Theoretical Computer Science 650 (2016) 57–72

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

Learning regular omega languages

Dana Angluin [a,1], Dana Fisman [b,*,2]

[a] Yale University, United States
[b] University of Pennsylvania, United States

ARTICLE INFO

Keywords:
Active learning
Language inference
Infinitary languages
Membership queries
Equivalence queries
L^*
Büchi automata

ABSTRACT

We provide an algorithm for learning an unknown regular set of infinite words using membership and equivalence queries. Three variations of the algorithm learn three different canonical representations of regular omega languages using the notion of families of DFAs. One is of size similar to $L_\$$, a DFA representation recently learned using L^* by Farzan et al. The second is based on the syntactic FORC, introduced by Maler and Staiger. The third is introduced herein. We show that the second and third can be exponentially smaller than the first, and the third is at most as large as the second, with up to a quadratic saving with respect to the second.

1. Introduction

The L^* algorithm learns an unknown regular language in polynomial time using membership and equivalence queries [2]. It has proved useful in many areas including AI, neural networks, geometry, data mining, verification and many more. Some of these areas, in particular verification, call for an extension of the algorithm to regular ω-languages, i.e., regular languages over infinite words.

Regular ω-languages are the main means to model reactive systems and are used extensively in the theory and practice of formal verification and synthesis. The question of learning regular ω-languages has several natural applications in this context. For instance, a major critique of *reactive-system synthesis*, the problem of synthesizing a reactive system from a given temporal logic formula, is that it shifts the problem of implementing a system that adheres to the specification in mind to formulating a temporal logic formula that expresses it. A potential customer of a computerized system may find it hard to specify his requirements by means of a temporal logic formula. Instead, he might find it easier to provide good and bad examples of ongoing behaviors (or computations) of the required system, or classify a given computation as good or bad — a classical scenario for interactive learning of an unknown language using membership and equivalence queries.

Another example concerns *compositional reasoning*, a technique aimed to improve scalability of verification tools by decomposing the original verification task into subproblems. The simplification is typically based on the assume-guarantee reasoning principles and requires identifying adequate environment assumptions for components. A recent approach to the automatic derivation of assumptions uses L^* [1,6,18] with a model checker to answer queries to the teacher. Using L^* allows learning only *safety* properties (a subset of ω-regular properties that state that something bad has not happened and that

* Corresponding author.
E-mail address: dana.fisman@gmail.com (D. Fisman).
1 Research of this author was supported by US NSF grant CCF-0916389.
2 Research of this author was supported by US NSF grant CCF-1138996.

http://dx.doi.org/10.1016/j.tcs.2016.07.031

can be expressed by automata on finite words). To learn *liveness* and *fairness* properties, we need to extend L^* to the full class of regular ω-languages — a problem considered open for many years [12].

The first issue confronted when extending to ω-languages is how to cope with infinite words. Some finite representation is needed. There are two main approaches for that: one considers only finite prefixes of infinite computations and the other considers ultimately periodic words, i.e., words of the form uv^ω where v^ω stands for the infinite concatenation of v to itself. It follows from McNaughton's theorem [16] that two ω-regular languages are equivalent if they agree on the set of ultimately periodic words, justifying their use for representing examples.

Work by de la Higuera and Janodet [7] gives positive results for polynomially learning in the limit *safe* regular ω-languages from *prefixes*, and negative results for learning any strictly subsuming class of regular ω-languages from prefixes. A regular ω-language L is *safe* if for all $w \notin L$ there exists a prefix u of w such that no extension of u is in L. This work is extended in [9] to learning bi-ω languages from subwords.

Saoudi and Yokomori [20] consider ultimately periodic words and provide an algorithm for learning in the limit the class of *local* ω-languages and what they call *recognizable* ω-languages. An ω-language is said to be *local* if there exist $I \subseteq \Sigma$ and $C \subseteq \Sigma^2$ such that $L = I\Sigma^\omega - \Sigma^* C \Sigma^\omega$. An ω-language is referred to as *recognizable* [20] if it is recognizable by a deterministic automaton all of whose states are accepting.

Maler and Pnueli [14] provide an extension of the L^* algorithm, using ultimately periodic words as examples, to the class of regular ω-languages which are recognizable by both deterministic Büchi and deterministic co-Büchi automata. This is the subset for which the straightforward extension of right congruence to infinite words gives a Myhill–Nerode characterization [21]. Generalizing this to wider classes calls for finding a Myhill–Nerode characterization for larger classes of regular ω-languages. This direction of research was taken in [11,15] and is one of the starting points of our work.

In fact the full class of regular ω-languages can be learned using the result of Calbrix, Nivat and Podelski [5]. They define for a given ω-language L the set $L_\$ = \{u\$v \mid u \in \Sigma^*, v \in \Sigma^+, uv^\omega \in L\}$ and show that $L_\$$ is regular by constructing an NFA and a DFA accepting it. Since DFAs are canonical for regular languages, it follows that a DFA for $L_\$$ is a canonical representation of L. Such a DFA can be learned by the L^* algorithm provided the teacher's counterexamples are ultimately periodic words, given e.g. as a pair (u, v) standing for uv^ω — a quite reasonable assumption that is common to the other works too. This DFA can be converted to a Büchi automaton recognizing it. This approach was studied and implemented by Farzan et al. [8]. For a Büchi automaton with m states, Calbrix et al. provide an upper bound of $2^m + 2^{2m^2+m}$ on the size of a DFA for $L_\$$.

So the full class of regular ω-languages can be learnt using membership and equivalence queries, yet not very efficiently. We thus examine an alternative canonical representation of the full class of regular ω-languages. Maler and Staiger [15] show that regular ω-languages can be represented by a family of right congruences (FORC, for short). With a given ω-language they associate a particular FORC, the *syntactic* FORC, which they show to be the coarsest FORC recognizing the language. We adapt and relax the notion of FORC to families of DFAS (FDFA, for short). We show that the syntactic FORC can be factorially smaller than $L_\$$. That is, there exists a family of languages L_n for which the syntactic FDFA is of total size $O(n^2)$ and the minimal DFA for $L_\$$ is of size $\Omega(n!)$. We then provide a third representation, the *recurrent* FDFA. We show that the recurrent FDFA is at most as large the syntactic FDFA, with up to a quadratic saving with respect to the syntactic FDFA.

We provide a learning algorithm L^ω that can learn an unknown regular ω-language using membership and equivalence queries. The learned representations use the notion of families of DFAS (FDFAS). Three variations of the algorithm can learn the three canonical representations: the periodic FDFA (the FDFA corresponding to $L_\$$), the syntactic FDFA (the FDFA corresponding to the syntactic FORC) and the recurrent FDFA. The running time of the three learning algorithms is polynomial in the size of the periodic FDFA.

2. Preliminaries

Let Σ be a finite set of symbols. The set of finite words over Σ is denoted Σ^*, and the set of infinite words, termed ω-words, over Σ is denoted Σ^ω. A *language* is a set of finite words, that is, a subset of Σ^*, while an ω-language is a set of ω-words, that is, a subset of Σ^ω. Throughout the paper we use u, v, x, y, z for finite words, w for ω-words, a, b, c for letters of the alphabet Σ, and i, j, k, l, m, n for natural numbers. We use $[i..j]$ for the set $\{i, i+1, \ldots, j\}$. We use $w[i]$ for the i-th letter of w and $w[i..k]$ for the subword of v starting at the i-th letter and ending at the k-th letter, inclusive.

An *automaton* is a tuple $M = \langle \Sigma, Q, q_0, \delta \rangle$ consisting of a finite alphabet Σ of symbols, a finite set Q of states, an initial state q_0 and a transition function $\delta : Q \times \Sigma \to 2^Q$. A run of an automaton on a finite word $v = a_1 a_2 \ldots a_n$ is a sequence of states $q_0, q_1, \ldots, q_n$ such that $q_{i+1} \in \delta(q_i, a_{i+1})$. A run on an infinite word is defined similarly and results in an infinite sequence of states. The transition function can be extended to a function from $Q \times \Sigma^*$ by defining $\delta(q, \lambda) = q$ and $\delta(q, av) = \delta(\delta(q, a), v)$ for $q \in Q$, $a \in \Sigma$ and $v \in \Sigma^*$. We often use $M(v)$ as a shorthand for $\delta(q_0, v)$ and $|M|$ for the number of states in Q. A transition function is *deterministic* if $\delta(q, a)$ is a singleton for every $q \in Q$ and $a \in \Sigma$, in which case we use $\delta(q, a) = q'$ rather than $\delta(q, a) = \{q'\}$.

By augmenting an automaton with an acceptance condition α, obtaining a tuple $\langle \Sigma, Q, q_0, \delta, \alpha \rangle$, we get an *acceptor*, a machine that accepts some words and rejects others. An acceptor accepts a word if one of the runs on that word is accepting. For finite words the acceptance condition is a set $F \subseteq Q$ and a run on v is accepting if it ends in an accepting state, i.e., if $\delta(q_0, v) \in F$. For infinite words, there are many acceptance conditions in the literature; here we mention three: Büchi, co-Büchi and Muller. Büchi and co-Büchi acceptance conditions are also a set $F \subseteq Q$. A run of a Büchi automaton