$+$
$*$	a	$-$

$\Rightarrow^{1.}_{ncl}$

	$\mathcal{T}_1$	ε
$*$	ε	$-$
$*$	b	$+$
$*$	a	$-$
$*$	bb	$+$
$*$	ba	$+$

$\Rightarrow^{2.}_{ce}$

	$\mathcal{T}_2$	ε	ab	b
$*$	ε	$-$	$-$	$+$
	b	$+$	$-$	$+$
$*$	a	$-$	$-$	$-$
	bb	$+$	$-$	$+$
$*$	ba	$+$	$-$	$-$

$\Rightarrow^{3.}_{ncl}$

	$\mathcal{T}_3$	ε	ab	b
$*$	ε	$-$	$-$	$+$
	b	$+$	$-$	$+$
$*$	a	$-$	$-$	$-$
	bb	$+$	$-$	$+$
$*$	ba	$+$	$-$	$-$
$*$	ab	$-$	$-$	$-$
$*$	aa	$-$	$-$	$-$

$\Rightarrow^{4.}_{ncl}$

	$\mathcal{T}_4$	ε	ab	b
$*$	ε	$-$	$-$	$+$
	b	$+$	$-$	$+$
$*$	a	$-$	$-$	$-$
$*$	ba	$+$	$-$	$-$
	bb	$+$	$-$	$+$
$*$	ab	$-$	$-$	$-$
$*$	aa	$-$	$-$	$-$
$*$	bab	$-$	$-$	$+$
$*$	baa	$-$	$-$	$-$

$\Rightarrow^{5.}_{ce}$

	$\mathcal{T}_5$	ε	ab	b	ba	a
$*$	ε	$-$	$-$	$+$	$+$	$-$
	b	$+$	$-$	$+$	$+$	$+$
$*$	a	$-$	$-$	$-$	$-$	$-$
$*$	ba	$+$	$-$	$-$	$+$	$-$
	bb	$+$	$-$	$+$	$+$	$+$
$*$	ab	$-$	$-$	$-$	$-$	$-$
$*$	aa	$-$	$-$	$-$	$-$	$-$
$*$	bab	$-$	$-$	$+$	$-$	$+$
$*$	baa	$-$	$-$	$-$	$-$	$-$

$\Rightarrow^{6.}_{ncl}$

	$\mathcal{T}_6$	ε	ab	b	ba	a
$*$	ε	$-$	$-$	$+$	$+$	$-$
	b	$+$	$-$	$+$	$+$	$+$
$*$	a	$-$	$-$	$-$	$-$	$-$
$*$	ba	$+$	$-$	$-$	$+$	$-$
$*$	bab	$-$	$-$	$+$	$-$	$+$
	bb	$+$	$-$	$+$	$+$	$+$
$*$	ab	$-$	$-$	$-$	$-$	$-$
$*$	aa	$-$	$-$	$-$	$-$	$-$
$*$	baa	$-$	$-$	$-$	$-$	$-$
$*$	$babb$	$+$	$-$	$-$	$+$	$-$
$*$	$baba$	$+$	$-$	$-$	$+$	$-$

1) closedness violation: Trying to make $\mathcal{T}_0$ closed with row: b ($[+]$)
2) found counterexample: ab for current model $\mathcal{A}_1$ (based on $\mathcal{T}_1$) (cf. Figure 17).
3) closedness violation: Trying to make $\mathcal{T}_2$ closed with row: a ($[-,-,-]$)
4) closedness violation: Trying to make $\mathcal{T}_3$ closed with row: ba ($[+,-,-]$)
5) found counterexample: ba for current model $\mathcal{A}_4$ (based on $\mathcal{T}_4$) (cf. Figure 17).
6) closedness violation: Trying to make $\mathcal{T}_5$ closed with row: bab ($[-,-,+,-,+]$)
7) Table $\mathcal{T}_6$ is closed and consistent. Final model calculated (cf. Figure 17).

Table 11. An example of an NL* run where an intermediate hypothesis is not an RFSA

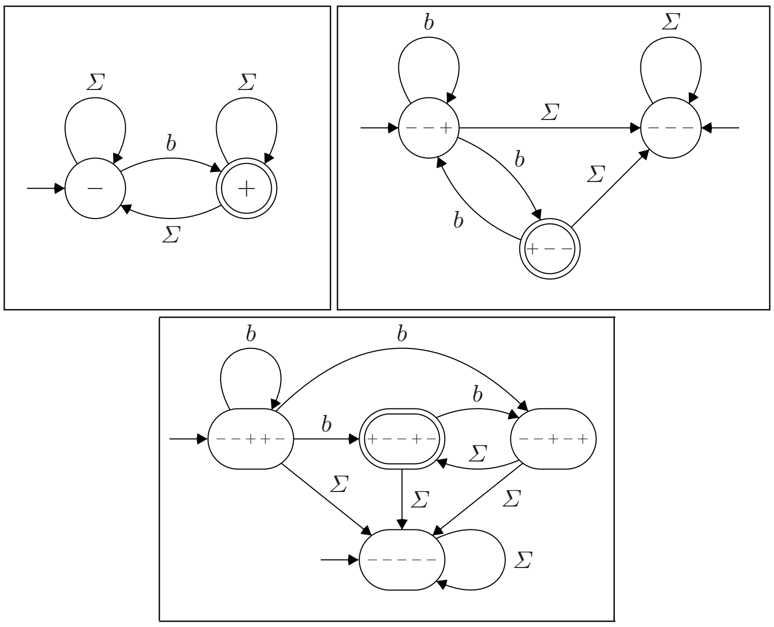

Fig. 17. Intermediate hypothesis automata $\mathcal{A}_1$ and $\mathcal{A}_4$ (for tables $\mathcal{T}_1$ and $\mathcal{T}_4$, respectively), and final hypothesis $\mathcal{A}_6$. Note that $\mathcal{A}_4$ in the upper right corner **is not an RFSA**, because the state at the bottom accepts bbb^*, which is not a residual of $L(\mathcal{A}_4) = bb^*$.

F An example demonstrating that the RFSA algorithm would not terminate if we added counterexamples to U instead of V

	$\mathcal{T}_0$	ε
$*$	ε	$+$
$*$	b	$+$
$*$	a	$+$

$\Longrightarrow^{1.}_{ce}$

	$\mathcal{T}_1$	ε
$*$	ε	$+$
$*$	a	$+$
$*$	aa	$-$
$*$	b	$+$
$*$	ab	$-$
$*$	aab	$-$
$*$	aaa	$-$

$\Longrightarrow^{2.}_{ncs}$

	$\mathcal{T}_2$	ε	b
	ε	$+$	$+$
$*$	a	$+$	$-$
$*$	aa	$-$	$-$
	b	$+$	$+$
$*$	ab	$-$	$+$
$*$	aab	$-$	$-$
$*$	aaa	$-$	$-$

$\Longrightarrow^{3.}_{ncl}$

	$\mathcal{T}_3$	ε	b
	ε	$+$	$+$
$*$	a	$+$	$-$
$*$	aa	$-$	$-$
$*$	ab	$-$	$+$
	b	$+$	$+$
$*$	aab	$-$	$-$
$*$	aaa	$-$	$-$
	abb	$+$	$+$
$*$	aba	$-$	$-$

1) found counterexample aa for current model based on $\mathcal{T}_0$.
2) consistency (1) violation: Trying to obtain consistency (1) for $\mathcal{T}_1$ by adding suffix b.
3) closedness violation: Trying to make $\mathcal{T}_2$ closed with row ab ($[-,+]$).
4) trying to add counterexample $a \in L(\mathcal{A}) \setminus L(\mathcal{R}_{\mathcal{T}_3})$ fails, because a is already present in $\mathcal{T}_3$ (cf. Figures 18 and 19).

Table 12. The non-termination problem with the algorithm that adds counterexamples to U

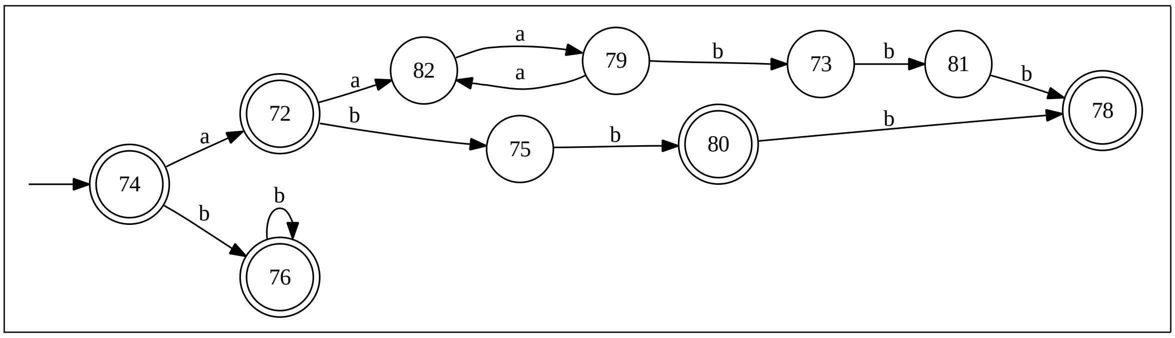

Fig. 18. Input DFA $\mathcal{A}$ (without trap)

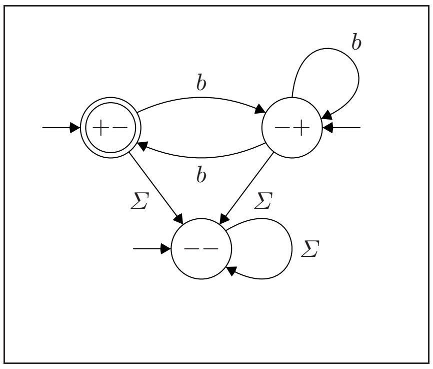

Fig. 19. Automaton $\mathcal{R}_{\mathcal{T}_3}$ for table $\mathcal{T}_3$

We now present a successful run of the correct algorithm NL* on the input language given by the DFA in Fugure 18. Several intermediate automata are depicted in the subsequent figures.

	$\mathcal{T}_0$	ε
$*$	ε	$+$
$*$	b	$+$
$*$	a	$+$

$\Longrightarrow^{1.}_{ce}$

	$\mathcal{T}_1$	ε	aa	a
$*$	ε	$+$	$-$	$+$
$*$	b	$+$	$-$	$-$
$*$	a	$+$	$-$	$-$

$\Longrightarrow^{2.}_{ncl}$

	$\mathcal{T}_2$	ε	aa	a
$*$	ε	$+$	$-$	$+$
$*$	b	$+$	$-$	$-$
$*$	a	$+$	$-$	$-$
$*$	bb	$+$	$-$	$-$
$*$	ba	$-$	$-$	$-$

$\Longrightarrow^{3.}_{ncl}$

	$\mathcal{T}_3$	ε	aa	a
$*$	ε	$+$	$-$	$+$
$*$	b	$+$	$-$	$-$
$*$	ba	$-$	$-$	$-$
$*$	a	$+$	$-$	$-$
$*$	bb	$+$	$-$	$-$
$*$	bab	$-$	$-$	$-$
$*$	baa	$-$	$-$	$-$

$\Longrightarrow^{4.}_{ce}$

	$\mathcal{T}_4$	ε	aa	a	ab	b
$*$	ε	$+$	$-$	$+$	$-$	$+$
$*$	b	$+$	$-$	$-$	$-$	$+$
$*$	ba	$-$	$-$	$-$	$-$	$-$
$*$	a	$+$	$-$	$-$	$-$	$-$
$*$	bb	$+$	$-$	$-$	$-$	$+$
$*$	bab	$-$	$-$	$-$	$-$	$-$
$*$	baa	$-$	$-$	$-$	$-$	$-$

$\Longrightarrow^{5.}_{ncl}$

	$\mathcal{T}_5$	ε	aa	a	ab	b
$*$	ε	$+$	$-$	$+$	$-$	$+$
	b	$+$	$-$	$-$	$-$	$+$
$*$	a	$+$	$-$	$-$	$-$	$-$
$*$	ba	$-$	$-$	$-$	$-$	$-$
	bb	$+$	$-$	$-$	$-$	$+$
$*$	bab	$-$	$-$	$-$	$-$	$-$
$*$	baa	$-$	$-$	$-$	$-$	$-$
$*$	ab	$-$	$-$	$-$	$-$	$+$
$*$	aa	$-$	$-$	$-$	$-$	$-$

$\Longrightarrow^{6.}_{ncl}$

	$\mathcal{T}_6$	ε	aa	a	ab	b
$*$	ε	$+$	$-$	$+$	$-$	$+$
	b	$+$	$-$	$-$	$-$	$+$
$*$	a	$+$	$-$	$-$	$-$	$-$
$*$	ba	$-$	$-$	$-$	$-$	$-$
$*$	ab	$-$	$-$	$-$	$-$	$+$
	bb	$+$	$-$	$-$	$-$	$+$
$*$	bab	$-$	$-$	$-$	$-$	$-$
$*$	baa	$-$	$-$	$-$	$-$	$-$
$*$	aa	$-$	$-$	$-$	$-$	$-$
	abb	$+$	$-$	$-$	$-$	$+$
$*$	aba	$-$	$-$	$-$	$-$	$-$

$\Longrightarrow^{7.}_{ce}$

	$\mathcal{T}_7$	ε	aa	a	ab	b	$abbbb$	$bbbb$	bbb	bb
$*$	ε	+	−	+	−	+	−	+	+	+
$*$	b	+	−	−	−	+	−	+	+	+
$*$	a	+	−	−	−	−	−	−	+	+
$*$	ba	−	−	−	−	−	−	−	−	−
$*$	ab	−	−	−	−	+	−	−	−	+
$*$	bb	+	−	−	−	+	−	+	+	+
$*$	bab	−	−	−	−	−	−	−	−	−
$*$	baa	−	−	−	−	−	−	−	−	−
$*$	aa	−	−	−	−	−	−	−	−	−
$*$	abb	+	−	−	−	+	−	−	−	−
$*$	aba	−	−	−	−	−	−	−	−	−

$\Longrightarrow^{8.}_{ncl}$

	$\mathcal{T}_8$	ε	aa	a	ab	b	$abbbb$	$bbbb$	bbb	bb
$*$	ε	+	−	+	−	+	−	+	+	+
$*$	b	+	−	−	−	+	−	+	+	+
$*$	a	+	−	−	−	−	−	−	+	+
$*$	ba	−	−	−	−	−	−	−	−	−
$*$	ab	−	−	−	−	+	−	−	−	+
$*$	abb	+	−	−	−	+	−	−	−	−
$*$	bb	+	−	−	−	+	−	+	+	+
$*$	bab	−	−	−	−	−	−	−	−	−
$*$	baa	−	−	−	−	−	−	−	−	−
$*$	aa	−	−	−	−	−	−	−	−	−
$*$	aba	−	−	−	−	−	−	−	−	−
$*$	$abbb$	+	−	−	−	−	−	−	−	−
$*$	$abba$	−	−	−	−	−	−	−	−	−

$\Longrightarrow^{9.}_{ncl}$

	$\mathcal{T}_9$	ε	aa	a	ab	b	$abbbb$	$bbbb$	bbb	bb
$*$	ε	+	−	+	−	+	−	+	+	+
$*$	b	+	−	−	−	+	−	+	+	+
$*$	a	+	−	−	−	−	−	−	+	+
$*$	ba	−	−	−	−	−	−	−	−	−
$*$	ab	−	−	−	−	+	−	−	−	+
$*$	abb	+	−	−	−	+	−	−	−	−
$*$	$abbb$	+	−	−	−	−	−	−	−	−
$*$	bb	+	−	−	−	+	−	+	+	+
$*$	bab	−	−	−	−	−	−	−	−	−
$*$	baa	−	−	−	−	−	−	−	−	−
$*$	aa	−	−	−	−	−	−	−	−	−
$*$	aba	−	−	−	−	−	−	−	−	−
$*$	$abba$	−	−	−	−	−	−	−	−	−
$*$	$abbbb$	−	−	−	−	−	−	−	−	−
$*$	$abbba$	−	−	−	−	−	−	−	−	−

$\Longrightarrow^{10.}_{ce}$

	$\mathcal{T}_{10}$	ε	aa	a	ab	b	$abbbb$	$bbbb$	bbb	bb	$aaabbb$	$aabbb$	$abbb$
$*$	ε	+	−	+	−	+	−	+	+	+	+	−	+
$*$	b	+	−	−	−	+	−	+	+	+	−	−	−
$*$	a	+	−	−	−	−	−	−	+	+	−	+	−
$*$	ba	−	−	−	−	−	−	−	−	−	−	−	−
$*$	ab	−	−	−	−	+	−	−	−	+	−	−	−
$*$	abb	+	−	−	−	+	−	−	−	−	−	−	−
$*$	$abbb$	+	−	−	−	−	−	−	−	−	−	−	−
$*$	bb	+	−	−	−	+	−	+	+	+	−	−	−
$*$	bab	−	−	−	−	−	−	−	−	−	−	−	−
$*$	baa	−	−	−	−	−	−	−	−	−	−	−	−
$*$	aa	−	−	−	−	−	−	−	−	−	+	−	+
$*$	aba	−	−	−	−	−	−	−	−	−	−	−	−
$*$	$abba$	−	−	−	−	−	−	−	−	−	−	−	−
$*$	$abbbb$	−	−	−	−	−	−	−	−	−	−	−	−
$*$	$abbba$	−	−	−	−	−	−	−	−	−	−	−	−

$\Longrightarrow^{11.}_{ncl}$

	$\mathcal{T}_{11}$	ε	aa	a	ab	b	$abbbb$	$bbbb$	bbb	bb	$aaabbb$	$aabbb$	$abbb$
$*$	ε	+	−	+	−	+	−	+	+	+	+	−	+
$*$	b	+	−	−	−	+	−	+	+	+	−	−	−
$*$	a	+	−	−	−	−	−	−	+	+	−	+	−
$*$	ba	−	−	−	−	−	−	−	−	−	−	−	−
$*$	ab	−	−	−	−	+	−	−	−	+	−	−	−
$*$	aa	−	−	−	−	−	−	−	−	−	+	−	+
$*$	abb	+	−	−	−	+	−	−	−	−	−	−	−
$*$	$abbb$	+	−	−	−	−	−	−	−	−	−	−	−
$*$	bb	+	−	−	−	+	−	+	+	+	−	−	−
$*$	bab	−	−	−	−	−	−	−	−	−	−	−	−
$*$	baa	−	−	−	−	−	−	−	−	−	−	−	−
$*$	aba	−	−	−	−	−	−	−	−	−	−	−	−
$*$	$abba$	−	−	−	−	−	−	−	−	−	−	−	−
$*$	$abbbb$	−	−	−	−	−	−	−	−	−	−	−	−
$*$	$abbba$	−	−	−	−	−	−	−	−	−	−	−	−
$*$	aab	−	−	−	−	−	−	−	−	−	−	−	−
$*$	aaa	−	−	−	−	−	−	−	+	−	−	+	−

$\Longrightarrow^{12.}_{ncl}$

	$\mathcal{T}_{12}$	ε	aa	a	ab	b	$abbbb$	$bbbb$	bbb	bb	$aaabbb$	$aabbb$	$abbb$
$*$	ε	+	−	+	−	+	−	+	+	+	+	−	+
$*$	b	+	−	−	−	+	−	+	+	+	−	−	−
	a	+	−	−	−	−	−	−	+	+	−	+	−
$*$	ba	−	−	−	−	−	−	−	−	−	−	−	−
$*$	ab	−	−	−	−	+	−	−	−	+	−	−	−
$*$	aa	−	−	−	−	−	−	−	−	−	+	−	+
$*$	abb	+	−	−	−	+	−	−	−	−	−	−	−
$*$	$abbb$	+	−	−	−	−	−	−	−	−	−	−	−
$*$	aaa	−	−	−	−	−	−	−	+	−	−	+	−
$*$	bb	+	−	−	−	+	−	+	+	+	−	−	−
$*$	bab	−	−	−	−	−	−	−	−	−	−	−	−
$*$	baa	−	−	−	−	−	−	−	−	−	−	−	−
$*$	aba	−	−	−	−	−	−	−	−	−	−	−	−
$*$	$abba$	−	−	−	−	−	−	−	−	−	−	−	−
$*$	$abbbb$	−	−	−	−	−	−	−	−	−	−	−	−
$*$	$abbba$	−	−	−	−	−	−	−	−	−	−	−	−
$*$	aab	−	−	−	−	−	−	−	−	−	−	−	−
$*$	$aaab$	−	−	−	−	−	−	−	−	+	−	−	−
$*$	$aaaa$	−	−	−	−	−	−	−	−	−	+	−	+

$\Longrightarrow^{13.}_{ncl}$

	$\mathcal{T}_{13}$	ε	aa	a	ab	b	$abbbb$	$bbbb$	bbb	bb	$aaabbb$	$aabbb$	$abbb$
$*$	ε	+	−	+	−	+	−	+	+	+	+	−	+
$*$	b	+	−	−	−	+	−	+	+	+	−	−	−
	a	+	−	−	−	−	−	−	+	+	−	+	−
$*$	ba	−	−	−	−	−	−	−	−	−	−	−	−
	ab	−	−	−	−	+	−	−	−	+	−	−	−
$*$	aa	−	−	−	−	−	−	−	−	−	+	−	+
	abb	+	−	−	−	+	−	−	−	−	−	−	−
$*$	$abbb$	+	−	−	−	−	−	−	−	−	−	−	−
$*$	aaa	−	−	−	−	−	−	−	+	−	−	+	−
$*$	$aaab$	−	−	−	−	−	−	−	−	+	−	−	−
$*$	bb	+	−	−	−	+	−	+	+	+	−	−	−
$*$	bab	−	−	−	−	−	−	−	−	−	−	−	−
$*$	baa	−	−	−	−	−	−	−	−	−	−	−	−
$*$	aba	−	−	−	−	−	−	−	−	−	−	−	−
$*$	$abba$	−	−	−	−	−	−	−	−	−	−	−	−
$*$	$abbbb$	−	−	−	−	−	−	−	−	−	−	−	−
$*$	$abbba$	−	−	−	−	−	−	−	−	−	−	−	−
$*$	aab	−	−	−	−	−	−	−	−	−	−	−	−
$*$	$aaaa$	−	−	−	−	−	−	−	−	−	+	−	+
$*$	$aaabb$	−	−	−	−	+	−	−	−	−	−	−	−
$*$	$aaaba$	−	−	−	−	−	−	−	−	−	−	−	−

$\Longrightarrow^{14.}_{ncl}$

	$\mathcal{T}_{14}$	ε	aa	a	ab	b	$abbbb$	$bbbb$	bbb	bb	$aaabbb$	$aabbb$	$abbb$
$*$	ε	+	−	+	−	+	−	+	+	+	+	−	+
$*$	b	+	−	−	−	+	−	+	+	+	−	−	−
	a	+	−	−	−	−	−	−	+	+	−	+	−
$*$	ba	−	−	−	−	−	−	−	−	−	−	−	−
	ab	−	−	−	−	+	−	−	−	+	−	−	−
$*$	aa	−	−	−	−	−	−	−	−	−	+	−	+
	abb	+	−	−	−	+	−	−	−	−	−	−	−
$*$	$abbb$	+	−	−	−	−	−	−	−	−	−	−	−
$*$	aaa	−	−	−	−	−	−	−	+	−	−	+	−
$*$	$aaab$	−	−	−	−	−	−	−	−	+	−	−	−
$*$	$aaabb$	−	−	−	−	+	−	−	−	−	−	−	−
$*$	bb	+	−	−	−	+	−	+	+	+	−	−	−
$*$	bab	−	−	−	−	−	−	−	−	−	−	−	−
$*$	baa	−	−	−	−	−	−	−	−	−	−	−	−
$*$	aba	−	−	−	−	−	−	−	−	−	−	−	−
$*$	$abba$	−	−	−	−	−	−	−	−	−	−	−	−
$*$	$abbbb$	−	−	−	−	−	−	−	−	−	−	−	−
$*$	$abbba$	−	−	−	−	−	−	−	−	−	−	−	−
$*$	aab	−	−	−	−	−	−	−	−	−	−	−	−
$*$	$aaaa$	−	−	−	−	−	−	−	−	−	+	−	+
$*$	$aaaba$	−	−	−	−	−	−	−	−	−	−	−	−
$*$	$aaabbb$	+	−	−	−	−	−	−	−	−	−	−	−
$*$	$aaabba$	−	−	−	−	−	−	−	−	−	−	−	−

1) found counterexample: aa for current model $\mathcal{A}_0$ (based on $\mathcal{T}_0$).
2) closedness violation: Trying to make $\mathcal{T}_1$ closed with row: b ($[+, -, -]$)
3) closedness violation: Trying to make $\mathcal{T}_2$ closed with row: ba ($[-, -, -]$)
4) found counterexample: ab for current model $\mathcal{A}_3$ (based on $\mathcal{T}_3$) (cf. Figure 20).
5) closedness violation: Trying to make $\mathcal{T}_4$ closed with row: a ($[+, -, -, -, -]$)
6) closedness violation: Trying to make $\mathcal{T}_5$ closed with row: ab ($[-, -, -, -, +]$)
7) found counterexample: $abbbb$ for current model $\mathcal{A}_6$ (based on $\mathcal{T}_6$) (cf. Figure 21).
8) closedness violation: Trying to make $\mathcal{T}_7$ closed with row: abb ($[+, -, -, -, +, -, -, -, -]$)
9) closedness violation: Trying to make $\mathcal{T}_8$ closed with row: $abbb$ ($[+, -, -, -, -, -, -, -, -]$)
10) found counterexample: $aaabbb$ for current model $\mathcal{A}_9$ (based on $\mathcal{T}_9$) (cf. Figure 22).
11) closedness violation: Trying to make $\mathcal{T}_{10}$ closed with row: aa ($[-, -, -, -, -, -, -, -, -, +, -, +]$)
12) closedness violation: Trying to make $\mathcal{T}_{11}$ closed with row: aaa ($[-, -, -, -, -, -, -, +, -, -, +, -]$)
13) closedness violation: Trying to make $\mathcal{T}_{12}$ closed with row: $aaab$ ($[-, -, -, -, -, -, -, -, +, -, -, -]$)
14) closedness violation: Trying to make $\mathcal{T}_{13}$ closed with row: $aaabb$ ($[-, -, -, -, +, -, -, -, -, -, -, -]$)
15) Table $\mathcal{T}_{14}$ is closed and consistent. Final model calculated (cf. Figure 23).

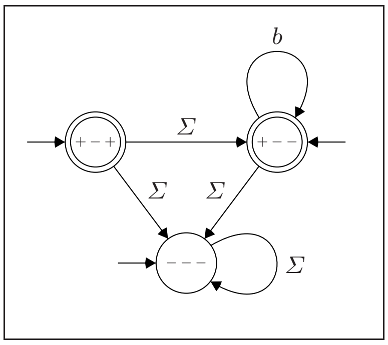

Fig. 20. Automaten $\mathcal{A}_3$ for table $\mathcal{T}_3$ (correct algorithm)

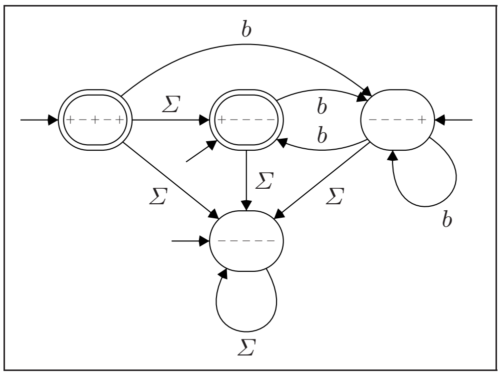

Fig. 21. Automaton $\mathcal{A}_6$ for table $\mathcal{T}_6$ (correct algorithm)

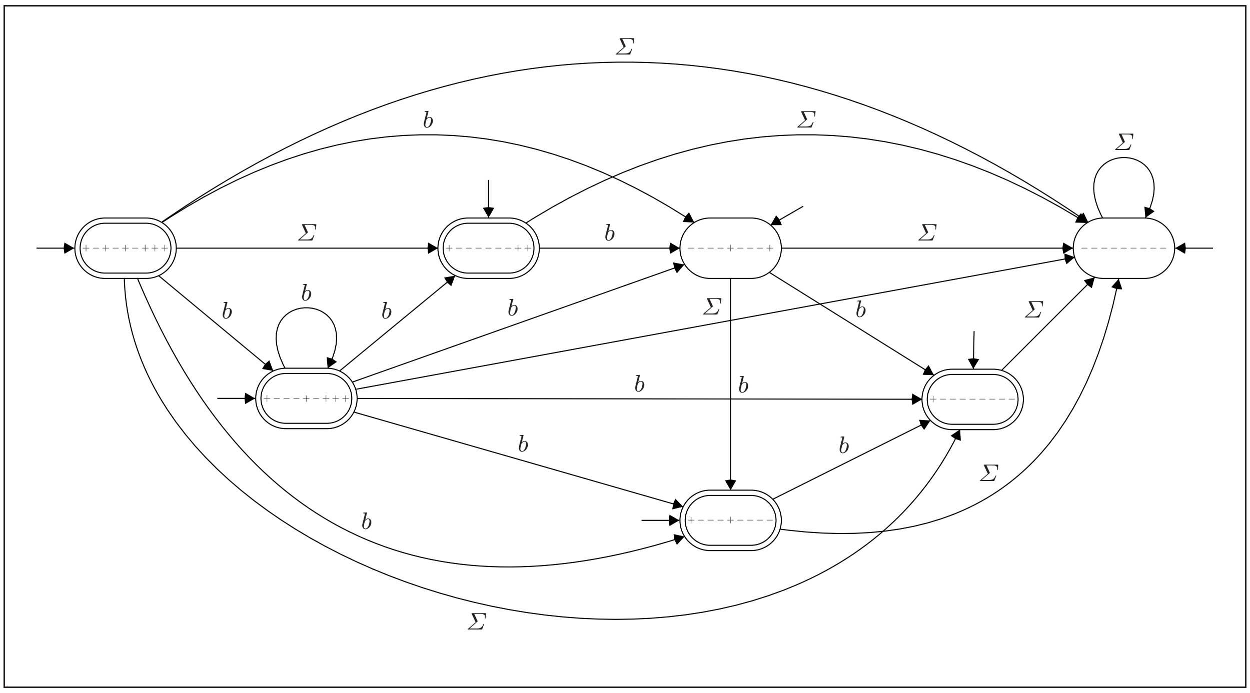

Fig. 22. Automaton $\mathcal{A}_9$ for table $\mathcal{T}_9$ (correct algorithm)

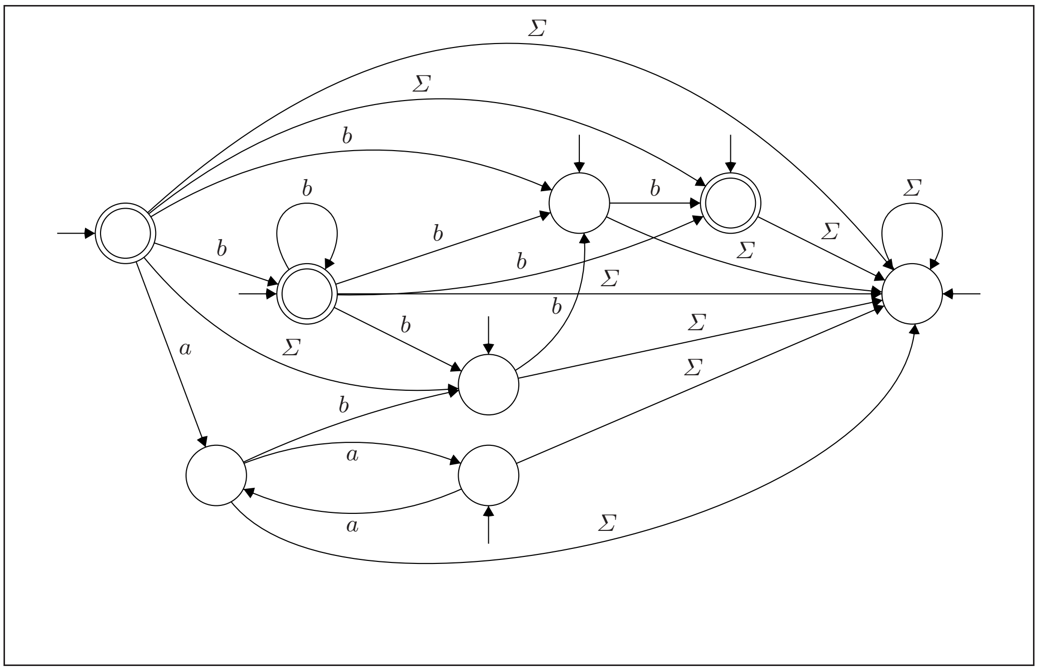

Fig. 23. Automaton $\mathcal{A}_{14}$ for table $\mathcal{T}_{14}$ (correct algorithm)

G Evolution of the measure of a table during an NL* run

In this section, an example run of the NL* algorithm will exemplify the evolution of the measure $M(\mathcal{T})$ that we associated with a table $\mathcal{T}$ in the proof of Theorem 3. Table 13 depicts the tables that we construct during an NL* run, as well as their associated measure. The input automaton is taken from Figure 24. An intermediate NFA and the final canonical RFSA are given by Figures 25 and 26, respectively.

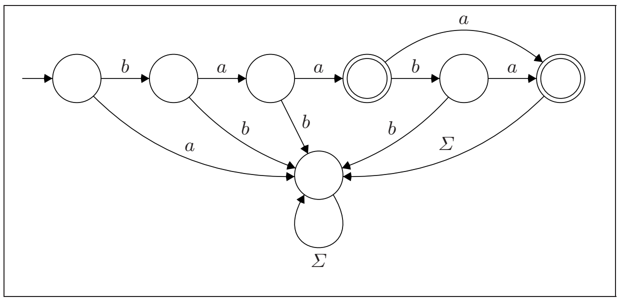

Fig. 24. Minimal DFA for the language to learn

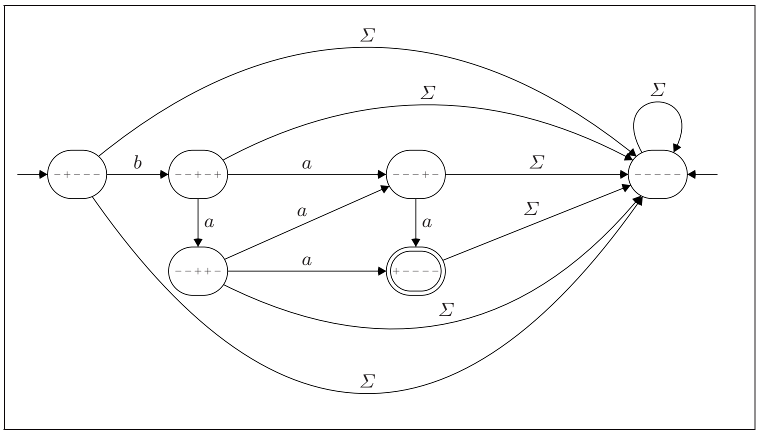

Fig. 25. Automaton $\mathcal{A}_8$ for table $\mathcal{T}_8$

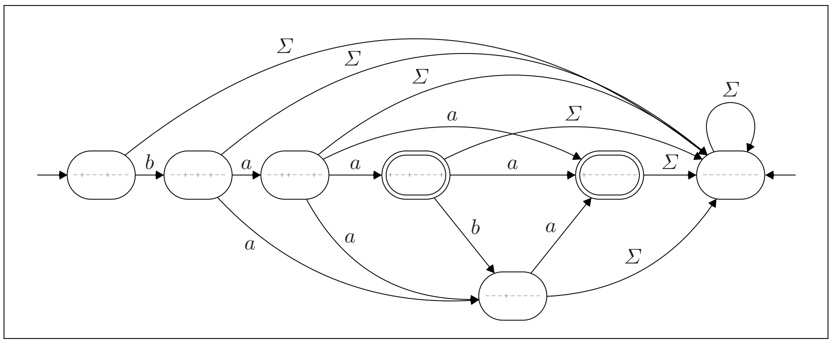

Fig. 26. Automaton $\mathcal{A}_9$ for table $\mathcal{T}_9$

$\mathcal{T}_0$	ε
$*\ \varepsilon$	$-$
$*\ b$	$-$
$*\ a$	$-$

$\Rightarrow^{1.}_{ce}$

$\mathcal{T}_1$	ε	baa	aa	a
$*\ \varepsilon$	$-$	$+$	$-$	$-$
$*\ b$	$-$	$-$	$+$	$-$
$*\ a$	$-$	$-$	$-$	$-$

$\Rightarrow^{2.}_{ncl}$

$\mathcal{T}_2$	ε	baa	aa	a
$*\ \varepsilon$	$-$	$+$	$-$	$-$
$*\ b$	$-$	$-$	$+$	$-$
$*\ a$	$-$	$-$	$-$	$-$
$*\ bb$	$-$	$-$	$-$	$-$
$*\ ba$	$-$	$-$	$+$	$+$

$\Rightarrow^{3.}_{ncl}$

$\mathcal{T}_3$	ε	baa	aa	a
$*\ \varepsilon$	$-$	$+$	$-$	$-$
$*\ b$	$-$	$-$	$+$	$-$
$*\ a$	$-$	$-$	$-$	$-$
$*\ bb$	$-$	$-$	$-$	$-$
$*\ ba$	$-$	$-$	$+$	$+$
$*\ ab$	$-$	$-$	$-$	$-$
$*\ aa$	$-$	$-$	$-$	$-$

$\Rightarrow^{4.}_{ncl}$

$\mathcal{T}_4$	ε	baa	aa	a
$*\ \varepsilon$	$-$	$+$	$-$	$-$
$*\ b$	$-$	$-$	$+$	$-$
$*\ a$	$-$	$-$	$-$	$-$
$*\ ba$	$-$	$-$	$+$	$+$
$*\ bb$	$-$	$-$	$-$	$-$
$*\ ab$	$-$	$-$	$-$	$-$
$*\ aa$	$-$	$-$	$-$	$-$
$*\ bab$	$-$	$-$	$-$	$-$
$*\ baa$	$+$	$-$	$-$	$+$

$\Rightarrow^{5.}_{ncl}$

$\mathcal{T}_5$	ε	baa	aa	a
$*\ \varepsilon$	$-$	$+$	$-$	$-$
$*\ b$	$-$	$-$	$+$	$-$
$*\ a$	$-$	$-$	$-$	$-$
ba	$-$	$-$	$+$	$+$
baa	$+$	$-$	$-$	$+$
$*\ bb$	$-$	$-$	$-$	$-$
$*\ ab$	$-$	$-$	$-$	$-$
$*\ aa$	$-$	$-$	$-$	$-$
$*\ bab$	$-$	$-$	$-$	$-$
$*\ baab$	$-$	$-$	$-$	$+$
$*\ baaa$	$+$	$-$	$-$	$-$

$\Rightarrow^{6.}_{ncl}$

$\mathcal{T}_6$	ε	baa	aa	a
$*\ \varepsilon$	$-$	$+$	$-$	$-$
$*\ b$	$-$	$-$	$+$	$-$
$*\ a$	$-$	$-$	$-$	$-$
ba	$-$	$-$	$+$	$+$
baa	$+$	$-$	$-$	$+$
$*\ baab$	$-$	$-$	$-$	$+$
$*\ bb$	$-$	$-$	$-$	$-$
$*\ ab$	$-$	$-$	$-$	$-$
$*\ aa$	$-$	$-$	$-$	$-$
$*\ bab$	$-$	$-$	$-$	$-$
$*\ baaa$	$+$	$-$	$-$	$-$
$*\ baabb$	$-$	$-$	$-$	$-$
$*\ baaba$	$+$	$-$	$-$	$-$

$\Rightarrow^{7.}_{ncl}$

$\mathcal{T}_7$	ε	baa	aa	a
$*\ \varepsilon$	$-$	$+$	$-$	$-$
$*\ b$	$-$	$-$	$+$	$-$
$*\ a$	$-$	$-$	$-$	$-$
ba	$-$	$-$	$+$	$+$
baa	$+$	$-$	$-$	$+$
$*\ baab$	$-$	$-$	$-$	$+$
$*\ baaa$	$+$	$-$	$-$	$-$
$*\ bb$	$-$	$-$	$-$	$-$
$*\ ab$	$-$	$-$	$-$	$-$
$*\ aa$	$-$	$-$	$-$	$-$
$*\ bab$	$-$	$-$	$-$	$-$
$*\ baabb$	$-$	$-$	$-$	$-$
$*\ baaba$	$+$	$-$	$-$	$-$
$*\ baaab$	$-$	$-$	$-$	$-$
$*\ baaaa$	$-$	$-$	$-$	$-$

$\Rightarrow^{8.}_{ncs}$

$\mathcal{T}_8$	ε	baa	aa	a	aaa
$*\ \varepsilon$	$-$	$+$	$-$	$-$	$-$
$*\ b$	$-$	$-$	$+$	$-$	$+$
$*\ a$	$-$	$-$	$-$	$-$	$-$
$*\ ba$	$-$	$-$	$+$	$+$	$-$
baa	$+$	$-$	$-$	$+$	$-$
$*\ baab$	$-$	$-$	$-$	$+$	$-$
$*\ baaa$	$+$	$-$	$-$	$-$	$-$
$*\ bb$	$-$	$-$	$-$	$-$	$-$
$*\ ab$	$-$	$-$	$-$	$-$	$-$
$*\ aa$	$-$	$-$	$-$	$-$	$-$
$*\ bab$	$-$	$-$	$-$	$-$	$-$
$*\ baabb$	$-$	$-$	$-$	$-$	$-$
$*\ baaba$	$+$	$-$	$-$	$-$	$-$
$*\ baaab$	$-$	$-$	$-$	$-$	$-$
$*\ baaaa$	$-$	$-$	$-$	$-$	$-$

$\Rightarrow^{9.}_{ce}$

$\mathcal{T}_9$	ε	baa	aa	a	aaa	$baaba$	$aaba$	aba	ba
$*\ \varepsilon$	$-$	$+$	$-$	$-$	$-$	$+$	$-$	$-$	$-$
$*\ b$	$-$	$-$	$+$	$-$	$+$	$-$	$+$	$-$	$-$
$*\ a$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
$*\ ba$	$-$	$-$	$+$	$+$	$-$	$-$	$-$	$+$	$-$
$*\ baa$	$+$	$-$	$-$	$+$	$-$	$-$	$-$	$-$	$+$
$*\ baab$	$-$	$-$	$-$	$+$	$-$	$-$	$-$	$-$	$-$
$*\ baaa$	$+$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
$*\ bb$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
$*\ ab$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
$*\ aa$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
$*\ bab$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
$*\ baabb$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
$*\ baaba$	$+$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
$*\ baaab$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$
$*\ baaaa$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$	$-$

	l_{up}	l	p	i
$\mathcal{T}_0$	1	1	1	0
$\Rightarrow_{ce}\ \mathcal{T}_1$	1	3	3	2
$\Rightarrow_{ncl}\ \mathcal{T}_2$	2	4	4	4
$\Rightarrow_{ncl}\ \mathcal{T}_3$	3	4	4	4
$\Rightarrow_{ncl}\ \mathcal{T}_4$	4	5	5	5
$\Rightarrow_{ncl}\ \mathcal{T}_5$	5	7	5	10
$\Rightarrow_{ncl}\ \mathcal{T}_6$	6	7	5	10
$\Rightarrow_{ncl}\ \mathcal{T}_7$	7	7	5	10
$\Rightarrow_{ncs}\ \mathcal{T}_8$	7	7	6	9
$\Rightarrow_{ce}\ \mathcal{T}_9$	7	7	7	9

1) found counterexample: *baa* for current model $\mathcal{A}_0$ (based on $\mathcal{T}_0$).
2) closedness violation: Trying to make $\mathcal{T}_1$ closed with row: b ($[-,-,+,-]$)
3) closedness violation: Trying to make $\mathcal{T}_2$ closed with row: a ($[-,-,-,-]$)
4) closedness violation: Trying to make $\mathcal{T}_3$ closed with row: ba ($[-,-,+,+]$)
5) closedness violation: Trying to make $\mathcal{T}_4$ closed with row: baa ($[+,-,-,+]$)
6) closedness violation: Trying to make $\mathcal{T}_5$ closed with row: $baab$ ($[-,-,-,+]$)
7) closedness violation: Trying to make $\mathcal{T}_6$ closed with row: $baaa$ ($[+,-,-,-]$)
8) consistency (1) violation: Trying to obtain consistency (1) for $\mathcal{T}_7$ by adding suffix: *aaa*
9) found counterexample: *baaba* for current model $\mathcal{A}_8$ (based on $\mathcal{T}_8$) (cf. Figure 25).
10) Table $\mathcal{T}_9$ is closed and consistent. Final model $\mathcal{A}_9$ can be calculated (cf. Figure 26).

Table 13. An NL* run exemplifying the termination proof