Efficient Learning of Real Time One-Counter Automata

Amr F. Fahmy and Robert S. Roos

1 (Contact author.) Aiken Computation Lab, Harvard University, Cambridge MA 02138, USA (amrfdas.harvard.edu). Research supported in part by ARPA contract no. F19628-92-C-0113 and by NSF grant CDA-9308833.

2 Department of Computer Science, Smith College, Northampton MA 01063, USA (roos@cs.smith.edu).

Abstract. We present an efficient learning algorithm for languages accepted by deterministic real time one-counter automata (ROCA). The learning algorithm works by first learning an initial segment, $B_n$, of the infinite state machine that accepts the unknown language and then decomposing it into a complete control structure and a partial counter. A new, efficient ROCA decomposition algorithm, which will be presented in detail, allows this result. The decomposition algorithm works in time $O(n^2 \log(n))$ where $nc$ is the number of states of $B_n$ for some language-dependent constant $c$. If Angluin's algorithm for learning regular languages is used to learn $B_n$ and the complexity of this step is $h(n, m)$, where $m$ is the length of the longest counterexample necessary for Angluin's algorithm, the complexity of our algorithm is $O(h(n, m) + n^2 \log(n))$.

1 Introduction

We present an efficient learning algorithm for languages accepted by deterministic real time one-counter automata (ROCA). The learning algorithm works by first learning an initial segment, $B_n$, of the infinite state machine that accepts the unknown language $L$ and then decomposing it into a complete control structure and a partial counter. A new, efficient ROCA decomposition algorithm, which will be presented in detail, allows this result. The decomposition algorithm works in $O(n^2 \log(n))$ where $nc$ is the number of states of $B_n$, $c$ being an upper bound on the number of states in a control structure that accepts $L$. The minimum value of $n$ needed to achieve decomposition could be exponentially larger than $c$ for some ROCA languages, but we claim that this $n$ is a more natural measure of the complexity of the language $L$.

If Angluin's algorithm for learning regular languages (appropriately modified) is used to construct $B_n$, and the complexity of this step is $h(n, m)$, where $m$ is the length of the longest counterexample necessary for Angluin's algorithm, the complexity of our algorithm is $O(h(n, m) + n^2 \log(n))$.

Roos and Berman [2] and Roos [8] were the first to find a polynomial time algorithm for the exact learning of deterministic one-counter automata (DOCA) as defined by Valiant and Paterson in [9]. The polynomial is of large degree, thus motivating this work to find a practical algorithm. (The differences between ROCA and
DOCA are described in Section 3.) Fahmy and Biermann [4] and Fahmy [5] introduced the idea of learning by automata decomposition. The method is applicable to a very wide class of real time languages using a variety of data structures such as counters, stacks, queues, and double counters; however, the algorithms they present are of exponential time in the worst case. The definitions of control structures, data structures, and behavior graphs that we use here and the relations between them were first given in [3] and subsequently in [4].

A discussion and an example of the learning process will be presented in Section 2. Following this, definitions of the control structure, the counter, the behavior graph, and the relations between them are given in Section 3. In Section 4 the decomposition theorem and decomposition algorithm will be presented.

2 The Learning Algorithm

We will present the learning process for a ROCA language using an example.

Consider the language \( L = \{a^n ccb d^i \mid n > 0\} \cup \{a^n d \mid n > 0\} \). This language is not regular and is accepted by an infinite state machine that we call the behavior graph (b.g.), denoted by \( B \). An “initial segment” of \( B \)—the submachine of \( B \) induced by all the states that are distance \( n \) or less from the initial state of \( B \)—will be denoted by \( B_n \). The b.g. for our example appears in Figure 1.

A ROCA \( A = (C, D) \) for \( L \) is a pair of state machines that also accepts \( L \). \( C \) is a finite state machine called the control structure (c.s.), and \( D \), called the data structure (d.s.), is an infinite state machine that simulates a counter. State diagrams for a counter and c.s. that accept \( L \) also appear in Figure 1. \( L \) is accepted by \( A \) in the following manner. Input symbols are read by \( C \) which, using the symbol, its current state, and the state of the counter, changes its state. While it is changing its state it sends a single instruction to the counter which it uses to change its state, too. The triples \( \{\text{sym, val, instr}\} \) appear on the transitions of the c.s. in Figure 1 where \( \text{sym} \) is the input symbol, \( \text{val} \) is 0 if the counter state is 0 and \(-\infty\) otherwise, and \( \text{instr} \) is the instruction sent to the counter. If the final symbol of an input string causes \( C \) to end up in a final state then we say that the ROCA has accepted the input string.

The learning process for a ROCA language \( L \) with behavior graph \( B \) starts by constructing a machine that contains \( B_n \), for some natural number \( n \). This is done using a slight modification of Angluin’s learning algorithm for regular languages. One of our modifications to Angluin’s “minimally adequate teacher” permits the ability to test the equivalence of a given submachine (the learner’s guess for the initial segment \( B_n \)) with the behavior graph of a ROCA language \( L \). We note that Angluin’s algorithm will not terminate if the state machine that accepts \( L \) has an infinite number of states (as in the case of ROCAs). Thus, we will assume that the teacher will choose a suitable depth \( n \) and request that the learner decompose \( B_n \) after constructing it using Angluin’s algorithm. (See remarks below on removing this requirement.) In addition, we assume that our teacher is “helpful” in the sense of fully exercising the counter; in other words, the teacher’s counterexamples will enable the full graph \( B_n \) to be constructed. These assumptions simplify our presentation; our algorithm makes no further use of the teacher once \( B_n \) has been constructed. If