A state merging algorithm for real-time automata

Sicco Verwer S.E.VERWER@TUDELFT.NL
Mathijs de Weerdt M.M.DEWEERDT@TUDELFT.NL
Cees Witteveen C.WITTEVEEN@TUDELFT.NL
Delft University of Technology, P.O.Box 5031 2600GA Delft

We are interested in identifying a model for discrete event systems from observations. A common way to model discrete event systems is by using *deterministic finite state automata* (DFA). When observing a system, however, there often is information in addition to the system events, namely, their *times of occurrence*. If this time information is important, a DFA is too limited. For example, it is impossible to distinguish between events that occur quickly after each other, and events that occur after each other with a significant delay between them. Consequently, we would like a model that can also deal with time information.

A well-known model for this purpose is the *timed automaton* (TA) (Alur & Dill, 1994). In this model the time information is represented by a finite set of clocks, which can be reset by state transitions. The values of these clocks are then used in guards of transitions. A guard is a Boolean constraint on the values of the clocks. A state transition can only occur when the guard of the transition is satisfied. We use intervals to intervals to denote a constraint for a single clock. Such a constraint is satisfied when the clock value is an element of this interval.

We study the problem of identifying TAs from a data sample containing both positively and negatively labeled *timed strings*. For now, we focus on a simple type of automaton, known as the real-time automaton (RTA) (Dima, 2001). An RTA only allows time constraints on the time elapsed between two consecutive events. In other words, there is just one clock which is reset at every transition. We call the value of this clock the *current delay*. The structure of an RTA is different from a DFA only in the transitions it uses.

A transition $\langle q, q', s, \phi \rangle$ of an RTA is interpreted as follows: whenever the automaton is in state q, reading symbol s, and the delay guard ϕ is satisfied by the current delay, then the machine will move to state q'. Thus, in a DA it is not only possible to activate a transition to another state, but it is also allowed to remain in the same state for some time.

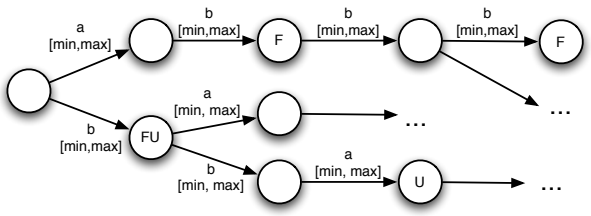

Figure 1. A normal APTA with delay guards (F stands for final, U for unfinal). The guards are initialized to the minimum and maximum values found in the input sample. The state reachable by a single b is both final and unfinal.

In previous work we have shown that learning an RTA is NP-complete by a reduction from the problem of learning a DFA (Verwer et al., 2006). Based on this reduction, we created a state merging algorithm (Lang et al., 1998) that can deal with time information. This algorithm works as follows:

- Assume there are no delay guards for all the transitions in the timed APTA. This results in a timed APTA such as the one in Figure 1. Note that such a timed APTA can contain conflicts.
- Continuously merge states in the standard fashion, but allowing conflicts.
- Sometimes (determined by a heuristic) split delay guards: For the delay guard $[t_1, t_2]$ of a transition m from state q, choose a time value s such that $t_1 < s < t_2$. Remove the part of the APTA from m on onward. Create two new transitions from state s: m_1 with guard $[t_1, s)$ and m_2 with guard $[s, t_2]$. Reconstruct the part of the timed APTA from m_1 and m_2 on onward. This split operation removes conflicts (but also consistent merges) from the DA. Figure 2 shows the result of a split operation.
- Then the determinization function continuously merges the transitions and target states of

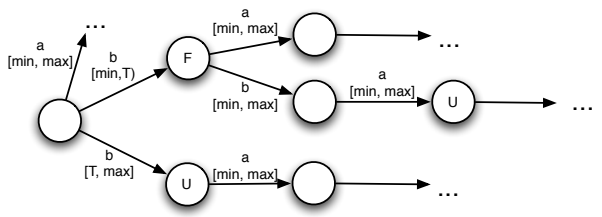

Figure 2. The result of a split of a part of the APTA of Figure 1. The state reachable by a single b is split using a delay value T.

overlapping transitions, until there is no nondeterministic choice left. The result of a merge of two overlapping intervals i and i' is an interval from the lowest lower-bound to the highest upper-bound of i and i'.

In this approach we can use a measure like the one used in the evidence driven state merging algorithm. The only difference is that now it has to deal with conflicting merges in addition to consistent merges. The measure we used in our algorithm is simply the number of added consistent merges minus the number of added conflicting merges. We created a red-blue fringe based algorithm that uses this measure. Conflicts in red nodes are disallowed. Algorithm 1 shows the pseudo code of this algorithm.

Algorithm 1 State merging delay automata

Require: A timed input sample S.
Ensure: A is a small DA that is consistent with the input sample S.
  Construct the timed APTA A from S.
  Color the root node red and all of its children blue.
  while Some nodes are colored blue **do**
    Evaluate all possible consistent merges between red and blue nodes.
    Evaluate all possible splits of the blue nodes.
    if All merges and splits score less or equal to 0 **then**
      Color the *worst scoring* blue node red.
    else
      Perform the *highest scoring* split or merge.
    end if
  end while
  return A

This algorithm can be used to learn an RTA from positive and negative data. The RTA learning algorithm is an alternative to the straightforward approach of first mapping the timed input sample to an untimed input sample, and then to learn the DFA from the untimed data. We sampled the data in the following manner:

- Obtain for each event e from a timed sample S, the value of the shortest delay d.
- For each occurrence of e, take its delay d_e, and determine the amount of samples $t = round(\frac{d_e}{d})$.
- Replace the timed occurrence of event e with t untimed occurrences of e.
- Repeat the procedure for $d = \frac{1}{2}d$, and $d = \frac{1}{4}d$.

We compared the performance of this approach with our RTA identification algorithm. Initial experimental results show that the RTA identification (95-99% correct classification) algorithm significantly outperforms the sampling method (70-80% correct classification).

Currently, our algorithm is only capable of correctly inferring small randomly generated RTAs. We are improving this result by trying to find better heuristics and data structures. As far as we know, no other learning algorithm exists to identify a TA from a timed sample.

A different approach to the identification of TAs is to identify it from a teacher. An algorithm for this purpose is given in (Grinchtein et al., 2006). Their approach, like ours, is to modify known algorithms for the inference of DFAs in order to deal with timed data. The main difference is that their approach makes use of a teacher. In our application area such a teacher is unavailable.

References

Alur, R., & Dill, D. L. (1994). A theory of timed automata. *Theoretical Computer Science*, *126*, 183–235.

Dima, C. (2001). Real-time automata. *Journal of Automata, Languages and Combinatorics*, *6*, 2–23.

Grinchtein, O., Jonsson, B., & Petterson, P. (2006). Inference of event-recording automata using timed decision trees. *CONCUR* (pp. 435–449). Springer.

Lang, K. J., Pearlmutter, B. A., & Price, R. A. (1998). Results of the abbadingo one dfa learning competition and a new evidence-driven state merging algorithm. *ICGI*. Springer.

Verwer, S., de Weerdt, M., & Witteveen, C. (2006). Identifying an automaton model for timed data. *Proceedings of the BENELEARN* (pp. 57–64).