IDENTIFYING AN AUTOMATON MODEL FOR TIMED DATA

Sicco Verwer Mathijs de Weerdt Cees Witteveen
Delft University of Technology, P.O.Box 5031 2600GA Delft

The full version of this paper appeared in: Proceedings of the Fifteenth Annual Machine Learning Conference of Belgium and the Netherlands (Benelearn 2006).

1 Introduction

In our paper we focus on learning systems of which the execution is determined by a finite set of discrete events. Such a system is known as a discrete event system (DES) [2]. A common way to model discrete event systems is by using deterministic finite state automata (DFA). The problem we study can be stated as follows: *Given a data sample of observations, how to identify the correct model for a specific DES.*

When observing a real-world system, however, there often is information in addition to the system events, namely, their times of occurrence. In other words, the data consists of *timed strings* of events, instead of just strings of events. 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.

Therefore, we propose a simple type of timed automaton [1] to model DES where the timing between two consecutive events is important. We call this type of automaton a delay automaton (DA). The structure of a DA is different from a DFA only in the transitions it uses:

Definition 1.1 *A transition $t \in T$ of a DA is a tuple $\langle q, q', s, \phi \rangle$, where q, q are the source and target states, s is a symbol, and ϕ is a delay guard defined by an interval in $\mathbb{R}^+$. A delay guard ϕ is* satisfied *by a time delay $d \in \mathbb{R}^+$ if $d \in \phi$.*

A transition $\langle q, q', s, \phi \rangle$ of a DA is interpreted as follows: whenever the automaton is in state q, reading 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 (delay).

The DA in Figure 1 models a 'smart' automatic bike light that does not turn the light off when the bike stops for a traffic light. The execution of this DA starts in the state off. A stop at a traffic light should not last longer than one minute. This is modeled by a light off event only occurring at time 60. A start event before 60 seconds will make the automaton return to the on state.

In our paper we prove that Learning a DA is NP-complete by a reduction from the problem of learning a DFA. Based on this reduction, we show how the currently best learning algorithm for DFAs (state merging [3]) can be adapted to deal with time information. The algorithm works as follows:

- Construct a timed augmented prefix tree acceptor (TAPTA) from an input sample. An APTA is a DA, shaped like a tree, that accepts the positive strings from the input sample, and rejects the negative strings.

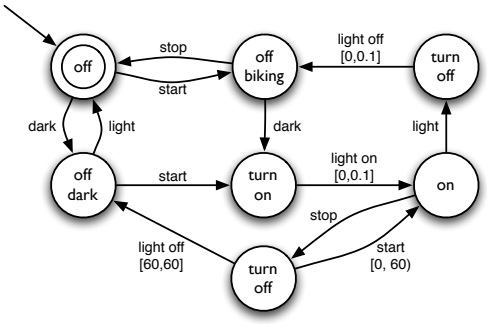

Figure 1: This DA models a 'smart' bike light that leaves the light on when the bike stops for less than 60 seconds.

- Each transition in the TAPTA is given as its delay guard the degenerate interval (of length zero) of exactly the delay value of the timed sample string used to create the transition.
- Continuously merge the states of the TAPTA in the normal way. But, also merge the intervals into larger intervals: If two states are merged and this causes two transitions with the same label to point to the same state, then merge the intervals into one. The result of a merge of two intervals i and i' is an interval from the lowest lower-bound to the highest upper-bound of i and i'.

This algorithm can be used to learn a DA from positive and negative data. As far as we know, currently no other learning algorithm exists to identify a timed automaton from a timed sample.

References

[1] Rajeev Alur. Timed automata. In *International Conference on Computer-Aided Verification*, pages 8–22. Springer-Verlag, 1999.

[2] Christoc G. Cassandras and Stephane Lafortune. *Introduction to Discrete Event Systems*, volume 11 of *The Kluwer International Series on Discrete Event Dynamic Systems*. Springer Verlag, 1999.

[3] Kevin J. Lang, Barak A. Pearlmutter, and Rodney A. Price. Results of the abbadingo one DFA learning competition and a new evidence-driven state merging algorithm. In *ICGI*, volume 1433 of *Lecture Notes in Computer Science*, pages 1–12, 1998.