Learning and Characterizing Fully-Ordered Lattice Automata

Dana Fisman and Sagi Saadon[⋆]

Ben-Gurion University

Abstract. Traditional automata classify words from a given alphabet as either *good* or *bad*. In many scenarios, in particular in formal verification, a finer classification is required. Fully-ordered lattice automata (FOLA) associate with every possible word a value from a finite set of values such as $\{0, 1, 2, \ldots, k\}$. In this paper we are interested in learning formal series that can be represented by FOLA. Such a series can be learned by a straight forward extension of the $\mathbf{L}^*$ algorithm. However, this approach does not take advantage of the special structure of a FOLA. In this paper we investigate FOLAs and provide a Myhill-Nerode characterization for FOLAs, which serves as a basis for providing a specialized algorithm for FOLAs, which we term $\mathbf{FOL}^*$. We compare the performance of $\mathbf{FOL}^*$ to that of $\mathbf{L}^*$ on synthetically generated FOLA. Our experiments show that $\mathbf{FOL}^*$ outperforms $\mathbf{L}^*$in the number of states of the obtained FOLA, the number of issued *value queries* (the extension of *membership queries* to the quantitative setting), and the number of issued *equivalence queries*.

1 Introduction

Automata, being a simple computational model on which many operations (such as union, intersection, complementation, emptiness, equivalence) can be efficiently computed, have found usages in many applications including pattern matching, syntax analysis, and formal verification. Traditional automata are *Boolean* in the sense that they associate with any given word one of two possible values. In many applications, such as biology, physics, cognitive sciences, control, and linguistics, it is desired to associate with any given word one of many possible values. These motivated the study of richer types of automata such as *weighted automata* in which a word is associated with a value from a given semiring over a large range of values [22].

Focusing on formal verification, of particular interests are semirings that form a (distributive) *lattice*. A lattice $\mathcal{L} = \langle A, \leq \rangle$ is a partially ordered set in which every two elements $a, b \in A$ have a least upper bound (a *join* b) and a greatest lower bound (a *meet* b). Lattices offer generalization for multi-valued logics, and as such arise in quantitative verification [12,15,18,6,21], abstraction methods [13], query checking [10,16], and verification under inconsistent view-points [29,17].

In recent years, *model learning* emerged as a useful technique in formal verification [30]. Model learning, roughly speaking, refers to the task of learning a black-box system,

[⋆] This research was partially supported by the Israel Science Foundation (ISF) grant 2507/21.

implemented by some automaton, by querying it with sequences of input and observing the received sequences of outputs. Model learning can be achieved using learning algorithms that ask membership and equivalence queries, as does the classical $\mathbf{L}^*$ algorithm developed by Angluin for learning regular languages represented by DFAs [2]. To this aim, the verification community seeks for query leaning algorithms for the automata types in use. Angluin-style algorithms have been developed for many automata types such as tree-automata [27], non-deterministic and alternating finite automata [9,4], Mealy machines [28], I/O-Automata [1], modular visibly pushdown automata [19], ω-automata [5], symbolic automata [11], strongly unambiguous Büchi automata [3], and structurally unambiguous probabilistic grammars [25].

In this work we are interested in learning fully-ordered lattice automata. A *fully-ordered lattice automata* (FOLA) is a lattice automata over a fully-ordered set $\{0, 1, \ldots, k\}$ where $\min$ and $\max$ are the meet and join operations, respectively. Roughly speaking a FOLA extends a DFA by annotating the transitions and states with values from the given lattice, as shown for instance in Fig.1. The value the FOLA gives an input word is computed as the meet of all the lattice values read along the run as well as the lattice value of the final state (a formal definition is provided in Sec.2). Thus the FOLA $\mathcal{A}$ of Fig.1 gives the word b the value $2 \wedge 1 = 1$, and the word ba the value $2 \wedge 2 \wedge 2 = 2$.

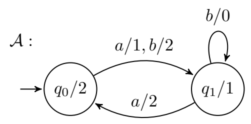

Fig. 1: A FOLA $\mathcal{A}$

We consider the active learning setting in which the algorithm can use *value queries* (VQ) (the extension of *membership queries* to the quantitative case) and *equivalence queries* (EQ). We focus on FOLAs, since besides nicely modeling multi-valued logics, they posses a polynomial minimization algorithm, while the minimization problem for general lattice automata is NP-complete [14]. Thus, assuming P$\neq$NP general lattice automata cannot be polynomially learned, since the learning algorithm can act as a minimization procedure.

In a FOLA, both transitions and states are annotated with values l from the given lattice $\mathcal{L}$. The value the automaton provides for a word depends on both the values traversed during the run, and the value of the state at the end of the run. If all transition values are the maximal value (thus do not affect the final value), the FOLA, is said to be a *simple* FOLA, abbreviated, SFOLA. A FOLA of size n over a lattice $\mathcal{L}$ can be simulated by an SFOLA of size $n \times |\mathcal{L}|$. This blowup is tight in the sense that there exists a family of languages $\{L_n\}_{n\in\mathbb{N}}$ over $\Sigma = \{a, b\}$ and lattice of size n which can be implemented by a FOLA with n states but there is no SFOLA with less than $n \times (n - 1)$ states. A FOLA for L_n is provided in Fig.2 and an equivalent SFOLA is given in Fig.10 in App.A.[1]

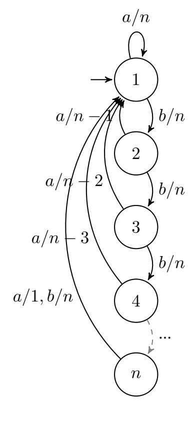

Fig. 2: A FOLA for L_n.

[1] The reason for the quadratic blowup can be understood by noticing that states of the SFOLA are required to record the traversal value up to that state (since all transitions values are $\top$), and for any $l \in \{2, 3, \ldots, n\}$ the FOLA needs to check whether an a will follow n consecutive b's (which requires of course n states), so in total $(n - 1) \times n$ states are required.

SFOLAs can be polynomially learned using a straight forward extension of the $\mathbf{L}^*$ algorithm.[2] It is therefore desired that the developed algorithm for FOLAs would perform better. However, we cannot expect the algorithm to have better worst-case complexity analysis since there are families of languages for which the minimal size of a FOLA is the same as the minimal size of a SFOLA. For instance this is the case for the family $\{L'_n\}_{n\in\mathbb{N}}$ over lattice $\mathcal{L} = \{0,\ldots,n-1\}$ which returns the size of the word modulo n. We do, however, expect a specialized algorithm can take into account the special structure of FOLAs and work better in practice.

In [8] an algorithm for learning *multiplicity automata*, an algebraic generalization of automata, that works with respect to a given *field* was developed. This algorithm was deployed by [7] to learn weighted automata, under the assumption that the semiring is a field. While a (distributive) lattice is a special case of a semiring, a lattice is not a field, since the property that every element $a \in A$ has an additive (and/or multiplicative) inverse may not hold. It follows that the algorithm for learning multiplicity automata cannot be deployed for learning lattice automata. FOLAs can be learned using a learning algorithm for automata based on a monoid action, called writer automata in [31], however the complexity of this algorithm for the case of FOLA is the same as using the extension of $\mathbf{L}^*$ to learn SFOLAs.

In order to obtain an algorithm that in practice would perform better on FOLAs than $\mathbf{L}^*$ for SFOLA, we must understand FOLAs better. To this aim, building on the work of Halamish and Kupferman [14] who studied minimization of FOLAs, we reveal: an equivalence relation for FOLAs; a canonical minimal FOLA; and a respective Myhill-Nerode characterization for FOLAs. Sec.3 is devoted for this investigation.

The provided characterization and insights allow us to design a specialized algorithm for FOLAs; this is the topic of Sec.4. Sec.5 compares the performance of $\mathbf{FOL}^*$ with that of the $\mathbf{L}^*$ algorithm on synthetically generated FOLAs. The experiments shows a clear advantage to our algorithm, with up to an $|\mathcal{L}|$ blowup. Sec.6 concludes. Due to space restrictions some proofs are deferred to the appendix of the full version.

2 Preliminaries

Words, Languages, Formal Series We use Σ for an *alphabet* i.e. a finite non-empty set of symbols. The set of word over Σ is denoted Σ^*. The length of a word $w = \sigma_1\sigma_2\ldots\sigma_m$, denoted $|w|$ is m. The prefix of w up to position i, namely $\sigma_1\sigma_2\ldots\sigma_i$, is denoted $w[..i]$. Similarly the suffix of w starting at position i, namely $\sigma_i\sigma_{i+1}\ldots\sigma_m$, is denoted $w[i..]$. A *language* is a subset of Σ^*. A *formal series* f is a function $f : \Sigma^* \to A$ mapping each word to a value in A, where A is some set. Such a formal series f is sometimes called an A-language. Note that a language is a special case of a formal series. That is $L \subseteq \Sigma^*$ can be thought of as a formal series $f_L : \Sigma^* \to \mathbb{B}$ where $\mathbb{B} = \{0,1\}$. In this work we are interested in formal series that map words to a value in a fully ordered set $\{0,1,\ldots,k\}$.

[2] In this extension, the observation table matrix holds values in the lattice instead of $\{0,1\}$, that is, the entry (i,j) holds the result of the value query for the word $s_i \cdot e_j$ where s_i is the title of row i and e_j the title of row j. Two rows in the observation table are considered equivalent if they are exactly the same. These are the only changes required w.r.t. to $\mathbf{L}^*$ for DFAs.

Lattice Let $\mathcal{L} = \langle A, \leq \rangle$ be a partially ordered set. An element $a \in A$ is an *upper bound* on A (denoted $\top$) if $b \leq a$ for all $b \in A$. An element $a \in A$ is a *lower bound* on A (denoted $\bot$) if $a \leq b$ for all $b \in A$. A partially (or fully) ordered set $\langle A, \leq \rangle$ is a *Lattice* if for every two elements $a, b \in A$ both the least upper bound, denoted as $a \vee b$, and the greatest lower bound, denoted as $a \wedge b$, of $\{a, b\}$ exist. A lattice is *complete* if for every subset $A' \subseteq A$ the least upper bound and the greatest upper bound exist. In a complete lattice $\top$ denotes the join of all elements in A and $\bot$ denotes their meet.

Lattice Automata Lattice automata are a generalization of finite-state automata [20]. Their deterministic version is defined as follows. A *deterministic lattice automaton* (LDFA) $\mathcal{A}$ is a tuple $\langle \mathcal{L}, \Sigma, Q, q_0, \delta, \eta, F \rangle$ where $\mathcal{L}$ is a complete lattice; Σ is the alphabet; Q is a finite set of states; $q_0 \in Q$ is the initial state; $\delta : Q \times \Sigma \to Q$ is the state transition function; $\eta : Q \times \Sigma \to \mathcal{L}$ is the transitions value function associating with every transition (from a state q on letter σ) a value ℓ from the lattice; and $F : Q \to \mathcal{L}$ is the state-value function associating with each state a value form the lattice.

A run of $\mathcal{A}$ on a word $w = \sigma_1 \sigma_2 \cdots \sigma_n$ is a sequence $r = q_0 \ldots q_n$ of $n + 1$ states. The *traversal value* of r on w, denoted $\mathit{trvl}(w)$ is the meet of all transitions involved, i.e., if $\eta(q_{i-1}, \sigma_i) = \ell_i$ then $\mathit{trvl}(w) = \bigwedge_{i=1}^{n} \ell_i$. The value of r on w is defined as $\mathit{val}(w) = \mathit{trvl}(w) \wedge F(q_n)$. Namely it is the meet of the traversal value and the state-value of the last state of the run.[3] The extension of δ from letters to words is denoted δ^* (i.e., $\delta^*(q, \epsilon) = q$, and $\delta^*(q, u\sigma) = \delta(\delta^*(q, u), \sigma)$ for $u \in \Sigma^*$ and $\sigma \in \Sigma$). The formal series defined by $\mathcal{A}$ is denoted $[\![\mathcal{A}]\!]$, and $[\![\mathcal{A}]\!](w)$ denotes the value $\mathcal{A}$ gives to word w.

A *fully-ordered lattice automaton* (FOLA) is a lattice automaton over a fully-ordered set $\{0, 1, \ldots, k\}$ where $\min$ and $\max$ are the meet and join operations, respectively.

Example 1. Recall the FOLA $\mathcal{A}$ over the lattice $\mathcal{L} = \{0, 1, 2\}$ and the alphabet $\Sigma = \{a, b\}$ from Fig.1. Consider the word $w = baa$. The run of $\mathcal{A}$ on w is the sequence $\rho = q_0 q_1 q_0 q_1$. Its traversal-value is $\mathit{trvl}(w) = \eta(q_0, b) \wedge \eta(q_1, a) \wedge \eta(q_0, a) = 2 \wedge 2 \wedge 1$. The value of $\mathcal{A}$ on w is $\mathit{val}(w) = \mathit{trvl}(w) \wedge F(q_1) = 1 \wedge 1 = 1$.

3 A Myhill-Nerode Characterization for FOLAs

For a language $L \subseteq \Sigma^*$, one defines the equivalence relation $\equiv_L \subseteq \Sigma^* \times \Sigma^*$ as follows $x \equiv_L y$ iff for every $z \in \Sigma^*$ it holds that $xz \in L \iff yz \in L$. The celebrated Myhill-Nerode theorem states that (i) L is a regular iff $\equiv_L$ has a finite index (i.e. $\equiv_L$ induces a finite number of equivalence classes), (ii) there is a one-to-one relation between the states of a minimal DFA for L and the equivalence classes of $\equiv_L$, and (iii) all DFAs with a minimal number of states are isomorphic to each other, or put otherwise there is a unique minimal DFA [23,24]. Many automata learning algorithms, including $\mathbf{L}^*$, rely on the correspondence between the equivalence classes and the states of the minimal representation. Therefore, we seek for a similar correspondence between an adequate equivalence relation for formal-series defined by FOLAs and minimal FOLAs.

[3] In non-deterministic lattice automata, there may be several runs on a given word, and each run may have a different value. In this case the value of the automaton on the word is the join of the values of all of its runs on that word.

3.1 No unique minimal FOLA

We first note that unlike the situation in regular languages, for formal series represented by FOLAs there may exists two FOLAs with a minimal number of states that are not isomorphic to each other. Fig.3 depicts two minimal distinct FOLAs, $\mathcal{A}_1$ and $\mathcal{A}_2$, implementing the formal-series $f : \{a\}^* \to \{0, 1\}$ that gives 1 iff the length of the word is at most one.

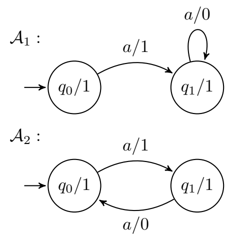

Fig. 3: Two minimal FOLAs for the same formal series.

Let us examine this closely. Let $\mathcal{A} = \langle \mathcal{L}, \Sigma, Q, q_0, \delta, \eta, F\rangle$ be a FOLA. It induces an equivalence relation $\equiv_{\mathcal{A}}$ between pairs of words, defined as follows. For $x, y \in \Sigma^*$ we have $x \equiv_{\mathcal{A}} y$ iff the run of $\mathcal{A}$ on x ends in the same state as the run of $\mathcal{A}$ on y. In the case of regular languages, if $\mathcal{A}_1$ and $\mathcal{A}_2$ are two minimal DFAs for the same language L, then $\equiv_{\mathcal{A}_1}$ and $\equiv_{\mathcal{A}_2}$ are exactly the same relation as $\equiv_L$.

Fig.3 shows that in the case of languages accepted by FOLAs, this is not necessarily the case. Indeed, while the FOLAs $\mathcal{A}_1$ and $\mathcal{A}_2$ define the same function, and are both minimal in the number of states, the induced equivalence relations are different: for $\equiv_{\mathcal{A}_1}$ we have $E_0 = \{\epsilon\}$ and $E_1 = \Sigma^+$, whereas for $\equiv_{\mathcal{A}_2}$ we have $E_0 = \{w : |w| \mod 2 = 0\}$ and $E_1 = \{w : |w| \mod 2 \neq 0\}$ where E_i describes the equivalence class of state q_i.

3.2 Difficulties in defining $\equiv_f$

Investigating minimization of FOLAs, Halamish and Kupferman [14] explain the difficulty in finding an equivalence relation for FOLAs. Their first observation is that the natural extension $x \equiv^1_f y$ iff for every $z \in \Sigma^*$ it holds that $f(xz) = f(yz)$ is too refined, as for the FOLA $\mathcal{B}$ over $\Sigma = \{a, b, c\}$ and $\mathcal{L} = \{0, 1, 2\}$ depicted in Fig.4 it will consist of three equivalence classes, while one suffices. Yet, this definition holds under the assumption that all transition values are $\top$. As mentioned earlier, FOLAs admitting this restriction are called *simple FOLAs* or in short SFOLAs.

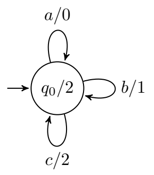

Fig. 4: A FOLA $\mathcal{B}$

Their second observation concerns the following definition $x \equiv^2_f y$ which states that $x \equiv^2_f y$ iff for every $z \in \Sigma^*$ exists $\ell_z \in \mathcal{L}$ such that $f(xz) = f(x) \wedge \ell_z$ and $f(yz) = f(y) \wedge \ell_z$. This definition seems intuitive for FOLAs for which all acceptance values are $\top$, however, it does not work in this case as well. The main problem with this definition is that it is not transitive and thus it is not an equivalence relation, as shown in Exmp.2.

Example 2 ([14]). To see that transitivity does not hold for $\equiv^2_f$ consider the FOLA $\mathcal{C}$ given in Fig.5. It defines a function from words over $\Sigma = \{a, b, c, \#\}^*$ to values in lattice $\mathcal{L} = \{0, 1, 2, 3\}$. The proposed definition satisfies that $a \equiv^2_f b$ and $b \equiv^2_f c$ yet $a \not\equiv^2_f c$. Indeed the FOLAs $\mathcal{C}_1$ and $\mathcal{C}_2$, depicted in Fig.5, are equivalent to $\mathcal{C}$ and each makes a different choice regarding equivalence of the string b.

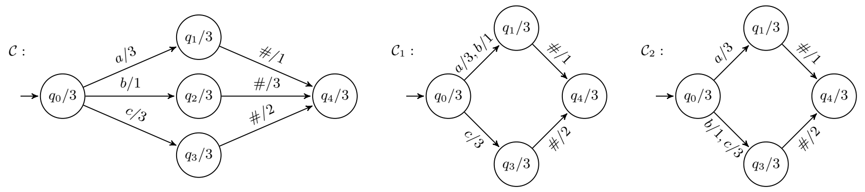

Fig. 5: The definition $\equiv_f^2$ breaks transitivity [14]

3.3 Defining the Equivalence Relation

The relation $\equiv_L$ for a regular language L captures that two words reach the same state of the minimal DFA. The relation $\equiv_f$ for a formal series f should capture that two words reach the same state of a minimal FOLA. To define it we make use of the notion of a Hankel Matrix. With every formal series $f : \Sigma^* \to \mathcal{L}$ we can associate its *Hankel Matrix* $\mathcal{H}_f$. The Hankel Matrix has infinitely many rows and infinitely many columns. The entry (i, j) has the value $f(w_i \cdot w_j)$ where w_i and w_j are the i-th and j-th words in an agreed enumeration of Σ^*. Consider the Hankel Matrix for a regular language L and two words w_1, w_2. The rows of w_1 and w_2 in $\mathcal{H}_L$ are exactly the same iff $w_1 \equiv_L w_2$. This is since if $w_1 \not\equiv_L w_2$ then there exists a word $z \in \Sigma^*$ s.t. $w_1 z \in L$ and $w_2 z \notin L$ or vice versa, thus $\mathcal{H}_L(w_1, z) \neq \mathcal{H}_L(w_2, z)$. To define $\equiv_f$ we need to understand how do two rows of words w_1 and w_2 resemble if w_1 and w_2 reach the same state of a minimal FOLA. Clearly they need not be exactly the same, since $f(w_1 z)$ relies also on the values traversed while reading w_1.

We use the term *observation table* for any sub-matrix of $\mathcal{H}_f$. Two subsets S and E of Σ^* define the observation table $\mathcal{T} = (S, E, T)$ where $T : S \times E \to \mathcal{L}$ is defined as $T(s, e) = f(s \cdot e)$ for every $s \in S$ and $e \in E$. We will define relations for an arbitrary observation table; when applied to the full Hankel Matrix, it will convey the desired equivalence relation. The algorithm will use the definitions for a finite observation table.

We say that the *row-potential* of a row s (or simply its *potential*) is l if there exists a column $e \in E$ such that $T(s, e) = l$ and there is no $e' \in E$ such that $T(s, e') > l$. This means that the traversal value of the correct automaton on reading s cannot be smaller than l as otherwise for no extension the value l can be obtained. However, according to the observed data, there is no reason to assign it a value greater than l.

Definition 3 (Row Potential). *Let $\mathcal{T} = (S, E, T)$ be an observation table, and $s \in S$. The row-potential of s, denoted $\text{pot}_{\mathcal{T}}(s)$, is $\max\{T(s \cdot e) : e \in E\}$.*

For every value $l \in \mathcal{L}$ and every pair of rows whose potential is at least l we would like to ask whether they should be distinguished according to the data. The following definitions make this precise.

Definition 4 ($\not\sim^l_{\mathcal{T}}$, $\not\approx^l_{\mathcal{T}}$, $\not\approx_{\mathcal{T}}$). *Let $l \in \mathcal{L}$ and $s, s' \in S$.*

1. *We use $s \not\sim^l_{\mathcal{T}} s'$ if $\text{pot}_{\mathcal{T}}(s) \geq l$, $\text{pot}_{\mathcal{T}}(s') \geq l$ and $\exists e \in E$ s.t. $T(s, e) \geq l$ and $T(s', e) < l$ or vice versa.*
2. *We use $s \not\approx^l_{\mathcal{T}} s'$ if for some $l' \leq l$ we have $s \not\sim^{l'}_{\mathcal{T}} s'$.*
3. *We use $s \not\approx_{\mathcal{T}} s'$ if $s \not\sim^l_{\mathcal{T}} s'$ for some $l \in \mathcal{L}$.*

It is easy to see that $x \approx^l_{\mathcal{T}} y$ implies $x \approx^{l-1}_{\mathcal{T}} y$ and that $x \approx_{\mathcal{T}} y$ iff $x \approx^k_{\mathcal{T}} y$ where $\mathcal{L} = \{0, 1, \dots, k\}$.

The following claim states that if we have two rows s_1 and s_2 such that in one column e the entry for $T(s_1, e)$ is strictly bigger than $T(s_2, e)$ whereas in another column e' the entry for $T(s_1, e')$ is strictly smaller than $T(s_2, e')$, then $s_1 \not\approx_{\mathcal{T}} s_2$.

Claim 5 *Let $\mathcal{T} = (S, E, T)$. Let $s_1, s_2 \in S$ and $e_1, e_2 \in E$. If $T(s_1, e_1) < T(s_2, e_1)$ while $T(s_1, e_2) > T(s_2, e_2)$ then $s_1 \not\approx_{\mathcal{T}} s_2$.*

We claim that if $s \not\approx_{\mathcal{T}} s'$ then strings s and s' cannot reach the same state of a FOLA for the respective formal series.

Lemma 6. *Let $\mathcal{T} = (S, E, T)$ be an observation table for formal series f, and let $s, s' \in S$. If $s \not\approx_{\mathcal{T}} s'$ then in no FOLA for f the words s, s' reach the same state.*

Proof. From $s \not\approx_{\mathcal{T}} s'$ it follows that exists $l \in \mathcal{L}$ such that $s \not\approx^l_{\mathcal{T}} s'$. From the definition of $\not\approx^l_{\mathcal{T}}$ it follows that *pot*$_{\mathcal{T}}(s) \geq l$, *pot*$_{\mathcal{T}}(s') \geq l$ and $\exists e \in E$ s.t. $T(s, e) \geq l$ and $T(s', e) < l$ or vice versa. Let $\mathcal{A} = \langle \mathcal{L}, \Sigma, Q, q_0, \delta, \eta, F \rangle$ be a FOLA for f. The traversal value of s in $\mathcal{A}$ must be at least l, as otherwise for every z, $\mathcal{A}(sz) < l$ but *pot*$_{\mathcal{T}}(s) \geq l$ implies there exists a $z \in E$ for which $T(s, z) \geq l$ so $\mathcal{A}_f$ disagrees with $\mathcal{T}$. The same argument shows that the traversal value of s' in $\mathcal{A}$ must be at least l. Assume towards contradiction that $\mathcal{A}$ upon reading s or s' reaches the same state q_s. Let q_e be the state that $\mathcal{A}$ reaches upon reading se (or $s'e$ as this must be the same state). The traversal value of e starting from the state q must be at least l and $F(q_e)$ must be at least l as otherwise $\mathcal{A}$ will be wrong regarding $s \cdot e$. But if this is the case then $\mathcal{A}$ is wrong regarding $s' \cdot e$. Contradiction. □

While the relation $\approx_{\mathcal{T}}$ differentiates words that do not reach the same state, it is not an equivalence relation. The reason is that it does not satisfy the transitivity requirement as shown by Fig.6. The following claim will help us strengthen it to get the desired equivalence relation.

T	e_1	e_2	e_3
s_1	1	2	1
s_2	1	1	1
s_3	2	1	2

Fig. 6: $s_1 \approx_{\mathcal{T}} s_2$ and $s_2 \approx_{\mathcal{T}} s_3$ but $s_1 \not\approx_{\mathcal{T}} s_3$

Claim 7 *Let $s_1, s_2, s_3 \in S$. If $s_1 \approx_{\mathcal{T}} s_2, s_2 \approx_{\mathcal{T}} s_3$, $s_2 \succcurlyeq_{\mathcal{T}} s_1$ and $s_2 \succcurlyeq_{\mathcal{T}} s_3$ then $s_1 \approx_{\mathcal{T}} s_3$.*

If we would like to pick one of a set of non-distinguishable words to be a representative, following claim 7 it makes sense to choose one with the highest potential. Since there could be several such, we define an order between two rows in the table. We use the *shortlex* order between strings, denoted $\leq_{slex}$.[4]

Definition 8 (Rows order). *Let $\mathcal{T} = (S, E, T)$ be an observation table and s, s' rows in S. We say that $s \succcurlyeq_{\mathcal{T}} s'$ if either [pot$_{\mathcal{T}}(s) \geq$ pot$_{\mathcal{T}}(s')$] or [pot$_{\mathcal{T}}(s) =$ pot$_{\mathcal{T}}(s')$ and $s \leq_{slex} s'$] (where $\leq_{slex}$ is the shortlex order).*

The representative for a set $S' \subseteq S$ of rows that cannot be distinguished from one another is chosen to be the minimal element in the shortlex order, among those in S' with the highest potential. That is, the set of representatives of an observation table $\mathcal{T}$ is defined as follows.

[4] The *shortlex* order (aka the *length-lexicographic* order) stipulates that string w_1 is smaller than string w_2, denoted $w_1 <_{slex} w_2$ if $|w_1| < |w_2|$ or $|w_1| = |w_2|$ and w_1 precedes w_2 in the lexicographic order.

Definition 9 **($\mathit{reps}(\mathcal{T})$, $\mathit{rep}_{\mathcal{T}}(s)$, $\equiv_{\mathcal{T}}$).** *Let $\mathcal{T} = (S, E, T)$ be an observation table.*

- *The set of* representatives *of the table is defined as $\mathit{reps}(\mathcal{T}) = \{s \in S \mid \forall s' \approx_{\mathcal{T}} s.\ \ s \not\succ_{\mathcal{T}} s'\}$.*
- *For a row $s \in S$ we use $\mathit{rep}_{\mathcal{T}}(s)$ for the row $s_* \in \mathit{reps}(\mathcal{T})$ such that $s \approx_{\mathcal{T}} s_*$ and for every $s' \in \mathit{reps}(\mathcal{T})$ satisfying $s' \approx_{\mathcal{T}} s$ we have $s_* \not\succ_{\mathcal{T}} s'$.*
- *Let s, s' be rows in S. We use $s \equiv_{\mathcal{T}} s'$ to denote that $\mathit{rep}_{\mathcal{T}}(s) = \mathit{rep}_{\mathcal{T}}(s')$. That is, two rows are equivalent if they have the same representative.*

Given a formal series $f : \Sigma^* \to \mathcal{L}$ let $\mathcal{T}_f = (\Sigma^*, \Sigma^*, T_f)$ be the Hankel Matrix for f. Let $\mathit{reps}(f)$, $\mathit{rep}_f(w)$ and $\mathit{pot}_f(w)$ abbreviate $\mathit{reps}(\mathcal{T}_f)$, $\mathit{rep}_{\mathcal{T}_f}(w)$ and $\mathit{pot}_{\mathcal{T}_f}(w)$. Likewise, let $\sim^l_f$, $\approx^l_f$, $\approx_f$, and $\equiv_f$ abbreviate $\sim^l_{\mathcal{T}_f}$, $\approx^l_{\mathcal{T}_f}$, $\approx_{\mathcal{T}_f}$, and $\equiv_{\mathcal{T}_f}$.

We show that $\equiv_f$ is an equivalence relation on Σ^* and a right congruence relation.

Claim 10 *The relation $\equiv_f$ is an equivalence relation.*

Claim 11 *The relation $\equiv_f$ is a right congruence relation. That is, $x \equiv_f y$ implies $xz \equiv_f yz$ for all $z \in \Sigma^*$.*

Note that if s_* is the representative of s, then for every $e \in E$ we have that $\mathcal{T}(se) \leq \mathcal{T}(s_*e)$ and more precisely $\mathcal{T}(se) = \mathcal{T}(s_*e) \wedge \mathit{pot}_{\mathcal{T}}(s)$.

Claim 12 *Let $\mathcal{T} = (S, E, T)$ be an observation table, $s \in S$ and $s_* = \mathit{rep}_{\mathcal{T}}(s)$. Then for all $e \in E$ (i) $T(s, e) \leq T(s_*, e)$ and moreover (ii) $T(s, e) = T(s_*, e) \wedge \mathit{pot}_{\mathcal{T}}(s)$.*

Proof. Assume towards contradiction that $\exists e \in E$ s.t. $T(s, e) > T(s_*, e)$. Assume $T(s, e) = l$. Then $T(s_*, e) < l \leq \mathit{pot}_{\mathcal{T}}(s) \leq \mathit{pot}_{\mathcal{T}}(s_*)$. Therefore, according to Def.4, $s \not\sim^l_{\mathcal{T}} s_*$ which contradicts that s_* is the representative of s (Def.9). This proves item (i).

For item (ii), assume toward contradiction that $\exists e \in E$ for which $\mathcal{T}(se) \neq \mathcal{T}(s_*e) \wedge \mathit{pot}_{\mathcal{T}}(s)$. It is clear that $\mathcal{T}(se) \leq \mathit{pot}_{\mathcal{T}}(s)$ and from item (i) we know that $\mathcal{T}(se) \leq \mathcal{T}(s_*e)$. Applying these conclusions, we get that $\mathcal{T}(se) < \mathcal{T}(s_*e) \wedge \mathit{pot}_{\mathcal{T}}(s)$, which implies that $\mathcal{T}(se) < \mathcal{T}(s_*e)$ and $\mathcal{T}(se) < \mathit{pot}_{\mathcal{T}}(s)$. Let $\ell \in \mathcal{L}$ be the minimal element for which $\ell > \mathcal{T}(se)$. Hence, $\mathit{pot}_{\mathcal{T}}(s), \mathit{pot}_{\mathcal{T}}(s_*) \geq \ell$, and $\mathcal{T}(s_*e) \geq \ell$, but $\mathcal{T}(se) < \ell$. Thus, according to Def.4, $s_* \not\equiv s$ reaching a contradiction. □

3.4 The correspondence between $\equiv_f$ and a minimal FOLA

Next we prove that for every formal series f defined by a FOLA the induced equivalence relation $\equiv_f$ has a one-to-one correspondence with a minimal FOLA for f.

Utilizing the provided definitions, we can associate with a given formal series $f : \Sigma^* \to \{0, 1, \ldots, k\}$, a specific FOLA which we denote $\mathcal{A}_f$.

Definition 13 **(The FOLA $\mathcal{A}_f$).** *Let $f : \Sigma^* \to \{0, 1, \ldots, k\}$ be a formal series. Let $\mathit{reps}(f) = \{r_0, r_1, \ldots, r_n\}$. The FOLA $\mathcal{A}_f = (\Sigma, Q, q_0, \delta, \eta, F)$ is defined as follows: $Q = \mathit{reps}(f)$, $q_0 = \mathit{rep}_f(\epsilon)$, $F(r_i) = f(r_i)$, $\delta(r_i, \sigma) = \mathit{rep}_f(r_i \cdot \sigma)$ and $\eta(r_i, \sigma) = \mathit{pot}_f(r_i \cdot \sigma)$.*

We claim in Thm.17 that $\mathcal{A}_f$ recognizes the formal series f.

To prove it we associate with the formal series f a tree $\mathbb{T}_f$, whose nodes are set of words, defined as follows.

Definition 14 (The tree $\mathbb{T}_f$, the sets W_ℓ). *Let $f : \Sigma^* \to \{0, 1, \ldots, k\}$ be a formal series. Let $W_\ell = \{w \mid \textsf{pot}_f(w) \geq \ell\}$. The tree $\mathbb{T}_f$ has $k+1$ layers. The set of nodes in layer ℓ consists of the equivalence classes of $\approx_f^\ell$ intersected with W_ℓ. There is an edge from node N in layer ℓ to node N' in layer $\ell + 1$ iff $N \supseteq N'$.*

Note that $W_0 = \Sigma^*$ and $\approx_\ell^0 = \sim_f^0$ has a single equivalence class. Thus, the first layer consists of a single node (the root) which is the set Σ^*. Note also that the nodes of layer ℓ partition the set W_ℓ (i.e. their union is this set, and they are pairwise disjoint). Moreover, if two words are in the same node of layer ℓ then they are also in the same node of layer $\ell - 1$ (since $x \approx_f^\ell y$ implies $x \approx_f^{\ell-1} y$). It follows that a node in layer $\ell + 1$ is connected to a single node in layer ℓ. Thus $\mathbb{T}_f$ is indeed a tree.

Example 15. Consider the FOLA $\mathcal{D}$ depicted in Fig.7 implementing a formal series $f_\mathcal{D} : \{a, b\}^* \to \{0, 1, 2, 3, 4\}$. In Fig.7 we show the tree $\mathbb{T}_{f_\mathcal{D}}$. The first layer, layer 0, of $\mathbb{T}_{f_\mathcal{D}}$, as always consists of a single node $W_0 = \Sigma^*$. Layer 1 of $\mathbb{T}_{f_\mathcal{D}}$ also consists of a single node Σ^* since according to $f_\mathcal{D}$ the potential of all words is at least one. That is, $W_1 = \Sigma^*$. Layer 2 of $\mathbb{T}_{f_\mathcal{D}}$ consists of two nodes $W_{2a} = \{\epsilon\}$ and $W_{2b} = a\Sigma^*$. Indeed the word ϵ is differentiated from all words in $a\Sigma^*$ by $\sim_f^2$ as evident by the word b. To see why note that the potential of both ϵ and a (for instance) is $4 \geq 2$ and $f(\epsilon \cdot b) = 1 < 2$ while $f(a \cdot b) = 3 \geq 2$. Observe that $W_2 = W_{2a} \cup W_{2b} = W_1 \setminus b\Sigma^*$, since no word starting with b has a potential of 2 or more. Layer 3 of $\mathbb{T}_{f_\mathcal{D}}$ consists of four nodes $W_{3a} = \{\epsilon\}$, $W_{3b} = a(ba^*b\Sigma)^*$, $W_{3c} = a(ba^*b\Sigma)^*ba^*$ and $W_{3d} = a(ba^*b\Sigma)^*ba^*b$. Note that $W_3 = W_2 \setminus a(ba^*b\Sigma)a\Sigma^*$, since once the a transition from q_2 to q_4 is taken the potential drops to 2. Layer 4 of $\mathbb{T}_{f_\mathcal{D}}$ consists of two nodes $W_{4a} = \{\epsilon\}$, and $W_{4b} = \{a\}$, since once the b transition from q_2 to q_3 is taken the potential drops to 3. The representatives are shown below the leaves.

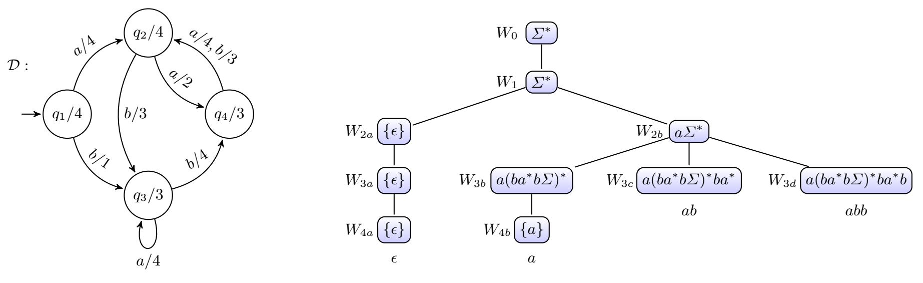

Fig. 7: A FOLA $\mathcal{D}$, and the tree $\mathbb{T}_{f_\mathcal{D}}$ induced by FOLA $\mathcal{D}$

Claim 16 connects $\equiv_f$ and the tree $\mathbb{T}_f$, and consequently the FOLA $\mathcal{A}_f$ and $\mathbb{T}_f$.

Claim 16 *Let L be a leaf in layer l of $\mathbb{T}_f$ and let u be the biggest word in L according to the $\succcurlyeq_f$ order. Then $L = \{u' \mid u' \equiv_f u\} \cap W_l$.*

Let $\mathit{leaves}(\mathbb{T}_f) = \{L_0, L_1, L_2 \ldots, L_n\}$ be the leaves of $\mathbb{T}_f$. It follows that there exists a one-to-one mapping $h : \mathit{reps}(f) \to \mathit{leaves}(\mathbb{T}_f)$ satisfying that $h(r_i) = L_i$ for $L_i = \{w \mid w \equiv_f r_i\} \cap W_{\mathit{pot}_f(r_i)}$. Since the states of $\mathcal{A}_f$ are $\mathit{reps}(f)$, this shows there is a one-to-one mapping between the states of $\mathcal{A}_f$ and the leaves of $\mathbb{T}_f$. Moreover, the representative of a word $u \in \Sigma^*$ can be found by searching for the deepest node N in the tree to which u belongs. This node is unique since nodes in the same layer are disjoint, and a node is subsumed by its parent. If N is a leaf, then the smallest word in N in the shortlex order is its representative. Otherwise let $L_1, \ldots, L_k$ be the deepest leaves in the sub-tree rooted by N. Then the smallest word in the shortlex order amongst $L_1, \ldots, L_k$ is its representative.

Thm.17 states that the desired relation between $\mathcal{A}_f$ and f holds. Its correctness follows from the stronger inductive claim, Clm.18.

Theorem 17. *The FOLA $\mathcal{A}_f$ of Def.13 correctly computes f. That is, $f(w) = \mathcal{A}_f(w)$ for every $w \in \Sigma^*$.*

Claim 18 *Let $f : \Sigma^* \to \{0, \ldots, k\}$ be a formal series, and let $\mathcal{A}_f$ be the FOLA from Def.13. Let $u \in \Sigma^*$. Then $\mathit{rep}_f(u) = r_i$ iff $\delta^*(q_0, u) = r_i$ and $\mathcal{A}_f(u) = f(u)$.*

4 The Learning Algorithm

The learning algorithm **FOL*** tries to distinguish the equivalence classes of $\equiv_f$. It does so by maintaining an observation table which keeps track of queried words. Starting with $S = \{\epsilon\} \cup \Sigma$ and $E = \{\epsilon\}$ it fills the missing entries of the table using value queries. This is done by procedure *Fill*. To extract a FOLA from a table, it is necessary to have for every distinguished equivalence class s, and any letter σ of the alphabet, a row for $s \cdot \sigma$. When this criterion holds we say that the table is *closed* as defined next.

Definition 19 (Closed Table). *An observation table $\mathcal{T} = (S, E, T)$ is termed closed if for every $s \in \mathit{reps}(\mathcal{T})$ and every $\sigma \in \Sigma$ there exists $s' \in \mathit{reps}(\mathcal{T})$ such that $s' \approx_{\mathcal{T}} s\sigma$.*

After extracting a FOLA the algorithm asks an equivalence query (EQ).[5] If the answer is "yes" the algorithm terminates. Otherwise the algorithm adds all suffixes of the counterexample w to the columns of the table and fills the table using value queries (VQ) s and repeats the process as specified in Alg.1. We show in Thm. 22 that the addition of the suffixes to the columns guarantees the learner makes progress towards identifying the correct formal series.[6]

[5] An EQ receives as an argument a FOLA $\mathcal{A}$, and checks if $[\![\mathcal{A}]\!] = f$ where f is the target formal series. If so it returns "yes", otherwise, it returns "no" with a counterexample, a word w such that $[\![\mathcal{A}]\!](w) \neq f(w)$. A value query (VQ) receives as an argument a word w and returns $f(w)$.

[6] The proof shows that Rivest and Schapire's optimization of adding just one of these suffixes [26] is possible here as well.

For each $s\sigma$ that is added to S (for $s \in S_*, \sigma \in \Sigma$), the algorithm checks (in lines 5-8) whether $s\sigma$ should be a new representative. There are three options to consider:

1. If for every $s' \in S_*$ we have $s\sigma \not\approx_{\mathcal{T}} s'$, then $s\sigma$ is indeed a new representative, and the algorithm sets $S_* \leftarrow S_* \cup \{s\sigma\}$
2. If there exists $s' \in S_*$ such that $s' \succcurlyeq_{\mathcal{T}} s\sigma$ and $s\sigma \approx_{\mathcal{T}} s'$ then no update needs to be done (and practically the algorithm defines $\mathit{rep}_{\mathcal{T}}(s\sigma) = s'$).
3. Otherwise, there exists $s' \in S_*$ such that $s\sigma \succ_{\mathcal{T}} s'$ and $s\sigma \approx_{\mathcal{T}} s'$. In this case $s\sigma$ replaces a current representative: $S_* \leftarrow (S_* \setminus \{s'\}) \cup \{s\sigma\}$. Note that there exists exactly one row s' as such in the current case, as we prove in Clm.20.

Algorithm 1 FOL*

```
1: $S := \{\epsilon\} \cup \Sigma,\ E := \{\epsilon\},\ S_* = \{\epsilon\},\ \mathcal{T} := (S, E, T),$    $\mathit{Fill}(\mathcal{T})$
2: while $\mathit{True}$ do
3:     if exists $s \in S_*$ and $\sigma \in \Sigma$ such that $s \cdot \sigma \notin S$ then
4:         $S := S \cup \{s \cdot \sigma\},$    $\mathit{Fill}(\mathcal{T})$
5:         if $s\sigma \not\approx_{\mathcal{T}} s_*$ for all $s_* \in S_*$ then
6:             $S_* \leftarrow S_* \cup \{s\sigma\}$            ▷ a new equivalence class is discovered
7:         else if $s\sigma \approx_{\mathcal{T}} s_*$ for some $s_* \in S_*$ and $s\sigma \succ_{\mathcal{T}} s_*$ then
8:             $S_* \leftarrow (S_* \setminus \{s_*\}) \cup \{s\sigma\}$     ▷ the potential of an equivalence class increased
9:     $\mathcal{A} = \mathit{ExtractAut}(S, E, T)$       ▷ the procedure ExtractAut applies Def.13 on $\equiv_{\mathcal{T}}$
10:    if EQ$(\mathcal{A})$ = ("no", $w$) then                ▷ $w$ is the counterexample
11:        $E := E \cup \mathit{Suffs}(w),$    $\mathit{Fill}(\mathcal{T})$
12:    else
13:        return $\mathcal{A}$
```

A running example is provided in App.C.

The following claim asserts that S_* never contains two representatives of the same class. Since the observation table $\mathcal{T}$ at every step of the algorithm is a subset of the Hankel Matrix $\mathcal{H}_f$ of the target series f, the size of S_* is bounded by n, the index of $\equiv_f$.

Claim 20 *In every step of the algorithm, $\forall s, s' \in S_*$ we have $s \not\approx_{\mathcal{T}} s'$.*

The following lemma asserts that if the algorithm terminates, it returns a minimal FOLA.

Lemma 21. *Let $\mathcal{T} = (S, E, T)$ be a closed observation table, and let $S_* = \mathit{reps}(\mathcal{T})$. Any FOLA consistent with $\mathcal{T}$ must have at least $|S_*|$ states.*

Termination follows from the following theorem, that guarantees that when a counterexample is received, the algorithm makes progress towards inferring the target series. It shows that either a new pair of rows is differentiated, namely a new equivalence class has been discovered, or the potential of one of the equivalence classes increases.

Theorem 22. *Let $\mathcal{T} = (S, E, T)$ be an observation table, and let $\mathcal{T}' = (S', E', T')$ be the table after processing the counterexample (i.e. after line 11). Then either $\exists s, s' \in S$*

such that $s \equiv_{\mathcal{T}} s'$ and $s \not\approx_{\mathcal{T}'} s'$ or $\equiv_{\mathcal{T}'}$ is the same as $\equiv_{\mathcal{T}}$ and $\exists s \in S$ for which $pot_{\mathcal{T}'}(rep_{\mathcal{T}'}(s)) > pot_{\mathcal{T}}(rep_{\mathcal{T}}(s))$.

Proof. Let $w = \sigma_1\sigma_2 \dots \sigma_m$ be the counterexample received for a FOLA $\mathcal{A}$ extracted from the table $\mathcal{T}$. Let $s_i = \delta(s_0, w[..i])$, that is, s_i is the state reached by the constructed FOLA $\mathcal{A}$ when reading the prefix of w of length i. Consider the following sequence (and recall that the states s_i of $\mathcal{A}$ are also strings).

$$\begin{aligned}
r_0 &= \text{VQ}(s_0 \cdot w[1..]) \\
r_1 &= pot_{\mathcal{T}}(s_0 \cdot \sigma_1) \wedge \text{VQ}(s_1 \cdot w[2..]) \\
r_2 &= pot_{\mathcal{T}}(s_0 \cdot \sigma_1) \wedge pot_{\mathcal{T}}(s_1 \cdot \sigma_1) \wedge \text{VQ}(s_2 \cdot w[3..]) \\
&\vdots \\
r_m &= pot_{\mathcal{T}}(s_0 \cdot \sigma_1) \wedge pot_{\mathcal{T}}(s_1 \cdot \sigma_2) \wedge \dots \wedge pot_{\mathcal{T}}(s_{m-1} \cdot \sigma_m) \wedge \text{VQ}(s_m \cdot \epsilon)
\end{aligned}$$

Note that r_0, the result of the first line in the sequence, is $f(w)$ since $s_0 = \epsilon$ and $w[1..] = w$, hence $r_0 = \text{VQ}(w)$. While r_m, the result of the last row, is $\mathcal{A}(w)$ because r_m corresponds exactly to the returned value of $\mathcal{A}$ on w. Since w is a counterexample $r_0 \neq r_m$. Consider the first i for which $r_i \neq r_0$. Let $r_0 = r_{i-1} = \ell$ and $r_i = \ell'$. I.e.

$$\begin{aligned}
\ell &= r_{i-1} = pot_{\mathcal{T}}(s_0 \cdot \sigma_1) \wedge \dots pot_{\mathcal{T}}(s_{i-2} \cdot \sigma_{i-1}) \wedge \text{VQ}(s_{i-1} \cdot w[i..]) \\
\ell' &= \ \ r_i \ \ = pot_{\mathcal{T}}(s_0 \cdot \sigma_1) \wedge \dots pot_{\mathcal{T}}(s_{i-2} \cdot \sigma_{i-1}) \wedge pot_{\mathcal{T}}(s_{i-1} \cdot \sigma_i) \wedge \text{VQ}(s_i \cdot w[i{+}1..])
\end{aligned}$$

There are two cases to consider.

1. Case $\ell' > \ell$:
   Since all components of the row r_{i-1} but the last one are also components of the row r_i, their value must be at least ℓ' (as otherwise the value of r_i will be less than ℓ'). It follows that the value of the last component of r_{i-1}, namely $\text{VQ}(s_{i-1} \cdot w[i..])$, is exactly ℓ (since $\ell' > \ell$, and $\text{VQ}(s_{i-1}, \sigma_i)$ is the only component in r_{i-1} that is not in r_i). While the values of $pot_{\mathcal{T}}(s_{i-1} \cdot \sigma_i)$ and $\text{VQ}(s_i \cdot w[i{+}1..])$ must be at least ℓ'. Consider the words $s = s_{i-1}\sigma_i$ and $s' = s_i$. In $\mathcal{T}$ the row s_i was the representative of $s_{i-1}\sigma$ (as per *ExtractAut*, namely Def.13), i.e., $s_{i-1}\sigma_i \equiv_{\mathcal{T}} s_i$. From $pot_{\mathcal{T}'}(s_{i-1}\sigma_i) \geq \ell'$ and $rep_{\mathcal{T}}(s_{i-1}\sigma_i) = s_i$ we get that also $pot_{\mathcal{T}'}(s_i) \geq pot_{\mathcal{T}}(s_i) \geq pot_{\mathcal{T}}(s_{i-1}\sigma_i) \geq \ell'$. Recall that we added all suffixes of w as columns in $\mathcal{T}'$. Considering the column $w[i+1..]$ we have that $T'(s_{i-1}\sigma_i, w[i+1..]) = T'(s_{i-1}, w[i..]) = \ell$ while $T'(s_i, w[i{+}1..]) \geq \ell' > \ell$. Therefore $s_{i-1}\sigma_i \not\approx^{\ell'}_{\mathcal{T}'} s_i$ proving $s_{i-1}\sigma_i \not\approx_{\mathcal{T}'} s_i$.

2. Case $\ell' < \ell$:
   Since all but the last two components of row r_i are also in row r_{i-1} their values must be at least ℓ (as otherwise the value of r_{i-1} will be less than ℓ). The value of the last two components must be at least ℓ', and at least one should be exactly ℓ'. We investigate both cases.

   (a) Case $\text{VQ}(s_i w[i+1..]) = \ell'$.
   Consider rows $s_{i-1}\sigma_i$ and s_i. From $\text{VQ}(s_{i-1}w[i..]) \geq \ell$ we get that $T(s_{i-1}\sigma_i, w[i+1..]) \geq \ell > \ell'$ while $T(s_i, w[i+1..]) = \ell'$. Since s_i is the representative of $s_{i-1}\sigma_i$ in $\mathcal{T}$, we know from Claim 12 that for all columns $e \in E$ (before adding the suffixes of the counterexample) we have $T(s_i, e) \geq T(s_{i-1}\sigma_i, e)$.

(i) If for one of the columns the relation is strict, namely $T(s_i, e) > T(s_{i-1}\sigma_i, e)$ then since in column $w[i+1..]$ we have the opposite relation by Claim 5 $s_i \not\approx_{\mathcal{T}'} s_{i-1}\sigma_i$ so the claims hold since we separated states.

(ii) Otherwise if the relation is $=$ in all columns $e \in E$ then $\mathit{pot}_{\mathcal{T}}(s_i) = \mathit{pot}_{\mathcal{T}}(s_{i-1}\sigma_i)$.
   - If $\mathit{pot}_{\mathcal{T}}(s_{i-1}\sigma_i) = \mathit{pot}_{\mathcal{T}}(\mathit{rep}_{\mathcal{T}}(s_{i-1}\sigma_i)) < \ell$ then the potential increased since now $\mathit{pot}_{\mathcal{T}'}(\mathit{rep}_{\mathcal{T}'}(s_{i-1}\sigma_i)) \geq \ell$.
   - Otherwise $\mathit{pot}_{\mathcal{T}}(s_{i-1}\sigma_i) \geq \ell$. Since $\mathit{pot}_{\mathcal{T}}(s_i) \geq \ell$ we get that $s_i \not\sim^{\ell}_{\mathcal{T}'} s_{i-1}\sigma_i$ (as evident by column $w[i+1..]$).

(b) Case $\text{VQ}(s_i w[i+1..]) > \ell'$ and $\mathit{pot}_{\mathcal{T}}(s_{i-1}\sigma_i) = \ell'$.
Since s_1 is the representative of $s_0\sigma_1$ we get that $T(s_1, w[2..]) \geq \ell$. This in turn implies from the same reasoning that $T(s_1\sigma_2, w[3..]) \geq \ell$ and $T(s_2, w[3..]) \geq \ell$. If we keep going on this way we get that $T(s_i\sigma_{i-1}, w[i+1..]) \geq \ell$. The potential of $s_{i-1}\sigma_i$ in $\mathcal{T}$ is $\ell' < \ell$. If the potential of its representative s_i was also ℓ' then the potential of this equivalence class in $\mathcal{T}'$ increased since it is now at least $\ell > \ell'$. If the potential of s_i is more than ℓ' then $s_i \not\sim^{\ell'+1} s_{i-1}\sigma_i$ since the potential of both is at least $\ell'+1$ and in column $w[i+1..]$ only one of them is less than $\ell'+1$. □

Corollary 23. *Let $\mathbb{FOLA}$ be the class of languages represented by FOLAs. The algorithm* ***FOL***** *terminates and correctly learns any target language $L \in \mathbb{FOLA}$.*

Following Thm.22 we can bound the number of equivalence queries, call it m_{EQ} by $n|\mathcal{L}|$, since every counterexample either reveals a new equivalence class, or provides evidence that the potential of a class is higher. The number of VQs is bounded by the size of the obtained table. The table has at most $n|\mathcal{L}| + n|\mathcal{L}||\Sigma|$ rows since a new row is added to S_* only if it revealed a new equivalence class or it increased the potential of a known class, and when a row is added to S_* all its one letter extensions are added to S. The number of columns is bounded by c times m_{EQ} where c is the size of the longest counterexample.[7] While these theoretical bounds are the same as $\mathbf{L}^*$ for SFOLA, as discussed in page.3 they cannot be better, and as we show in Sec.5, in practice the number of EQ and VQ issued by our algorithm is significantly smaller than that by $\mathbf{L}^*$.

5 Empirical Results

We implemented the algorithm and compared its performance on randomly generated FOLAs against the straightforward extension of $\mathbf{L}^*$ to learn SFOLAs.[8] We compared them in terms of (a) the number of states obtained (b) the number of issued value queries and (c) the number of issued equivalence queries. We used a binary alphabet $\Sigma = \{a, b\}$, the number of states N was chosen uniformly at random amongst the values $\{1, ..., 70\}$ and the size of the lattice K was chosen uniformly at random amongst $\{2, ..., 70\}$ (i.e. $\mathcal{L} = \{0, ..., K\}$). For each state q and letter σ, the state to transit to was

[7] This can be strengthened to $\log(c)$ times m_{EQ} using the optimization that finds one suffix to add to the columns, as described in the proof of Thm.22.

[8] In this extension the observation table has answers to value queries (as in **FOL***) but two rows are determined equivalent iff they are exactly the same. All transitions values are set to $\top$, and the state values are determined by the value of the respective row in the column ϵ.

chosen uniformly at random amongst $\{1, ..., N\}$ and the transition value was chosen uniformly at random amongst $\{1, ..., K\}$. The initial state was fixed to be 1. Finally, for each state the state-value was chosen uniformly at random amongst $\{1, ..., K\}$.

Note that the generated automata may not necessarily be minimal in terms of the number of states, and may not utilize all the available $K+1$ lattice values. We thus define n to be the number of states in the minimal FOLA for the formal series $f : \Sigma^* \to \{0, ..., K\}$ computed by the generated automaton, and k as the number of values that are possible outputs of this automaton, meaning $k = |\mathit{Image}(f)|$.[9] In addition, we define n_s to be the number of states in the minimal SFOLA for that language. The implementation of the algorithm and the tests are available in https://github.com/sagisaa/Learning_FOLA.

We generated 10334 automata as specified, and ran both algorithms $\mathbf{L}^*$ and $\mathbf{FOL}^*$ on the languages induced by these automata. A VQ for a word w was answered by running the word on the generated automata, and the EQs were answered using a complete equivalence check, as specified in [14]. The gray bars on the graphs show the number of samples for a certain x, (denoted 'Count') and their scale is placed on the right y-axis. Each point on the graphs indicates the average result of the samples with the same x.

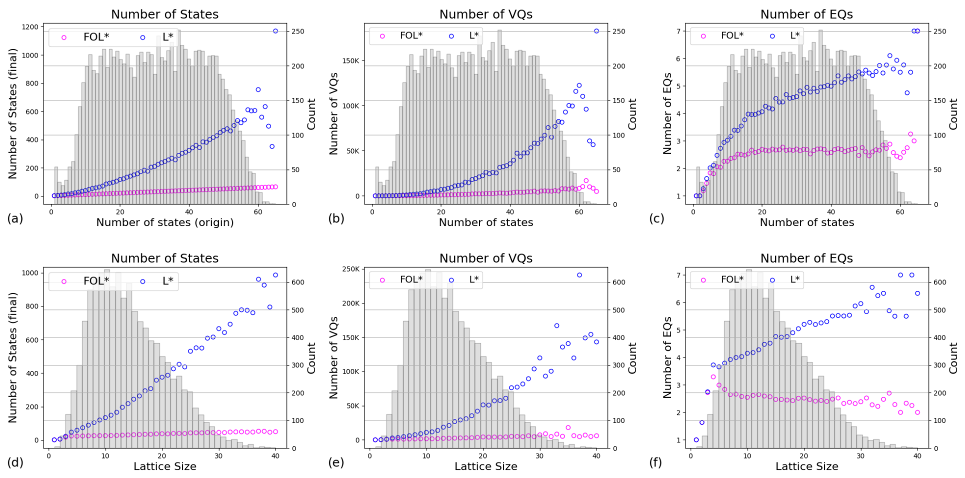

Fig. 8: $\mathbf{L}^*$and $\mathbf{FOL}^*$comparison: Number of states, VQs, EQs

The graphs are organized as three pairs, measuring number of states of the resulting automaton, number of issued VQ, and number of issued EQ. The upper row measures these with respect to the actual number of states (n), and the lower row with respect to the actual lattice size (k).

The first pair of graphs (a) and (d) provide the number of states of the resulting automaton in $\mathbf{L}^*$ vs $\mathbf{FOL}^*$ measured with respect to n and k, resp. Recall that the output of $\mathbf{L}^*$ is an SFOLA and the output of $\mathbf{FOL}^*$ is a FOLA, and both algorithms return the

[9] Note that k, the number of lattice values occurring in transitions or state-values, is bounded by $n + n|\Sigma|$ where n is the number of states. Thus, for a constant-size alphabet it is $O(n)$.

minimal one. These graphs show that the number of states in SFOLA is about k times bigger than the minimal FOLA. This conclusion is supported with regression testing on the relation between k and the number of states in each type given in App. D.

The second pair of graphs (b) and (e) provide the number of VQs issued by $\mathbf{L}^*$ vs $\mathbf{FOL}^*$ measured with respect to to n and k, resp. These graphs show that the relation between the number of states in the minimal matching representation and the number of VQs is roughly quadratic. This result is compatible with the structure of the algorithm, since the data of the VQs is organized in a table in which the number of rows and the number of columns are $O(n)$ each.[10]

Last important factor we looked at is the number of EQs required in order for the algorithm to converge. The third pair of graphs (c) and (f) provide the number of EQs issued by $\mathbf{L}^*$ vs $\mathbf{FOL}^*$ measured with respect to n and k, resp. These graphs show that the number of EQs required by the $\mathbf{FOL}^*$algorithm *decreases* when the lattice size k increases. This can be explained by the fact that the higher the lattice size is, the easier it is to distinguish between rows. With that said, less EQs are needed since states are discovered sooner when closing the table. To make sure of that result, we use confidence interval method (CI) of 99% to distinguish between the 2 methods, see Fig.9.

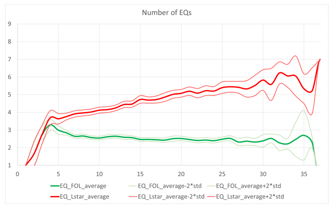

Fig. 9: EQs in relation to k

6 Conclusions

We provided a definition of equivalence classes for a formal series recognizable by a FOLA, which yields a canonical minimal FOLA and a Myhill-Nerode theorem, namely a one-to-one relation between the equivalence classes and the canonical FOLA. Based on it we designed a specialized learning algorithm that outputs the canonical FOLA and compared it against $\mathbf{L}^*$ on synthetically generated FOLAs. Our experiments show a clear advantage to using $\mathbf{FOL}^*$ as it outperforms $\mathbf{L}^*$ in the number of states of the obtained FOLA, the number of issued VQs, and the number of issued EQs.

References

1. F. Aarts and F. Vaandrager. Learning I/O automata. In *CONCUR*, 2010.
2. D. Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75(2), 1987.
3. D. Angluin, T. Antonopoulos, and D. Fisman. Strongly unambiguous Büchi automata are polynomially predictable with membership queries. In *28th Computer Science Logic CSL*, 2020.
4. D. Angluin, S. Eisenstat, and D. Fisman. Learning regular languages via alternating automata. In *Proc. of the 24th Intr. Joint Conf. on Artificial Intelligence, IJCAI*, 2015.
5. D. Angluin and D. Fisman. Learning regular omega languages. In *Algorithmic Learning Theory - 25th Intr. Conf., ALT, Proc.*, 2014.
6. A. Bakhirkin, T. Ferrère, and O. Maler. Efficient parametric identification for stl. In *Proc. of the 21st Intr. Conf. on Hybrid Systems: Computation and Control*, HSCC '18, 2018.

[10] The number of columns is bounded by nc where c is the size of the longest counterexample.

7. B. Balle and M.r Mohri. Learning weighted automata. In *Algebraic Informatics - 6th Intr. Conf., CAI. Proc.*, 2015.
8. A. Beimel, F. Bergadano, N. H. Bshouty, E. Kushilevitz, and S. Varricchio. Learning functions represented as multiplicity automata. *J. ACM*, 47(3), 2000.
9. B. Bollig, P. Habermehl, C. Kern, and M. Leucker. Angluin-style learning of NFA. In *IJCAI Proc. of the 21st Intr. Joint Conf. on Artificial Intelligence*, 2009.
10. W. Chan. Temporal-locig queries. In *Computer Aided Verification, 12th Intr. Conf., CAV, Proc.*, 2000.
11. D. Drews and L. D'Antoni. Learning symbolic automata. In *Tools and Algorithms for the Construction and Analysis of Systems - 23rd Intr. Conf., TACAS*, 2017.
12. S. M. Easterbrook, M. Chechik, B. Devereux, A. Gurfinkel, A. Y. C. Lai, V. Petrovykh, A. Tafliovich, and C. D. Thompson-Walsh. A model checker for multi-valued reasoning. In *Proc. of the 25th Intr. Conf. on Software Engineering*, 2003.
13. S. Graf and H. Saïdi. Construction of abstract state graphs with PVS. In *Computer Aided Verification, 9th Intr. Conf., CAV, Proc.*, 1997.
14. S. Halamish and O. Kupferman. Minimizing deterministic lattice automata. *ACM Trans. Comput. Log.*, 16(1), 2015.
15. T. A. Henzinger. From boolean to quantitative notions of correctness. In *The 37th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages, POPL*, 2010.
16. S. Huang and R. Cleaveland. Temporal-logic query checking over finite data streams. In *Formal Methods for Industrial Critical Systems - 25th Intr. Conf., FMICS*, 2020.
17. A. Hussain and M. Huth. On model checking multiple hybrid views. *Theor. Comput. Sci.*, 404(3), 2008.
18. S. Jakšić, E. Bartocci, R. Grosu, and D. Ničković. An algebraic framework for runtime verification. *IEEE Trans. on Computer-Aided Design of Integ. Circuits and Systems*, 2018.
19. V. Kumar, P. Madhusudan, and M. Viswanathan. Minimization, learning, and conformance testing of boolean programs. In *CONCUR*, 2006.
20. O. Kupferman and Y. Lustig. Lattice automata. In *Verification, Model Checking, and Abstract Interpretation*, 2007.
21. K. Mamouras, A. Chattopadhyay, and Z. Wang. A compositional framework for quantitative online monitoring over continuous-time signals. In *Runtime Verification*, 2021.
22. M. Mohri. Finite-state transducers in language and speech processing. *Comput. Linguistics*, 23(2), 1997.
23. J. Myhill. Finite automata and the representation of events. Technical report, Wright Patterson AFB, Ohio, 1957.
24. A. Nerode. Linear automaton transformations. In *Proc. of the American Mathematical Society, 9(4)*, 1958.
25. D. Nitay, D. Fisman, and M. Ziv-Ukelson. Learning of structurally unambiguous probabilistic grammars. In *35th AAAI Conf. on Artificial Intelligence, AAAI*.
26. R. L. Rivest and R. E. Schapire. Inference of finite automata using homing sequences. In *Machine Learning: From Theory to Applications*, volume 661 of *LNCS*, 1993.
27. V. Sakakibara. Learning context-free grammars from structural data in polynomial time. In *Proc. of the First Ann. Workshop on Computational Learning Theory, COLT*, 1988.
28. M. Shahbaz and R. Groz. Inferring Mealy machines. In *FM Formal Methods, Second World Congress. Proc.*, 2009.
29. J. Streb and P. Alexander. Using a lattice of coalgebras for heterogeneous model composition. In *MoDELS'06*, 2006.
30. F. W. Vaandrager. Model learning. *Commun. ACM*, 60(2), 2017.
31. G. van Heerdt, M. Sammartino, and A. Silva. Learning automata with side-effects. In *Coalgebraic Methods in Computer Science - 15th IFIP WG 1.3 Intr. Workshop, CMCS*, 2020.

Appendix

A a smallest SFOLA equivalent to the FOLA of Fig.2

Recall the FOLA provided in Fig.2, page.2.

An equivalent SFOLA, is shown in Fig.10. As explained in the introduction (on page 2), it require $|\mathcal{L}|$ times more states.

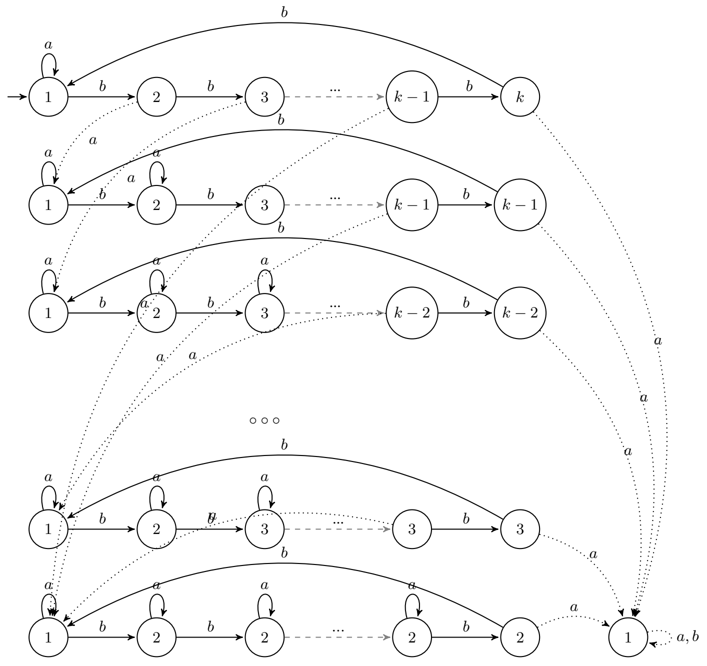

Fig. 10: An equivalent SFOLA for the FOLA in Fig 2

B Omitted Proofs

Lemma 5 (restated) *Let $\mathcal{T} = (S, E, T)$. Let $s_1, s_2 \in S$ and $e_1, e_2 \in E$. If $T(s_1, e_1) < T(s_2, e_1)$ while $T(s_1, e_2) > T(s_2, e_2)$ then $s_1 \not\approx_{\mathcal{T}} s_2$.*

Proof. Let $T(s_i, e_j) = \ell_{ij}$. Then $\ell_{11} < \ell_{21}$ and $\ell_{12} > \ell_{22}$. Assume w.l.o.g. that $\ell_{12} \leq \ell_{21}$. Clearly, $pot_{\mathcal{T}}(s_1) \geq \ell_{12}$. Since $\ell_{21} \geq \ell_{12}$ we get that $pot_{\mathcal{T}}(s_2) \geq \ell_{12}$. Looking at column e_2 we have that $T(s_1, e_2) \geq \ell_{12}$ while $T(s_2, e_2) < \ell_{12}$. Thus $s_1 \not\sim^{\ell_{12}} s_2$ implying $s_1 \not\approx_{\mathcal{T}} s_2$. □

Lemma 7 (restated) *Let $s_1, s_2, s_3 \in S$. If $s_1 \approx_{\mathcal{T}} s_2, s_2 \approx_{\mathcal{T}} s_3$, $s_2 \not\succ_{\mathcal{T}} s_1$ and $s_2 \not\succ_{\mathcal{T}} s_3$ then $s_1 \approx_{\mathcal{T}} s_3$.*

Proof. Let $s_1, s_2, s_3 \in S$ and suppose that $s_1 \approx_{\mathcal{T}} s_2, s_2 \approx_{\mathcal{T}} s_3$ and $s_2 \not\succ_{\mathcal{T}} s_1, s_3$. Assume toward contradiction that $s_1 \not\approx_{\mathcal{T}} s_3$, therefore there exists $l \in \mathcal{L}$ such that $pot_{\mathcal{T}}(s_1), pot_{\mathcal{T}}(s_3) \geq l$ and there exists $e \in E$ for which w.l.o.g. $T(s_1, e) \geq l$ and $T(s_3, e) < l$. Consider $T(s_2, e)$: If $T(s_2, e) \geq l$, then $s_2 \not\sim^{l}_{\mathcal{T}} s_3$, and therefore $s_2 \not\approx_{\mathcal{T}} s_3$. Otherwise, $T(s_2, e) < l$, and since $pot_{\mathcal{T}}(s_2) \geq l$, we get that $s_2 \not\approx_{\mathcal{T}} s_1$. A contradiction is achieved in both cases. □

Lemma 10 (restated) *The relation $\equiv_f$ is an equivalence relation.*

Proof. Trivially, $x \equiv x$ for any x, thus $\equiv_f$ is reflexive. Clearly, $x \equiv_f y$ iff $y \equiv_f x$ thus $\equiv_f$ is symmetric. It remains to show that $\equiv_f$ is transitive. Assume $x \equiv_f y$ and $y \equiv_f z$. Let $r_x = rep_f(x)$, $r_y = rep_f(y)$ and $r_z = rep_f(z)$. It follows from Def.9 that $r_x = r_y = r_z$ implying $x \equiv_f z$. □

Lemma 11 (restated) *The relation $\equiv_f$ is a right congruence relation. That is, $x \equiv_f y$ implies $xz \equiv_f yz$ for all $z \in \Sigma^*$.*

Proof. Assume towards contradiction that $\exists x, y, z \in \Sigma^*$ s.t. $x \equiv_f y$ yet $xz \not\equiv_f yz$. Then $\exists l \in L$, s.t. $pot_f(xz) \geq l$, $pot_f(yz) \geq l$ and $\exists w \in \Sigma^*$ s.t. $f(xz, w) \geq l$ and $f(yz, w) < l$ or vice versa. This implies that $pot_f(x) \geq l$, $pot_f(y) \geq l$ and $\exists w \in \Sigma^*$ s.t. $f(x, zw) \geq l$ and $f(y, zw) < l$ or vice versa, which contradicts that $x \equiv_f y$. □

Lemma 16 (restated) *Let L be a leaf in layer l of $\mathbb{T}_f$ and let u be the biggest word in L according to the $\not\succ_f$ order. Then $L = \{u' \mid u' \equiv_f u\} \cap W_l$.*

Proof. Let $u' \in L$. By definition of $\mathbb{T}_f$ since L is in layer l it must be that $u' \in W_l$. We show that $u' \equiv_f u$. If $u' \not\equiv_f u$ then by Def.9 they have a different representative, namely $u' \not\approx_f u$. From Def.4 we get that $u \not\approx^{l'}_f u'$ for some $l' \in \mathcal{L}$. Thus u and u' will be separated in layer l' of the tree contradicting that both reside in leaf L. Thus $u' \equiv_f u$.

Let $u' \in W_l$ s.t. $u' \equiv_f u$. We show that $u' \in L$. Since $u' \in W_l$ then $u' \in W_{l'}$ for every $l' < l$. Since $u' \equiv_f u$ we have that $u' \approx^{l}_f u$ for every $l \in \mathcal{L}$. Thus u' and u will not be separated. □

Lemma 18 (restated) *Let $f : \Sigma^* \to \{0, \ldots, k\}$ be a formal series, and let $\mathcal{A}_f$ be the FOLA from Def.13. Let $u \in \Sigma^*$. Then $rep_f(u) = r_i$ iff $\delta^*(q_0, u) = r_i$ and $\mathcal{A}_f(u) = f(u)$.*

Proof. The proof is by induction on the length of the word. For $w = \epsilon$ we have that

1. $rep_f(\epsilon) = q_0$ and $\delta^*(q_0, \epsilon) = q_0$.
2. $\mathcal{A}_f(\epsilon) = trvl(\epsilon) \wedge F(\epsilon) = \top \wedge f(\epsilon) = f(\epsilon)$.

Let $w = u\sigma$ for $u \in \Sigma^*$ and $\sigma \in \Sigma$.

1. $\delta^*(q_0, u\sigma) = \delta(\delta^*(q_0, u), \sigma)$ by def of δ^*. By induction hypothesis $\delta^*(q_0, u) = rep_f(u)$, thus $\delta(\delta^*(q_0, u), \sigma) = \delta(rep_f(u), \sigma)$. By def of $\mathcal{A}_f$, $\delta(rep_f(u), \sigma) = rep_f(u\sigma)$.

2. Assume $f(w) = l$ and $w = \sigma_1\sigma_2\ldots\sigma_m$. Let $r_i = \mathit{rep}_f(w[..i])$ for $0 \leq i \leq m$. Now $\mathcal{A}_f(w) = \delta(q_0,\sigma_1) \wedge \delta(r_1,\sigma_2) \wedge \ldots \wedge \delta(r_{m-1},\sigma_m) \wedge F(r_m)$.Thus $\mathcal{A}_f(w) = \mathit{pot}_f(r_0\sigma_1) \wedge \mathit{pot}_f(r_1\sigma_2) \wedge \ldots \wedge \mathit{pot}_f(r_{m-1}\sigma_m) \wedge f(r_m)$. For every $0 \leq i < m$ we have $\mathit{pot}_f(r_i\sigma_{i+1}) \geq \mathit{pot}_f(w[..i]\sigma_{i+1}) \geq \mathit{pot}_f(w)$ (where first inequality holds since r_i is the representative of $w[..i]$ and the second since $w[..i]\sigma_{i+1}$ is a prefix of w). In addition $f(r_m) \geq f(w)$, again since r_m is the representative of w (Clm.12). Thus $\mathcal{A}_f(w) \geq \mathit{pot}_f(w) \wedge f(w) \geq f(w)$.
   If $f(r_m) = f(w)$ then the value of $\mathcal{A}_f(w)$ is correct. Otherwise, we show that $\mathit{pot}_f(r_i\sigma_{i+1}) = l$ for some $0 \leq i < m$, which proves the claim. Assume not, then $\mathit{pot}_f(r_i\sigma_{i+1}) > l$ for all $0 \leq i < m$. This implies that $\mathit{pot}_f(\mathit{rep}_f(r_i\sigma_{i+1})) > l$ for all $0 \leq i < m$. Since $\mathit{rep}_f(r_i\sigma_{i+1}) = r_{i+1}$ we get $\mathit{pot}_f(\mathit{rep}_f(r_i)) > l$ for all $1 \leq i \leq m$. In addition, $\mathit{pot}_f(q_0) = \top > l$. Thus r_i are in layer $l' > l$ of $\mathbb{T}_f$ for all $0 \leq i \leq m$. It follows from Def.14 and Claim 16, that r_i is in $W_{l'}$ for some $l' > l$ for all $0 \leq i \leq m$. In particular, since $w \equiv_f r_m$ (by Claim 16) we get that $w \in W_{l'}$ for some $l' > 1$ contradicting that $f(w) = l$. □

Lemma 20 (restated) *In every step of the algorithm, $\forall s, s' \in S_*$ we have $s \not\approx_{\mathcal{T}} s'$.*

Proof. Let S_i denote the set S_* in the i-the iteration. When the algorithm starts we have $S_0 = \{\epsilon\}$ for which the claim holds. It is enough to show that every update of the representatives set does not change the status of this claim. It is clear why the first two cases in the update do not affect it. As for the third case, assume toward contradiction that at some point $\exists s, s' \in S_*$ such that $s \approx_{\mathcal{T}} s'$ and let S_k be the first in which this happened. That is, this occurred after applying $S_k \leftarrow (S_{k-1} \setminus \{s'\}) \cup \{s\sigma\}$. Since the claim holds for S_{k-1}, the change is due to the addition of $\{s\sigma\}$, so we have $s'' \in S_k$ such that $s\sigma \approx_{\mathcal{T}} s''$ and $s\sigma \succ_{\mathcal{T}} s''$ (or else $s\sigma$ would not have been added). From case 3 we have the same for s': $s\sigma \approx_{\mathcal{T}} s'$ and $s\sigma \succ_{\mathcal{T}} s'$. Since $s', s'' \in S_{k-1}$, it holds that $s' \not\approx_{\mathcal{T}} s''$. Assume w.l.o.g that $s' \not\succ_{\mathcal{T}} s''$, so $\exists l \in \mathcal{L}, e \in E$ such that $\mathit{pot}_{\mathcal{T}}(s') \geq l$, $\mathit{pot}_{\mathcal{T}}(s'') \geq l$ and $\exists e \in E$ s.t. $T(s', e) \geq l$ and $T(s'', e) < l$. Since $s\sigma \approx_{\mathcal{T}} s'$ and $s\sigma \succ_{\mathcal{T}} s'$ we get from Clm.12 that $\mathit{pot}_{\mathcal{T}}(s\sigma) \geq l$ and $T(s\sigma, e) \geq l$, and therefore $s\sigma \not\approx_{\mathcal{T}} s''$, contradiction.

Lemma 21 (restated) *Let $\mathcal{T} = (S, E, T)$ be a closed observation table, and let $S_* = \mathit{reps}(\mathcal{T})$. Any FOLA consistent with $\mathcal{T}$ must have at least $|S_*|$ states.*

Proof. Assume toward contradiction that there exists a FOLA with less than $|S_*|$ states that is consistent with the table. Then by Clm.20 there must exist $s, s' \in S_*$ such that $s \not\equiv_{\mathcal{T}} s'$ but $\mathcal{A}$ on reading s and s' reaches the same state. From the definition of $\equiv_{\mathcal{T}}$ and *reps* this implies $s \not\approx_{\mathcal{T}} s'$. But from Lemma.6 this contradicts that s, s' reach the same state. □

C Running Example

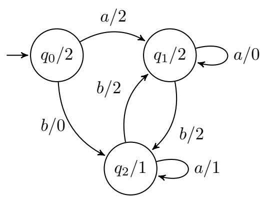

Fig. 11: A FOLA $\mathcal{M}$

Consider the FOLA $\mathcal{M}$ given in Fig.11 where $\Sigma = \{a, b\}$ and $\mathcal{L} = \{0, 1, 2\}$ and let $f = [\![\mathcal{M}]\!]$ be the formal series induced by $\mathcal{M}$. The algorithm maintains the observation table, starting with the rows $\{\epsilon, a, b\}$ and column $\{\epsilon\}$, and fills the entries using VQ, resulting in the table marked in the inner frame (in blue) of Fig.12.

Looking at ϵ and a, we can see that $a \approx_{\mathcal{T}} \epsilon$ since for every $\ell \in \mathcal{L}$ we have $a \sim^{\ell}_{\mathcal{T}} \epsilon$. We make the same conclusion for b. As $\epsilon \succcurlyeq_{\mathcal{T}} a, b$, we define $\mathit{rep}_{\mathcal{T}}(a) = \mathit{rep}_{\mathcal{T}}(b) = \epsilon$, which means that $\mathit{reps}(\mathcal{T}) = \{\epsilon\}$. The table is closed since $\forall s_* \in \mathit{reps}(\mathcal{T})$ and $\sigma \in \Sigma$ we have $s_* \cdot \sigma \in S$. An automaton can thus be extracted. The extracted automaton is $\mathcal{A}_0$ depicted in Fig.12.

Since $\mathcal{A}_0$ does not recognize the target formal series, we get a counterexample. Suppose the counterexample is $w = aa$ (for which $\mathcal{A}_0$ gives the value of 2, but the correct value is 0). We add the suffixes of w, i.e. a and aa, to the columns E, and use VQs to fill the missing entries.

We now realize that $a \not\approx_{\mathcal{T}} \epsilon$ since, considering $\ell = 2$, we have that $\mathit{pot}_{\mathcal{T}}(\epsilon) \geq 2$ and $\mathit{pot}_{\mathcal{T}}(a) \geq 2$ but the column a distinguishes them since $T(\epsilon, a) \geq 2$ while $T(a, a) < 2$, thus $a \not\sim^{2}_{\mathcal{T}} \epsilon$. However, we still have $b \approx_{\mathcal{T}} \epsilon$ and $a \approx_{\mathcal{T}} b$. Since $\epsilon \not\succcurlyeq_{\mathcal{T}} a$, we have $\mathit{rep}_{\mathcal{T}}(b) = \epsilon$ and $\mathit{rep}_{\mathcal{T}}(a) = a$. Thus, the representatives are $\mathit{reps}(\mathcal{T}) = \{\epsilon, a\}$. For the purpose of closing the table, we add the rows $a \cdot \Sigma = \{aa, ab\}$. Considering ab, we get that $ab \not\approx_{\mathcal{T}} \epsilon$ since $ab \not\sim^{1}_{\mathcal{T}} \epsilon$ because aa is a distinguishing column (indeed $\mathit{pot}_{\mathcal{T}}(\epsilon) \geq 1$, $\mathit{pot}_{\mathcal{T}}(ab) \geq 1$, $T(\epsilon, aa) < 1$ and $T(ab, aa) \geq 1$). Similarly, we get that $ab \not\approx_{\mathcal{T}} a$. Thus, the representatives are now $\mathit{reps}(\mathcal{T}) = \{\epsilon, a, ab\}$. Once again we close the table, with the rows $ab \cdot \Sigma$. Since the table is closed and no new representative was found, we construct the automaton $\mathcal{A}_1$ depicted in Fig.12.

While the automaton $\mathcal{A}_1$ has the correct number of states it is still incorrect. Suppose the returned counterexample is now abb (on which $\mathcal{A}_1$ returns 1 but the correct answer

	ϵ	aa	a	abb	bb	b
● ϵ	2	0	2	2	0	0
● a	2	0	0	0	2	1
b	0	0	0	0	0	0
aa	0	0	0	0	0	0
● ab	1	1	1	1	1	2
aba	1	1	1	1	1	2
abb	2	0	0	0	2	1

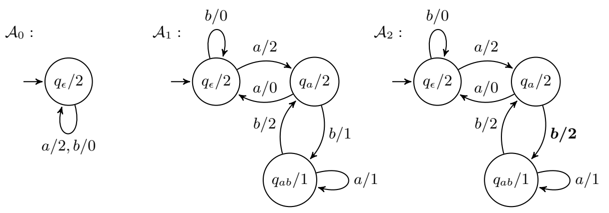

Fig. 12: The observation table maintained by **FOL*** in learning $f = [\![\mathcal{M}]\!]$ from Fig.1, the first, second and third conjectures, $\mathcal{A}_0$, $\mathcal{A}_1$ and $\mathcal{A}_2$ respectively.

is 2). The algorithm adds all suffixes of abb namely $\{abb, abb, b\}$ to the table, and fills in the missing entries. The new entry $T(ab, b)$ has value 2 increasing the potential of ab from 1 to 2. The new extracted automaton $\mathcal{A}_2$ depicted in Fig.12 (which differs from $\mathcal{A}_1$ in the transition value of b from q_a) correctly recognizes the target language. This conjecture is indeed correct, and so the algorithm terminates. Note that the resulting FOLA is different from the target FOLA, but it is a minimal FOLA accepting the same formal series, and is the canonical one.

D Regressions

Below we provide the results using regression analysis.

Regression for n_s/n and k We use the method of least squares to check the dependency of the number of states n in the minimal FOLA and the number of states n_s in the minimal SFOLA, with respect to k, the actual size of the lattice in the smallest FOLA.

Dependent Variable: n_s/n
Method: Least Squares
Sample (adjusted): 1 10334
Included observations: 10334 after adjustments

Variable	Coefficient	Std. Error	t-Statistic	Prob.
k	0.484337	0.000987	490.7155	0.0000

R-squared	0.804151	Mean dependent var	6.876312
Adjusted R-squared	0.804151	S.D. dependent var	3.546382
S.E. of regression	1.569447	Akaike info criterion	3.739420
Sum squared resid	25451.86	Schwarz criterion	3.740121
Log likelihood	-19320.58	Hannan-Quinn criter.	3.739657

Table 1: n_s/n and k

The results are shown in Table.1. We can see that $Prob.$ is 0 which supports that k, the lattice size explains the difference.

Regression for $\text{VQ}_{\mathbf{FOL}^*}/\text{VQ}_{\mathbf{L}^*}$ and n/n_s (Table 2) We use the method of least squares to check the dependency of the number of VQ issued by $\mathbf{FOL}^*$ which we denote $\text{VQ}_{\mathbf{FOL}^*}$ and the the number of VQ issued by $\mathbf{L}^*$ which we denote $\text{VQ}_{\mathbf{L}^*}$, with respect to $\frac{n}{n_s}$ the ratio between the number of states in the smallest FOLA and smallest SFOLA. The results are shown in Table.2. We can see that the coefficient of the dependency of $\text{VQ}_{\mathbf{FOL}^*}/\text{VQ}_{\mathbf{L}^*}$ in $\frac{n}{n_s}$ is 1 (approximately) indicating a tightest correlation.

Dependent Variable: $\mathrm{VQ}_{\mathbf{FOL}^*}/\mathrm{VQ}_{\mathbf{L}^*}$
Method: Least Squares
Sample: 1 10335
Included observations: 10335

Variable	Coefficient	Std. Error	t-Statistic	Prob.
n/n_s	1.002312	0.005571	179.9041	0.0000

R-squared	0.541265	Mean dependent var	0.194035
Adjusted R-squared	0.541265	S.D. dependent var	0.205058
S.E. of regression	0.138885	Akaike info criterion	-1.110237
Sum squared resid	199.3344	Schwarz criterion	-1.109536
Log likelihood	5738.150	Hannan-Quinn criter.	-1.110000

Table 2: $\mathrm{VQ}_{\mathbf{FOL}^*}/\mathrm{VQ}_{\mathbf{L}^*}$ and n/n_s