Adaptive Localizer Based on Splitting Trees

Roland Groz, Adenilso Simao, Catherine Oriat

Adaptive localizer based on splitting trees

Roland Groz[1], Adenilso Simao[2], and Catherine Oriat[1]

[1] Univ. Grenoble Alpes, CNRS, LIG, F-38000 Grenoble, France,
First.Last@imag.fr
[2] Universidade de São Paulo, Brasil, adenilso@icmc.usp.br

Abstract. When testing a black box system that cannot be reset, it may be useful to use a localizer procedure that will ensure that the test sequence goes at some point through a state that can be identified with a characterizing set of input sequences. In this paper, we propose a procedure that will localize when the separating sequences are organized in a splitting tree. Compared to previous localizing sequences based on characterization sets, using the tree structure one can define an adaptive localizer, and the complexity of localizing depends on the height of the tree instead of the number of states.

1 Introduction

In testing methods based on FSM models, it is important to ensure that the black box implementation under test is in a known state at key points after applying some prefix input sequence, and before applying a trailing sequence. The same is also true when the test is used to retrieve some kind of state model information from a black box, as in the inference problem.

Hennie [2] and Kohavi [4] introduced approaches based on locating sequences, to build checking sequences, that can test for conformance. More recently, we introduced a localizer procedure to infer a FSM model of a black box system which cannot be reset [1] with two key assumptions: first, an upper bound n on the number of states, and second a characterization set (aka W-set) for the system. Whatever the initial state of the system, the application of the localizer will make it possible to ascertain the state reached at the end of it, or more precisely before applying the last characterizing sequence. For $W = \{w_1, w_2, w_3\}$, the localizer's input sequence is $(w_1^{2n-1}w_2)^{2n-1}(w_1^{2n-1}w_3)$. The core trick, as already suggested by Hennie, is that after applying at most $n+1$ times a sequence, the machine must have entered a cycle, and we proved [1] that another $n-2$ applications were enough (in worst case) to identify where the machine is in the cycle, so as to predict the next application. After w_1^{2n-1} the answer of the machine to w_1 can be predicted, so by applying w_2 we can know its answers to both w_1 and w_2. Similarly, after $(w_1^{2n-1}w_2)^{2n-1}$ we know what its answers would be to w_1 AND w_2 after applying w_1^{2n-1}, so we now substitute w_3 and therefore we identify the state reached after $(w_1^{2n-1}w_2)^{2n-1}w_1^{2n-1}$. The length of the sequence is exponential in the cardinal of the characterization set.

In this paper, we propose an improved localizer when the characterizing sequences are organized in a splitting tree [5]. In that case, the length of the localizing sequence is exponential in the height of the tree, so it would reduce to linear in the number of sequences when the tree is balanced. The main difficulty lies in the fact that each input sequence from the W-set must be applied after repeating a fixed number of times $(2n-1)$ a fixed sub-localizing sequence to ensure that the next iteration would be a repeated situation identifiable from the previous observations. The localizer procedure from [1] uses a fixed order of input sequences from the W-set to ensure that the same sequence is repeated at each level. With a splitting tree, the sequence to repeat may not be fixed. In this paper, we show that it is possible to combine variable sequences with predictability: differing from [3], we allow cycles that hop through different states and levels up and down the tree. Our new localizer is adaptive, and it does not produce a fixed input sequence, but an adaptive one.

2 Definitions

In this paper, a FSM is assumed to be a *strongly connected* complete deterministic Mealy machine $M = (Q, I, O, \delta, \lambda)$, with finite state, input and output sets Q, I, O; δ and λ as transition and output mappings.

Two states $q, q' \in Q$ are distinguishable by a set $H \subset I^*$ if there exists a *separating sequence* $\gamma \in H$ such that $\lambda(q, \gamma) \neq \lambda(q', \gamma)$. An FSM is *minimal* if all states are pairwise distinguishable. A set W of sequences of inputs (therefore conventionally called a W-set) is a *characterization set* for an FSM M if each pair of states is distinguishable by W.

Figure 1 shows an FSM M_0, taken from [5], with 6 states, 2 inputs (a, b) and two outputs $(0, 1)$. $W = \{b, aab, ab, aaab\}$ is a characterization set for M_0.

A *localizer* w.r.t. a set $H \subset I^*$ and $n \in \mathbb{N}$ is a procedure that applies an input sequence $\Xi\gamma$ to an *unknown* machine with at most n states such that $\gamma \in H$ and the output responses to each sequence in H can be ascertained in the state reached after applying Ξ. Note that H is not required to be a characterization set, and we only assume we know a bound on the number of states.

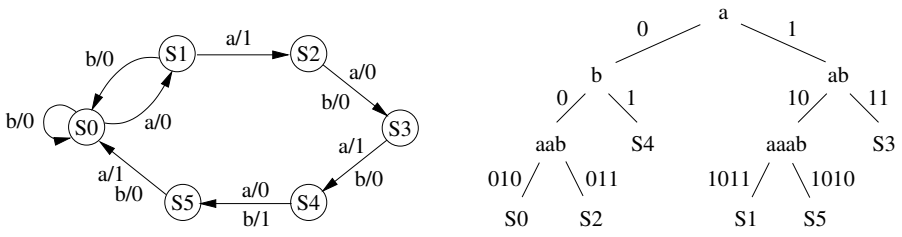

Fig. 1. FSM M_0 and a splitting tree T for it

Following [5], we now define splitting trees. A *splitting tree* for an FSM $M = (Q, I, O, \delta, \lambda)$ is a rooted tree T with a finite set of nodes such that:

- each non-leaf node N is labelled by a non-empty sequence of inputs $\gamma \in I^+$;
- each edge $e : N_1 \to N_2$ is labelled by a non-empty output sequence of outputs $\theta \in O^+$ of the same length as the label of the source node N_1.

A splitting tree T is separating for M if each node can be associated with a class of a partition of its states, such that:

- the root of T is associated with the full set of states Q of the machine M;
- let $e : N_1 \to N_2$ be an edge of T such that N_1 is labelled by $\gamma \in I^+$, e is labelled by $\theta \in O^+$. If N_1 is associated with $Q_1 \subseteq Q$, then N_2 is associated with $Q_2 = \{q \in Q_1 \ ; \ \lambda(q, \gamma) = \theta\}$.

A splitting tree is *fully separating* if each leaf node is associated with a single state. In that case, the input sequences from the root of the tree to a leaf separate the state on this leaf from all other states. The advantage of structuring a W-set as a splitting tree is that a state can be identified by applying less input sequences: with a W-set, we potentially have to apply $\operatorname{Card} W$ sequences, while with a splitting tree, we only have to apply $\log(\operatorname{Card} W)$ sequences, if the tree is balanced. As the localizer algorithm is exponential in $\operatorname{Card} W$, the gain from using splitting trees rather than W-sets is crucial. In figure 1, we choose an initial sequence $a \notin W$ that better balances the tree and splits on a short sequence.

Given a splitting tree T, and a node N of T, we will use the following notations:

- We denote the subtree rooted at node N also by N.
- $\text{root}(T)$ is the root node of T. Following the previous notation, this root node could also be denoted by T.
- $I(N)$ is the input sequence that labels node N.
- $\Lambda(N)$ is the output sequence that labels the edge *leading* to node N.
- Given a child node N' of N, $T(N \leftarrow N')$ is the modified tree where N has been replaced by the tree rooted at N'. In other words, the subtree rooted at N' is grafted one level upwards.

3 Adaptative localizer procedure

We assume we are given a FSM whose structure is unknown (so we cannot compute a homing sequence for it) but we are given T, a known separating splitting tree for it (see section 4 for a short discussion). The procedure L will bring the machine to an identified state, meaning that we know its output responses for all input sequences in a path of the splitting tree from the root to a leaf N. The procedure L should be called initially with d equal to the height of the splitting tree T.

The boolean function Predictable(**in** i, **in** Nt, **out** $N1$) returns true when we can be sure that the next (i-th) application of $L(d-1, T)$ would return a node $N1$. Nt is an array of nodes from the tree indexed from 0 to $i-1$.

```
1  procedure L(d, T) return Node N                    // d is max-depth to use
2
3      if d = 0 then
4          return root(T)
5      else if d = 1 then
6          apply I(root(T)), observe some Λ(N)
7          return N
8      else
9          i := 0
10         repeat
11             Nt[i] := L(d − 1, T)
12             if Nt[i] is a leaf then
13                 return Nt[i]        // a leaf will be returned through all
                                    // recursive calls as we are now localized
14             end
15             i := i + 1
16         until Predictable(i, Nt, N1)
17         N := L(d − 1, T(parent(N1) ← N1))
18         return N
19     end
20 end
```

We now illustrate the algorithm on the example FSM M_0 and its associated splitting tree T. In the following, we will denote a node N of the tree T by the path from the root of T to N. For instance, the node associated with $S3$ will be denoted by $\langle a/1, ab/11\rangle$. We use primes to differentiate inner values of variables in recursive calls. We suppose we start in the state $S0$. The depth of the splitting tree T is 3, so the first call to the adaptative localizer is $L(3, T)$.

$L(3, T)$
  $i := 0$
  $Nt[0] := L(2, T)$
    $i' := 0$
    $Nt'[0] := L(1, T)$
      apply a, observe 0 `// we are now in` $S1$
    $Nt'[0] = \langle a/0\rangle$
    $i' := i' + 1 = 1$
    $Nt'[1] := L(1, T)$
      apply a, observe 1 `// we are now in` $S2$
    $Nt'[1] = \langle a/1\rangle$
    ...

After five more iterations, we are in $S1$ and $Nt' = [\langle a/0\rangle, \langle a/1\rangle, \langle a/0\rangle, \langle a/1\rangle, \langle a/0\rangle, \langle a/1\rangle, \langle a/0\rangle]$
It is predictable that if we applied a, we would observe 1. Predictable(i, Nt', $N1$)
is true and $N1 = \langle a/1\rangle$

```
1  procedure Predictable(i, Nt, out N1) return boolean
2      r := 0                                              // repetition factor
3      Ns := empty_set          // set of states in last r elements of Nt
4      repeat
5          r := r + 1
6          Ns := Ns ∪ leaves(Nt[i − r])                    // set of all leaf nodes
7          s := card(Ns)
8      until r = i or s ≤ r
9      if s > r then
10         return false
11     else
                                // we have entered a loop, can we predict N1?
12         while leaves(Nt[i − r − 1]) ⊆ Ns and r ≤ i − 1 do
13             r := r + 1
14         end
15         find greatest j such that
16             i − r ≤ j < i − 1 and ∀m ∈ [0, r − s − 1], Nt[j − m] = Nt[i − 1 − m]
17         N1 := Nt[j + 1]
18         while j > i − s do
19             j := j − 1
20             if ∀m ∈ [0, r − s − 1], Nt[j − m] = Nt[i − 1 − m] and Nt[j + 1] ≠ N1
               then
21                 return false
22             end
23         end
24         return true
25     end
26 end
```

$N := L(1, T(root(T) \leftarrow \langle a/1 \rangle)$
apply ab, observe 10 `// we are now in` $S3$
return $\langle a/1, ab/10 \rangle$
$Nt[0] = \langle a/1, ab/10 \rangle$
$i := i + 1 = 1$
$Nt[1] := L(2, T)$
$i' := 0$
$Nt'[0] := L(1, T)$
apply a, observe 1 `// we are now in` $S4$
$Nt'[0] = \langle a/1 \rangle$
...

After 6 iterations, we are in $S4$ and $Nt' = [\langle a/1 \rangle, \langle a/0 \rangle, \langle a/1 \rangle, \langle a/0 \rangle, \langle a/1 \rangle, \langle a/0 \rangle, \langle a/1 \rangle]$
It is predictable that if we applied a, we would observe 0. Predictable(i, Nt', $N1$) is true and $N1 = \langle a/0 \rangle$

$N := L(1, T(root(T) \leftarrow \langle a/0 \rangle)$
apply b, observe 1 `// we are now in` $S5$

return $\langle a/0, b/1 \rangle$
// We have fully identified $S4$, and we are in $S5$
$Nt[1] = \langle a/0, b/1 \rangle$

The whole localizing sequence has $7 + 2 + 7 + 1 = 17$ inputs as follows (here decorated with landmark steps S', S''):

$$a/0\ a/1\ a/0\ a/1\ a/0\ a/1\ a/0\ S'\ ab/10\ a/1\ a/0\ a/1\ a/0\ a/1\ a/0\ a/1\ S''\ b/1$$

Let us recap: in the state S', we know that if we applied a, we would get 1. We thus apply the input sequence associated with node $\langle a/1 \rangle$, which is ab, and get 10. In the state S'', we know that if we applied a, we would get 0. We thus apply the input sequence associated with node $\langle a/0 \rangle$, which is b and get 1. We thus have fully identified the state S'' in the trace, which is $S4 = \langle a/0\, b/1 \rangle$.

If the procedure is started from $S1$, it localizes in 17 steps also. From $S2, S3, S4$ and $S5$ the length of the sequences yielded by the procedure are 9, 8, 26 and 25 respectively. The non-adaptive localizer with optimal ordering (by increasing size) of $W = \{b, ab, aab, aaab\}$ would yield $((b^{11}ab)^{11}b^{11}aab)^{11}(b^{11}ab)^{11}b^{11}aaab)$ requiring 1885 inputs. Even when the tree is not balanced the gain is huge, because only the worst case will require the full height of the tree.

4 Perspectives

An adaptive localizer as presented here lowers the complexity of localizing, from an exponential in the number of sequences to an exponential in the height of the splitting tree. This paves the way for new applications of localizing, in conformance testing or inference for non-resettable machines. In particular, it is possible to consider adapting an inference algorithm [1] so that the sequences separating subsets of states are discovered instead of being given. Thus it would be possible to infer incrementally with just a bound on the number of states, and with no initial knowledge of separating sequences for a black box system.

Acknowledgments The authors acknowledge feedback from A. Petrenko.

References

1. R. Groz, A. Simao, A. Petrenko, and C. Oriat. Inferring finite state machines without reset using state identification sequences. In *Proceedings of the International Conference on Testing Software and Systems, ICTSS 2015*, Dubai, nov 2015.
2. F.C. Hennie. Fault-detecting experiments for sequential circuits. In *Proc. Fifth Annual Symp. On Circuit Theory and Logical Design*, pages 95–110, 1965.
3. G-V. Jourdan, H. Ural, and H. Yenigun. Reducing locating sequences for testing from finite state machines. In *ACM-SAC'16*, pages 1654–1659, 2016.
4. Kohavi, Rivierre, and Kohavi. Checking experiments for sequential machines. *Information Science*, 7:11–28, 1974.
5. R. Smetsers, J. Moerman, and D. N. Jansen. Minimal separating sequences for all pairs of states. In A.-H. Dediu, J. Janoušek, C. Martín-Vide, and B. Truthe, editors, *Proceedings of the 13th International Conference on Languages and Automata Theory and Applications (LATA 2016)*, number 9618 in LNCS, 2016.