Query Learning of Residual Symbolic Automata

Kaizaburo Chubachi
Graduate School of Information Sciences
Tohoku University
kaizaburo_chubachi@shino.ecei.tohoku.ac.jp

Ryo Yoshinaka
Tohoku University
ryoshinaka@tohoku.ac.jp

Diptarama Hendrian
Tohoku University
diptarama@tohoku.ac.jp

Ayumi Shinohara
Tohoku University
ayumisg@tohoku.ac.jp

Extended abstract of the 14th International Conference on Grammatical Inference *ICGI'18*, September 5–7, 2018, Wrocław, Poland.

Abstract

We propose a learning algorithm for *residual symbolic finite-state automata* (RSFA) under the minimally adequate teacher (MAT) model [1]. MAT learning algorithms for residual finite-state automata (RFA) [4] and deterministic symbolic finite state automata (SFA) [6] have been proposed in [3] and [2], respectively. This work extends these works.

Author Keywords

MAT learning algorithms; residual symbolic finite-state automata.

In an SFA, transitions carry predicates over a decidable Boolean algebra $\mathcal{A}$ on a (typically huge or infinite) alphabet $\mathfrak{D}$. Following the definition in [2], we will use the notation $\mathcal{A} = (\mathfrak{D}, \Psi, [\![\_]\!], \bot, \top, \vee, \wedge, \neg)$ for an effective Boolean algebra, where Ψ is a set of predicates closed under the Boolean connectives and $[\![\_]\!]\colon \Psi \to 2^{\mathfrak{D}}$ is a denotation function. We assume it is decidable whether $[\![\varphi]\!] = \emptyset$ for any $\varphi \in \Psi$ and moreover there is an effective procedure to find an element of $[\![\varphi]\!]$ unless $[\![\varphi]\!] = \emptyset$. An SFA is $M = (\mathcal{A}, Q, Q_0, F, \Delta)$ where Q is the state set, $Q_0 \subseteq Q$ and $F \subseteq Q$ are the initial and final state sets, respectively, and $\Delta \subseteq Q \times \Psi \times Q$ is the transition relation. For Δ, we use δ to denote the transition function $\delta\colon Q \times \mathfrak{D} \to 2^Q$ such that $\delta(q, a) = \{q' \mid (q, \varphi, q') \in \Delta,\ a \in [\![\varphi]\!]\}$ for $q \in Q$ and $a \in \mathfrak{D}$. As usual, δ is extended to $\delta\colon Q \times \mathfrak{D}^* \to 2^Q$. Let

$L_q = \{w \in \mathfrak{D}^* \mid \delta(q, w) \cap F \neq \emptyset\}$ for $q \in Q$. The language $L(M)$ accepted by M is $\bigcup_{q \in Q_0} L_q$.

A language L' is a *residual language* of L if there is $u \in \mathfrak{D}^*$ with $L' = \{v \in \mathfrak{D}^* \mid uv \in L\}$. The set of residual languages of L is denoted by $Res(L)$. A language L is called *prime* in a class $\mathbb{L}$ of languages if $L \neq \bigcup\{ L' \in \mathbb{L} \mid L' \subsetneq L \}$. The set of prime languages in $\mathbb{L}$ is denoted by $Prm(\mathbb{L})$. Following [4], we define an RSFA as an SFA such that L_q is a residual language for each state $q \in Q$. An RSFA M is called *reduced* if $|Q| = |Prm(Res(L(M)))|$ and $L_q \in Prm(Res(L(M)))$ for each $q \in Q$.

Assuming that a MAT learning algorithm Λ for Ψ is available, Argyros and D'Antoni [2] have given a MAT learner for deterministic SFAs. Following their setting, we propose a learning algorithm for RSFAs, which pretends to be a MAT for instances of Λ and answers membership queries (MQs) and equivalence queries (EQs) using an observation table. An observation table $\mathcal{T} = (U, V, T)$ is filled by MQs in the usual way, where U is a prefix-closed set of words, V is a suffix-closed set, and T is a map $T\colon U \times V \to \{+, -\}$. For $u \in U$, define $\mathrm{row}(u) = \{ v \in V \mid T(uv) = + \}$. Note that, differently from [3], the domain of T is $U \times V$ rather than $(U \cup U\mathfrak{D}) \times V$.

Using $\mathcal{T} = (U, V, T)$, the proposed algorithm builds a hypothesis $\mathcal{H} = (\mathcal{A}, Q, Q_0, F, \Delta)$ with $Q = \{u \in U \mid \mathrm{row}(u) \in Prm(\{\mathrm{row}(u') \mid u' \in U\})\}$, $Q_0 = \{u \in Q \mid \mathrm{row}(u) \subseteq \mathrm{row}(\epsilon)\}$, $F = \{u \in Q \mid \epsilon \in \mathrm{row}(u)\}$. The transition relation Δ is obtained using $|Q|^2$ instances $\Lambda^{(u,u')}$ for all $u, u' \in Q$ of the learning algorithm Λ for Ψ. When $\Lambda^{(u,u')}$ asks an MQ on $a \in \mathfrak{D}$, we return $+$ if $\mathrm{row}(u') \subseteq \{v \in V \mid \mathrm{MQ}(uav) = +\}$ and $-$ otherwise. When $\Lambda^{(u,u')}$ asks an EQ on $\varphi \in \Psi$, we add (u, φ, u') to Δ. Our answer to this EQ is suspended and will be generated by analyzing the built automaton or a counterexample for an EQ on the automaton in the following procedure.

The algorithm checks the following three conditions on the built automaton before raising an EQ to ensure that the final output hypothesis will be a reduced RSFA.

- *Condition 1*: For $u, u' \in Q$ and $a \in \mathfrak{D}$, $\mathrm{row}(u) \subseteq \mathrm{row}(u')$ implies $\delta(u, a) \subseteq \delta(u', a)$.
- *Condition 2*: For $u \in U$ and $u' \in \delta(Q_0, u)$, we have $\mathrm{row}(u') \subseteq \mathrm{row}(u)$.
- *Condition 3*: For $u \in Q$ and $v \in V$, we have $v \notin \mathrm{row}(u)$ iff $v \notin L_u$.

Condition 1 is verified by checking whether $[\![\varphi \wedge \neg\varphi']\!] = \emptyset$ for $(u, \varphi, x), (u', \varphi', x) \in \Delta$ and $x \in Q$. When one of the three conditions is not satisfied, either a counterexample for $\Lambda^{(u,u')}$ is found and Δ is updated, or $\mathcal{T}$ is extended and all instances of Λ are discarded. Note that, in [3], the automaton derived from an observation table always satisfies essentially the same conditions as above, which ensures that their algorithm finally outputs the canonical RSA for the learning target, since their observation table has rows ua for all $u \in U$ and $a \in \mathfrak{D}$. On the other hand, our observation table does not have rows ua for all $a \in \mathfrak{D}$ since $\mathfrak{D}$ is too huge or infinite. This is why we need to introduce and check explicitly the above conditions.

When the hypothesis satisfies the three conditions above, the algorithm asks an EQ. We can prove that if the hypothesis passes the equivalence test, it is a reduced RSFA which accepts the target language. When a counterexample w is given by an EQ, we process the counterexample by the following procedure, which is a modification of the one in [2] for the non-deterministic setting. Firstly, we check if $\mathrm{MQ}(w) = -$ and there is $q \in Q_0$ with $\mathrm{MQ}(qw) = +$. If it is the case, q shall be removed from Q_0 by adding all suffixes of w to V. Otherwise, we have $\bigvee_{q \in Q_0} \mathrm{MQ}(qw) = \mathrm{MQ}(w)$

Table 1: The upper bound of EQs and MQs

		Deterministic	Residual
FA	EQ	n	$O(n^2)$
	MQ	$O(n^2\|\mathfrak{D}\| + n \log m)$ [5]	$O(n^3 m\|\mathfrak{D}\|)$ [3]
SFA	EQ	$O(n^3\mathcal{E}')$	$O(n^4\mathcal{E})$
	MQ	$O(n^4\mathcal{M}' + n^4\mathcal{E}' \log m)$[1] [2]	$O(n^6 m(\mathcal{M}+\mathcal{E}))$

and we can find $u, v \in \mathfrak{D}^*$, $a \in \mathfrak{D}$ and $q \in \delta(Q_0, u)$ such that $w = uav$ and $\mathrm{MQ}(qav) \neq \bigvee_{q' \in \delta(q,a)} \mathrm{MQ}(q'v)$. By checking a-moves from state q, either a counterexample for $\Lambda^{(q,q')}$ is found and Δ is updated, or $\mathcal{T}$ is extended and all instances of Λ are discarded.

We introduce some parameters to evaluate the query complexity of our algorithm. For the learning target L_* and $L \in Res(L_*)$, let $\Gamma_L = \{\{a \in \mathfrak{D} \mid L' \subseteq a^{-1}L\} \mid L' \in Res(L_*)\}$. For a subset $D \subseteq \mathfrak{D}$, we assume that Λ requires $\mathcal{C}_{\mathrm{EQ}}(D)$ EQs and $\mathcal{C}_{\mathrm{MQ}}(D)$ MQs to learn D. The set of denotations of predicates that may appear in an automaton built by the learner during the learning process is $\Phi = \{\bigcup_{D \in S} D \mid S \subseteq \Gamma_L \text{ for } L \in Res(L_*)\}$.

Theorem 1. Let $\mathcal{E} = \max_{D \in \Phi} \mathcal{C}_{\mathrm{EQ}}(D)$, $\mathcal{M} = \max_{D \in \Phi} \mathcal{C}_{\mathrm{MQ}}(D)$, $n = |Res(L_*)|$, and m be the length of the biggest counterexample returned by an EQ. Then, the proposed algorithm will learn a reduced RSFA for L_* using Λ with $O(n^4\mathcal{E})$ EQs and $O(n^6 m(\mathcal{E}+\mathcal{M}))$ MQs.

A query complexity comparison among previous algorithms and our algorithm for related classes of automata is shown in Table 1. The query complexity of the proposed algorithm is higher than that of the SFA learning algorithm in [2], especially for MQs. This is indeed the case when comparing the learning of (non-symbolic) RFAs and DFAs but in practice learning RFAs requires less queries than learning DFAs [3]. We expect it would be the case in the learning of RSFAs and SFAs in practice. It is future work to implement our algorithm and evaluate how many queries are required in practice.

[1] $\mathcal{E}' = \max_{D \in \Phi'} \mathcal{C}_{\mathrm{EQ}}(D)$ and $\mathcal{M}' = \max_{D \in \Phi'} \mathcal{C}_{\mathrm{MQ}}(D)$, where $\Gamma'_L = \{\{a \in \mathfrak{D} \mid L' = a^{-1}L\} \mid L' \in Res(L_*)\}$ and $\Phi' = \{\bigcup_{D \in S} D \mid S \subseteq \Gamma'_L \text{ for } L \in Res(L_*)\}$

REFERENCES

1. Dana Angluin. 1987. Learning Regular Sets from Queries and Counterexamples. *Information and Computation* 75, 2 (1987), 87–106.
2. George Argyros and Loris D'Antoni. 2018. The Learnability of Symbolic Automata. In *Computer Aided Verification. CAV 2018*. 427–445.
3. Benedikt Bollig, Peter Habermehl, Carsten Kern, and Martin Leucker. 2009. Angluin-Style Learning of NFA. In *IJCAI*. 1004–1009.
4. François Denis, Aurélien Lemay, and Alain Terlutte. 2002. Residual finite state automata. *Fundamenta Informaticae* 51, 4 (2002), 339–368.
5. Ronald L. Rivest and Robert E. Schapire. 1993. Inference of Finite Automata Using Homing Sequences. *Information and Computation* 103, 2 (1993), 299–347.
6. Margus Veanes, Peli de Halleux, and Nikolai Tillmann. 2010. Rex: Symbolic Regular Expression Explorer. In *Third International Conference on Software Testing, Verification and Validation, ICST 2010, Paris, France, April 7-9, 2010*. 498–507.