A Lazy Reachability Algorithm for Timed Automata

B. Srivathsan

LaBRI, Université Bordeaux 1

The reachability problem for timed automata refers to deciding if there exists a path from its initial state to a given target state. We propose a new algorithm for this problem. The first solution to the reachability problem was proposed in the paper introducing timed automata [1]. It was based on *regions*: equivalence classes of clock valuations. Their definition is parametrized by a threshold up to which the clock values should be considered. Subsequent research has shown that the region abstraction is very inefficient. Another method using *zones* instead of regions has been proposed which can be implemented efficiently using DBMs [6]. It is used at present in all timed-verification tools [3,5].

While simple at first sight, the zone abstraction was delicate to get right. The number of reachable zones can be infinite, so one needs an abstraction operator to get a finite approximation. The simplest is to approximate a zone with a set of regions it intersects: so called *Closure* of a zone. Unfortunately closure may not always be convex and no efficient representation of closures is known. For this reason implementations use another approximation *Approx* that is also based on (refined) regions [4].

Another important step in efficient implementation is calculation of thresholds for approximations. Definition of an approximation depends on the regions that in turn depend on the choice of a threshold. The safe choice is to take the maximal constant appearing in the transitions of the automaton. A considerable gain in efficiency can be obtained by analyzing the graph of the automaton and calculating thresholds specific for each clock and state of the automaton [2].

We present a different approach to the reachability testing. Our algorithm works directly on (unapproximated) zones: the reachability tree it constructs is labeled with zones. In order to ensure termination we use the simple zone closure operator. Since we have no efficient way to represent closures, we compute them each time an inclusion test is performed. We show that we can do this at no additional cost: we provide an algorithm for checking $Z \subseteq Closure(Z')$ that has the same complexity as comparing $Z \subseteq Z'$. Thus we can use the efficient representation of zones and the *Closure* approximation without having to manipulate closures explicitly. Since *Closure* is a coarser approximation than *Approx*, its use gives an important performance gain. The second feature of our algorithm involves computation of thresholds for the closure approximation on-the-fly. We show how one could use the information gathered during reachability analysis to calculate better thresholds. This way of calculating thresholds adapts well to a, very common, case of analysis of parallel compositions of timed automata.

This is joint work with Frédéric Herbreteau, Igor Walukiewicz from LaBRI, and Dileep Kini from IIT Bombay.

References

1. R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
2. G. Behrmann, P. Bouyer, E. Fleury, and K. G. Larsen. Static guard analysis in timed automata verification. In *TACAS'03*, volume 2619 of *LNCS*, pages 254–270. Springer, 2003.
3. G. Behrmann, A. David, K. G Larsen, J. Haakansson, P. Pettersson, W. Yi, and M. Hendriks. Uppaal 4.0. In *QEST'06*, pages 125–126, 2006.
4. P. Bouyer. Forward analysis of updatable timed automata. *Form. Methods in Syst. Des.*, 24(3):281–320, 2004.
5. M. Bozga, C. Daws, O. Maler, A. Olivero, S. Tripakis, and S. Yovine. KRONOS: a mode-checking tool for real-time systems. In *CAV'98*, volume 1427 of *LNCS*, pages 546–550. Springer, 1998.
6. D. Dill. Timing assumptions and verification of finite-state concurrent systems. In *Automatic Verification Methods for Finite State Systems*, volume 407 of *Lecture Notes in Computer Science*, pages 197–212. Springer, 1989.