Improving Model Inference of Black Box Components having Large Input Test Set

Muhammad Naeem Irfan IRFAN@IMAG.FR
Roland Groz GROZ@IMAG.FR
Catherine Oriat ORIAT@IMAG.FR
LIG, Computer Science Lab, Grenoble Institute of Technology, France

Editors: Jeffrey Heinz, Colin de la Higuera and Tim Oates

Abstract

The deterministic finite automata (DFA) learning algorithm L^* has been extended to learn Mealy machine models which are more succinct for *input/output* (i/o) based systems. We propose an optimized learning algorithm L_1 to infer Mealy models of software black box components. The L_1 algorithm uses a modified observation table and avoids adding unnecessary elements to its columns and rows. The proposed improvements reduce the worst case time complexity. The L_1 algorithm is compared with the existing Mealy inference algorithms and the experiments conducted on a comprehensive set confirm the gain.

Keywords: black box; Mealy machine; model inference; testing;

1. Introduction

For software black box components, models can be learned from interactions with components, available specifications, knowledge of experts and other such sources. Automatically learning and testing models from interactions with software components is a well known approach. Software models help to steer testing of such components. The L^* algorithm (Angluin (1987)) learns unknown models as a DFA. It has been adapted formally (Niese (2003)) to infer black box implementations as Mealy machines.

In this paper, we propose an optimized Mealy inference algorithm L_1. To record the observations, the L_1 algorithm uses an observation table and initially keeps its columns empty. The columns of the table are only populated to process counterexamples. The rows of the table keep only the sequences which correspond to valid states. The paper is organized as follows: Section 2 introduces the basic definitions and notations. In Section 3, the Mealy inference algorithm L_1 is formally presented. Final section concludes the paper.

2. Preliminaries

Definition 1 *A Mealy Machine is a sextuple* $(Q, I, O, \delta, \lambda, q_0)$, *where* Q *is the non-empty finite set of states,* $q_0 \in Q$ *is the initial state,* I *is the non-empty finite set of input symbols,* O *is the non-empty finite set of output symbols,* $\delta : Q \times I \to Q$ *is the transition function that maps pairs of states and input symbols to the corresponding next states,* $\lambda : Q \times I \to O$ *is the output function that maps pairs of states and input symbols to the corresponding output symbols. We assume* $dom(\delta) = dom(\lambda) = Q \times I$, *i.e. the Mealy machine is input enabled.*

A word ω is a sequence of inputs $i_1 i_2 \ldots i_n \in I^*$ and the empty word is denoted by ϵ. The size of a word is the number of letters that ω contains and it is denoted by $|\omega|$. The transition function for ω is extended as $\delta(q_0, \omega) = \delta(\ldots \delta(\delta(q_0, i_1), i_2) \ldots, i_n)$ and the output function for ω is extended as $\lambda(q_0, \omega) = \lambda(\ldots \lambda(\lambda(q_0, i_1), i_2) \ldots, i_n)$. If a non-empty word $\omega = u \cdot v$ then u and v are *prefix* and *suffix* of ω, respectively. Let $\mathit{suffix}^j(\omega)$ denote the *suffix* of a word ω of length j and $\mathit{prefix}^j(\omega)$ denotes the *prefix* of a word ω of length j, where $j \in \{1, 2, \ldots |\omega|\}$. Let $\mathit{output}^j(\omega)$ denote the output for jth input symbol in ω. For example, if we have a sequence of outputs *0010* for a word *aaba* then $\mathit{output}^3(\mathit{aaba}) = \mathit{1}$. A set is *prefix closed* iff all the prefixes of every element of the set also belong to the set. A set is *suffix closed* iff all the suffixes of every element of the set also belong to the set. If a word ω is executed on a machine and a state s_ω is reached then ω is the *access string* for s_ω, it is a *valid access string* if s_ω corresponds to a state and *invalid access string*, otherwise.

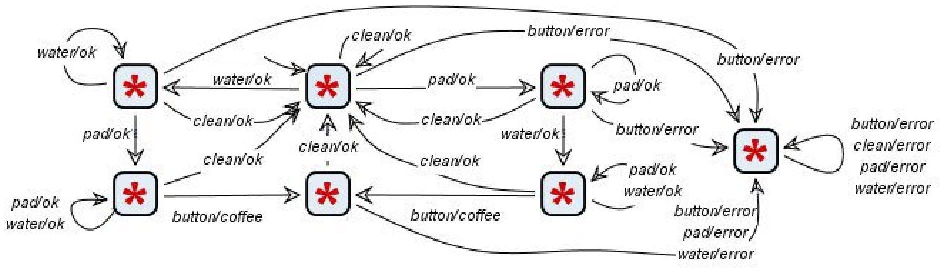

Figure 1: Coffee machine with $I = \{\mathit{pad}, \mathit{water}, \mathit{button}, \mathit{clean}\}$

3. Optimized Mealy Inference Algorithm

The Mealy adaptations of L^* learn a target model by asking *output queries* (Niese (2003); Shahbaz and Groz (2009)) which fill the observation table with output strings. The columns of the table are initialized with the inputs of I. This enables the algorithm to detect i/o to annotate the edges in Mealy models. The rows of the table are labeled with access strings of the states of a target model and the columns are strings that distinguish these states.

If the input set I has a large number of symbols and the columns of the table are initialized with I, there is a strong possibility that the columns are initialized with too many symbols and this results in increased number of output queries. The L_1 algorithm initially keeps the columns of the table empty with the intent to add only those elements from the set I which are really required. To enable L_1 to calculate the output labels for the transition edges of a conjectured Mealy model, we record the output for the last input symbol of the access strings that label rows of the table.

We assume that the target system is deterministic and input enabled. To attain this, the Mealy adaptations use the collection of all possible inputs I for each state. While inferring models, when an access string of a new state is identified then its one letter extensions are added to the table to identify its successor states. But for software applications valid inputs for every state are smaller than I. For L_1 the output recorded along access strings helps to identify that an access string is valid or invalid, and only valid access strings are kept in the table. Invalid access strings can be kept in the table but they are excluded from subsequent tests. To process a counterexample L_1 adds the suffixes of the counterexample by increasing length to the columns of the table until the table is not closed (Irfan et al. (2010)). Since L_1 does not initialize the columns with I and adds only those elements from I which are really required, the gain with the L_1 algorithm increases with increase in $|I|$ for a target black box model. The table used by L_1 is described as follows.

3.1. Observation Table

The observation table is defined as a quadruple (S, E, L, T), where $S \subseteq I^*$ is a prefix closed non-empty finite set of access strings and its elements label the rows of the table, $E \subseteq I^+$ is a suffix closed finite set, which labels the columns of the table, for $S' = S \cup S \cdot I$, the finite function T maps $S' \times E$ to outputs O^+, and the finite function L maps $S' \backslash \{\epsilon\}$ to outputs O. The access strings are concatenated with the distinguishing sequences to construct the output queries as $s \cdot e$[1], for all $s \in S'$ and $e \in E$. In the table, $\forall s \in S'$, $\forall e \in E$, $T(s,e) = \mathit{suffix}^{|e|}(\lambda(q_0, s \cdot e))$, and $L(s) = \mathit{output}^{|s|}(\lambda(q_0, s \cdot e))$. By means of function L, all the access strings $s \in S' \backslash \{\epsilon\}$ labeling the rows of the table contain the output o for the last input symbol of s and S' is a prefix closed set, this implies that $\lambda(q_0, s)$ can be calculated from these outputs. Thus, recording the suffix of the output query answer $\mathit{suffix}^{|e|}(\lambda(q_0, s \cdot e))$ to the cell labeled by row s and column e in the table is sufficient. Initial table (S, E, L, T) for Mealy inference of the machine in Figure 1 is presented in Table 1.

Table 1: Initial Observation Table for the Mealy machine in Figure 1.

		E
		$\emptyset$
S	ϵ	
$S \cdot I \backslash S$	*pad/ok* *water/ok* ~~*button/error*~~ *clean/ok*	

The equivalence of rows in the table is defined with the help of the function T. Two rows $s_1, s_2 \in S'$ are said to be equivalent, iff $\forall e \in E$, $T(s_1, e) = T(s_2, e)$, and it is denoted as $s_1 \cong s_2$. For every row $s \in S'$, the equivalence class of a row s is denoted by $[s]$. To construct a conjecture, L_1 requires the table to satisfy the closure and compatibility properties. The table is closed, if $\forall s_1 \in S \cdot I \backslash S$, there exists $s_2 \in S$ such that $s_1 \cong s_2$. The table is compatible, if whenever two rows $s_1 \cong s_2$ for $s_1, s_2 \in S$ then $s_1 \cdot i \cong s_2 \cdot i$ for $\forall i \in I$. Since the rows size $|S|$ of the table increases only to make it closed, L_1 always maintains the condition, for all $s_1, s_2 \in S$, $s_1 \ncong s_2$. Thus, the compatibility condition is always trivially satisfied. When the table is closed and compatible, a conjecture is defined as follows.

Definition 2 *Let (S, E, L, T) be a closed and compatible observation table then the Mealy machine conjecture $Conj_1 = (Q_C, I, O, \delta_C, \lambda_C, q_{0C})$ is defined, where*

- $Q_C = \{[s] | s \in S\}$, *(* $\forall s_1, s_2 \in S$, *always* $s_1 \ncong s_2$, *thus,* $|S| = |Q_C|$*)*
- $q_{0C} = [\epsilon]$, $\epsilon \in S$ *is the initial state of the conjecture*
- $\delta_C([s], i) = [s \cdot i]$, *for all* $s \in S, i \in I$
- $\lambda_C([s], i) = L(s \cdot i), s \in S, i \in I$ $\exists! s \cdot i \in S'$

1. $s \cdot e$ for $s \in S', e \in E$ and $s \cdot i$ for $s \in S', i \in I$ are constructed by considering the access strings only.

Theorem 3 *If (S, E, L, T) is a closed and compatible observation table then the Mealy conjecture $Conj_1$ from (S, E, L, T) is consistent with the finite function T. Any other conjecture consistent with T but inequivalent to $Conj_1$ must have more states.*

```
Input: Black box and set of inputs $I$
Output: Mealy Machine Conjecture
begin
    initialize the rows $S = \{\epsilon\}$, and columns $E = \emptyset$;
    execute output queries for $S \cdot I$ strings;
    construct a conjecture $C$; repeat
        search counterexamples;
        if the oracle replies with a counterexample $CE$ then
            while $CE$ is a counterexample do
                for $j = 1$ to $|CE|$ do
                    if $suffix^j(CE) \notin E$ then
                        add $suffix^j(CE)$ to $E$;
                        complete $(S, E, T, L)$ by asking output queries $s \cdot e$ such that $s \in S' \wedge e \in E$;
                        if $(S, E, T, L)$ is not closed then
                            break for loop;
                        end
                    end
                end
                while $(S, E, L, T)$ is not closed do
                    find $s_1 \in S \cdot I \backslash S$ such that $s_1 \not\cong s_2$, for all $s_2 \in S$;
                    move $s_1$ to $S$;
                    add valid $s_1 \cdot i$ to $S \cdot I$, for all $i \in I$;
                    complete table by asking output queries for new added rows;
                end
                construct a conjecture $C$;
            end
        end
    until oracle says yes to the conjecture $C$;
end
```

Algorithm 1: The Algorithm L_1

We explain the L_1 algorithm with the help of an example in Appendix B and show some experiments for learning random machines in Appendix A.

4. Conclusion

L_1 restricts the table columns to distinguishing sequences and their suffixes, and the table rows to access strings of distinct states and their valid one letter extensions. These improvements bring down the theoretical and practical time complexity of the learning algorithm. The theoretical worst case time complexity for the L_1 algorithm is $O(|I|mn^2)$. Real world systems work on huge data sets as their inputs, thus, gain with L_1 is obvious for inferring models of such systems.

References

Dana Angluin. Learning regular sets from queries and counterexamples. *Inf. Comput.*, 75 (2):87–106, 1987.

Muhammad Naeem Irfan, Catherine Oriat, and Roland Groz. Angluin style finite state machine inference with non-optimal counterexamples. In *Proceedings of the First International Workshop on Model Inference In Testing*, MIIT '10, pages 11–19. ACM, 2010.

Oliver Niese. *An Integrated Approach to Testing Complex Systems.* PhD thesis, University of Dortmund, 2003.

Muzammil Shahbaz and Roland Groz. Inferring mealy machines. In *FM*, pages 207–222, 2009.

Bernhard Steffen, Falk Howar, and Maik Merten. Introduction to active automata learning from a practical perspective. In *SFM*, pages 256–296, 2011.

Appendix A. Experiments

Shahbaz and Groz (2009) show that their Mealy adaptation algorithm ${L_M}^+$ outperforms Mealy inference with rest of the counterexample processing methods. We have performed an experimental evaluation to compare L_1 and ${L_M}^+$.

We use random machines, which allow to study the influence of the various parameters (number of inputs, states, etc) on the learning algorithms. For both of the following sets of experiments in order to increase our confidence, we repeat the learning for every target machine for 30 times and average on the calculated data. We record the number of output queries to analyze the learning algorithms for two sets of experiments. We have simulated an oracle so that the algorithms can ask the equivalence queries for the correctness of conjectured models.

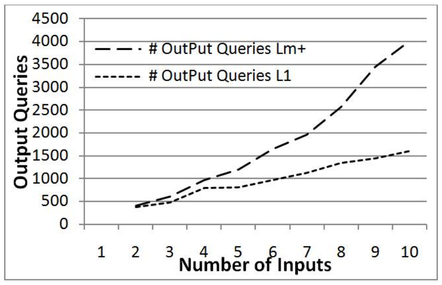

Figure 2: $|I|\in\{2, 3 \dots 10\}|O|=5$ & $n=40$

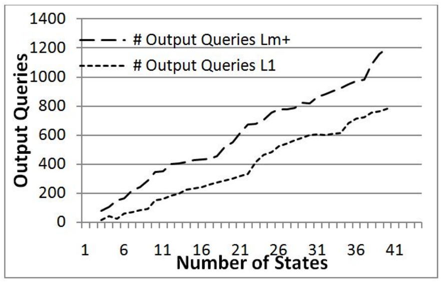

Figure 3: $|I|=5$,$|O|=7$ & $n\in\{3, 4 \dots 40\}$

On average L_1 requires 16 output queries to learn the machine with states size $n = 3$ and ${L_M}^+$ requires 80 output queries. So there is a gain of 80% for output queries. For the largest machine having states size $n = 40$, L_1 requires 784 output queries and ${L_M}^+$ requires 1202 output queries, there is a gain of 34.78% output queries. Thus, clearly L_1 outperforms ${L_M}^+$ for both of the considered sets of experiments.

Appendix B. Example

We illustrate the L_1 algorithm by learning the Mealy model $\mathcal{M}_{coffee}$ of the coffee machine presented in Figure 1 and show how invalid access strings can be excluded from subsequent tests using the output for the last input of the access strings.

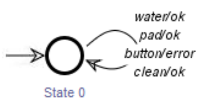

Figure 4: $Conj_{coffee_1}$ from Table 1

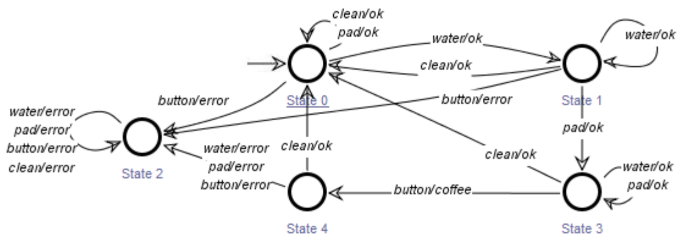

Figure 5: $Conj_{coffee_2}$ from Table 6(c)

The oracle replies with a counterexample $water \cdot pad \cdot button$ for $Conj_{coffee_1}$ in Figure 4. L_1 adds the smallest suffix $button$ of the counterexample to the columns. The observation table remains closed so the suffix $pad \cdot button$ is added, which makes the table unclosed. To make the table closed the row $water$ is moved to S as presented in Table 6(a).

	$button$	$pad \cdot button$
ϵ	$error$	$ok \cdot error$
$water/ok$	$error$	$ok \cdot coffee$
pad/ok	$error$	$ok \cdot error$
$clean/ok$	$error$	$ok \cdot error$
$water \cdot water/ok$	$error$	$ok \cdot coffee$
$water \cdot pad/ok$	$coffee$	$ok \cdot coffee$
~~$water \cdot button$~~$/error$	$error$	
$water \cdot clean/ok$	$error$	$ok \cdot error$

(a) Close the table by moving $water$ to S.

	$button$	$pad \cdot button$
ϵ	$error$	$ok \cdot error$
$water/ok$	$error$	$ok \cdot coffee$
$water \cdot pad/ok$	$coffee$	$ok \cdot coffee$
pad/ok	$error$	$ok \cdot error$
$clean/ok$	$error$	$ok \cdot error$
$water \cdot water/ok$	$error$	$ok \cdot coffee$
$water \cdot clean/ok$	$error$	$ok \cdot error$
$water \cdot pad \cdot water/ok$	$coffee$	$ok \cdot coffee$
$water \cdot pad \cdot pad/ok$	$coffee$	$ok \cdot coffee$
$water \cdot pad \cdot button/coffee$	$error$	$error \cdot error$
$water \cdot pad \cdot clean/ok$	$error$	$ok \cdot error$

(b) Close the table by moving $water \cdot pad$ to S.

	$button$	$pad \cdot button$
ϵ	$error$	$ok \cdot error$
$water/ok$	$error$	$ok \cdot coffee$
$water \cdot pad/ok$	$coffee$	$ok \cdot coffee$
$water \cdot pad \cdot button/coffee$	$error$	$error \cdot error$
pad/ok	$error$	$ok \cdot error$
$clean/ok$	$error$	$ok \cdot error$
$water \cdot water/ok$	$error$	$ok \cdot coffee$
$water \cdot clean/ok$	$error$	$ok \cdot error$
$water \cdot pad \cdot water/ok$	$coffee$	$ok \cdot coffee$
$water \cdot pad \cdot pad/ok$	$coffee$	$ok \cdot coffee$
$water \cdot pad \cdot clean/ok$	$error$	$ok \cdot error$
~~$water \cdot pad \cdot button \cdot water$~~$/error$	$error$	
~~$water \cdot pad \cdot button \cdot pad$~~$/error$	$error$	
~~$water \cdot pad \cdot button \cdot button$~~$/error$	$error$	
$water \cdot pad \cdot button \cdot clean/ok$	$error$	$ok \cdot error$

(c) Move $water \cdot pad \cdot button$ to S.

Table 2: Processing the counterexample $water \cdot pad \cdot button$.

	$button$	$pad \cdot button$	$water \cdot button$
ϵ	$error$	$ok \cdot error$	$ok \cdot error$
$water/ok$	$error$	$ok \cdot coffee$	$ok \cdot error$
$water \cdot pad/ok$	$coffee$	$ok \cdot coffee$	$ok \cdot coffee$
$water \cdot pad \cdot button/coffee$	$error$	$error \cdot error$	$error \cdot error$
pad/ok	$error$	$ok \cdot error$	$ok \cdot coffee$
$clean/ok$	$error$	$ok \cdot error$	$ok \cdot error$
$water \cdot water/ok$	$error$	$ok \cdot coffee$	$ok \cdot error$
$water \cdot clean/ok$	$error$	$ok \cdot error$	$ok \cdot error$
$water \cdot pad \cdot water/ok$	$coffee$	$ok \cdot coffee$	$ok \cdot coffee$
$water \cdot pad \cdot pad/ok$	$coffee$	$ok \cdot coffee$	$ok \cdot coffee$
$water \cdot pad \cdot clean/ok$	$error$	$ok \cdot error$	$ok \cdot error$
$water \cdot pad \cdot button \cdot clean/ok$	$error$	$ok \cdot error$	$ok \cdot error$

(d) After adding $water \cdot button$ to E.

	$button$	$pad \cdot button$	$water \cdot button$
ϵ	$error$	$ok \cdot error$	$ok \cdot error$
$water/ok$	$error$	$ok \cdot coffee$	$ok \cdot error$
$water \cdot pad/ok$	$coffee$	$ok \cdot coffee$	$ok \cdot coffee$
$water \cdot pad \cdot button/coffee$	$error$	$error \cdot error$	$error \cdot error$
pad/ok	$error$	$ok \cdot error$	$ok \cdot coffee$
$clean/ok$	$error$	$ok \cdot error$	$ok \cdot error$
$water \cdot water/ok$	$error$	$ok \cdot coffee$	$ok \cdot error$
$water \cdot clean/ok$	$error$	$ok \cdot error$	$ok \cdot error$
$water \cdot pad \cdot water/ok$	$coffee$	$ok \cdot coffee$	$ok \cdot coffee$
$water \cdot pad \cdot pad/ok$	$coffee$	$ok \cdot coffee$	$ok \cdot coffee$
$water \cdot pad \cdot clean/ok$	$error$	$ok \cdot error$	$ok \cdot error$
$water \cdot pad \cdot button \cdot clean/ok$	$error$	$ok \cdot error$	$ok \cdot error$
$pad \cdot water/ok$	$coffee$	$ok \cdot coffee$	$ok \cdot coffee$
$pad \cdot pad/ok$	$error$	$ok \cdot error$	$ok \cdot coffee$
~~$pad \cdot button$~~$/error$	$error$		
$pad \cdot clean/ok$	$error$	$ok \cdot error$	$ok \cdot error$

(e) Moving the row pad to S.

Table 3: Processing the counterexample $pad \cdot pad \cdot water \cdot button$

We have taken the Coffee machine example used by Steffen et al. (2011), they ask 155 output queries to conjecture its model, whereas L_1 asks only 54 output queries.