My Article

Sybe van Hijum

Today

1 Introduction

Timed Automata [1] are a widely used modelling formalism. A recent usage of this formalism is the modelling of biological signalling pathways [15]. This leads however to large state spaces, and sometimes to models that are too large to handle by conventional methods. Therefore the ANIMO [15] tool at this time uses simulation of models and not complete state space generation and property checking.
BDDs(Binary Decision Diagrams) [?] and variations like LDDs(List Decision Diagrams) [5] and MDDs(Multi-valued Decision Diagrams) [16] have proven their value in model checking algorithms. Due to advances in this field models with much larger state spaces can be explored on the same machine. This progress has not translated directly to more efficient methods for Timed Automata. Several methods have been proposed, like CDDs(Clock Difference Diagrams) [12], CMDs(Constraint Matrix Diagrams [10], CRDs(Clock Restriction Diagrams) [18] and DDDs(Difference Decision Diagrams) [13]. All of these methods show some extra difficulties or time complexity over BDDs or some limitations.
LTSmin [4, 11] is a language independent on the fly model checker with several algorithmic backends. Its symbolic backend uses LDDs to both represent the state space and the transition relations of models. LTSmin has a language module for the UPPAAL [2] through the opaal [7] lattice model checker. For this language at this time, only the multicore backend can be used [6]. This multicore approach showed efficient enough to compete with the latest version of the UPPAAL model checker. It showed significant speedups on multicore machines, at the cost of some memory increase however.
The symbolic backend of LTSmin provides both a memory reduction by using LDDs and a speedup by using multi-threaded search algorithms and

the multi-threaded LDD package Sylvan [17]. Using this together with the UPPAAL language frontend will hopefully result in a model checker that can compete both on time and memory consumption with the UPPAAL model checker. We propose a method that uses LDDs to represent both the discrete states as the clocks and that uses DBMs [3,8] only in the next state generation. This way we can combine existing techniques to build a complete model checker. It will also remain possible to use the other techniques that LTSmin has, such as transition caching, variable reordering and LTL checking. This will result in a complete model checker for Timed Automata.

2 Related Work

Already several model checkers for Timed Automata exist such as UPPAAL [2] and RED [18]. We focus mainly on the UPPAAL tool as we use the same input format. Opaal uses the XML format that is created by the UPPAAL tools. This way we can use the UPPAAL user interface to create and adapt models. We also use the UPPAAL DBM library to represent zones. Several methods exist to represent the clock variables in a timed model. The most used methods are digitization and zones. Digitization approximates the continuous values of clocks by using discrete values. This approach is however very sensitive to the granularity of the values used and the upper bound of the clock values. An advantage of this approach is that basic model checking approaches can be used and no extra complexity due to zone calculations is added. In [14] a similar approach is proposed by using clock tick actions and removing clock variables altogether.
The most established method to represent clock zones are DBMs [3, 8]. DBMs use a matrix structure that gives an lower and upper bound to each clock and to the difference between each pair of clocks. By this approach convex zones of clocks can be created. By using graph algorithms a normal form can be found quite efficiently. The downside of this approach is that only these convex zones can be represented, when a state has multiple zones that are not a convex combination multiple DBMs are needed and thus increasing the memory usage.
Several methods based on BDDs have been developed to represent zones. All of these are based on DMBs in the sense that the use upper and lower bounds of clocks and of the difference between a pair of clocks. CDDs [12] use single nodes for each value with a larger fanout with disjoint intervals on each edge. DDDs [13] use a constraint on each node that can either be true or false. This requires a fixed ordering based on the variables, values and op-

erators. CRDs [18] differ mainly from CDDs by using not disjoint intervals but possibly overlapping upperbounds on their edges. They also use different normal forms for the diagrams which results in different performances. It is also shown that CRDs can be combined with BDDs into a single structure to represent state space. CMDs [10] combine CDDs, CRDs and DMBs into a single structure. This diagram type differs from the others by having multiple constraint per edge, resulting in a diagram with few nodes. CMDs do not have a normal form so only reduced forms are proposed. In [9, 19] a method is proposed purely based on BDDs by translating the constraints into BDD nodes. This results in a unified structure for both the discrete variables and the clock constraints. The method has not been implemented in a model checker and no performance results are known, only the concept is known.

2.1 Timed Automata

Timed Automata is a formalism that extends labelled transition systems with one ore more, but finite, clocks. As our work continues on [6] we use the same definition of timed automata.

Definition 1 (Timed Automata)**.** *An extended timed automaton is a 7-tuple* $A = < L, C, Act, s_0, \rightarrow, I_c >$ *where*

- L *is a finite set of locations, typically denoted by* l
- C *is ia finite set of clocks, typically denoted by* c
- Act *is a finite set of actions*
- $s_0 \in L$ *is the initial location*
- $\rightarrow \subseteq L \times G(C) \times Act \times 2^C \times L$ *is the (non-deterministic) transition relation. We noramlly write* $l \xrightarrow{g,a,r} l'$ *for a transition., where* l *is the source location,* g *is the guard over the clocks,* a *is the action, and* r *is the set of clocks reset.*
- $I_C : L \rightarrow G(C)$ *is a function mapping locations to downwards closed clock invariants.*

With this definition we can combine timed automata to a network of timed automata.

Definition 2 (Network of timed automata [6]). *Let $Act = \{ch!, ch?|ch \in Chan\} \cup \{\tau\}$ be a finite set of actions, and let C be a finite set of clocks. Then the parallel composition of extended timed automata $A_i = (L_i, C, Act, S_0^i, \rightarrow_i, I_C^i)$ for all $1 \leq i \leq n$, where $n \in \mathbb{N}$, is a network of timed automata, denoted $A = A_1||A_2||..||A_n$.*

2.2 Zones

For basic transition systems the state space can grow exponentially for the size of the system. For Timed Automata the growth can become even larger and in some cases become unbounded. To tackle this problem most model checkers use a notion of zones for the representation of time. A zone can be seen as a set of constraints over the clocks C of the form $c_i \sim x$ and $c_i - c_j \sim x$ where $\sim \in \{<, \leq, =, \geq, >\}$ and $x \in \mathbb{N}$. To represent these zones several data structures have been developed. One of the most common used structures are Difference Bound Matrices(DMB's) [3].
These matrices use both a column and a row for each clock, and on each position (i, j) an upperbound on the difference between the clocks c_i and c_j is given in the form $c_i - c_j \preceq x$ where $\preceq \in \{<, \leq\}$ and $x \in \mathbb{N}$. For the constraints over the single clocks an extra clock **O** with a constant value 0 is added.

References

[1] Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183 – 235, 1994.

[2] Gerd Behrmann, Alexandre David, and KimG. Larsen. A tutorial on uppaal. In Marco Bernardo and Flavio Corradini, editors, *Formal Methods for the Design of Real-Time Systems*, volume 3185 of *Lecture Notes in Computer Science*, pages 200–236. Springer Berlin Heidelberg, 2004.

[3] Johan Bengtsson. *Clocks, DBMS and States in Timed Systems (Uppsala Dissertations from the Faculty of Science Technology, 39).* Uppsala Universitet, 7 2002.

[4] S. C. C. Blom, J. C. van de Pol, and M. Weber. Ltsmin: Distributed and symbolic reachability. In T. Touili, B. Cook, and P. Jackson, editors, *Computer Aided Verification, Edinburgh*, volume 6174 of *Lecture Notes in Computer Science*, pages 354–359, Berlin, July 2010. Springer Verlag.

[5] Stefan Blom and Jaco van de Pol. Symbolic reachability for process algebras with recursive data types. In J.S. Fitzgerald, A.E. Haxthausen, and H. Yenigun, editors, *Theoretical Aspects of Computing*, volume 5160 of *Lecture Notes in Computer Science*, pages 81–95, Berlin, Germany, August 2008. Springer Verlag.

[6] A. E. Dalsgaard, A. W. Laarman, K. G. Larsen, M. C. Olesen, and J. C. van de Pol. Multi-core reachability for timed automata. In M. Jurdzinski and D. Nickovic, editors, *10th International Conference on Formal Modeling and Analysis of Timed Systems, FORMATS 2012, London, UK*, volume 7595 of *Lecture Notes in Computer Science*, pages 91–106, London, September 2012. Springer Verlag.

[7] Andreas Engelbredt Dalsgaard, Ren Rydhof Hansen, Kenneth Yrke Jrgensen, Kim Gulstrand Larsen, Mads Chr. Olesen, Petur Olsen, and Ji Srba. opaal: A lattice model checker. In Mihaela Bobaru, Klaus Havelund, GerardJ. Holzmann, and Rajeev Joshi, editors, *NASA Formal Methods*, volume 6617 of *Lecture Notes in Computer Science*, pages 487–493. Springer Berlin Heidelberg, 2011.

[8] DavidL. Dill. Timing assumptions and verification of finite-state concurrent systems. In Joseph Sifakis, editor, *Automatic Verification Methods for Finite State Systems*, volume 407 of *Lecture Notes in Computer Science*, pages 197–212. Springer Berlin Heidelberg, 1990.

[9] Junwei Du, Huiping Zhang, Gang Yu, and Xi Wang. A full symbolic compositional reachability analysis of timed automata based on bdd. In *Advanced Computational Intelligence (ICACI), 2015 Seventh International Conference on*, pages 218–222, March 2015.

[10] R. Ehlers, D. Fass, M. Gerke, and H.-J. Peter. Fully symbolic timed model checking using constraint matrix diagrams. In *Real-Time Systems Symposium (RTSS), 2010 IEEE 31st*, pages 360–371, Nov 2010.

[11] A. W. Laarman, J. C. van de Pol, and M. Weber. Multi-core ltsmin: Marrying modularity and scalability. In M. Bobaru, K. Havelund, G. Holzmann, and R. Joshi, editors, *Proceedings of the Third International Symposium on NASA Formal Methods, NFM 2011, Pasadena, CA, USA*, volume 6617 of *Lecture Notes in Computer Science*, pages 506–511, Berlin, July 2011. Springer Verlag.

[12] Kim Larsen, Carsten Weise, Wang Yi, and Justin Pearson. Clock difference diagrams. *BRICS Report Series*, 5(46), 1998.

[13] Jesper Mller, Jakob Lichtenberg, HenrikReif Andersen, and Henrik Hulgaard. Difference decision diagrams. In Jrg Flum and Mario Rodriguez-Artalejo, editors, *Computer Science Logic*, volume 1683 of *Lecture Notes in Computer Science*, pages 111–125. Springer Berlin Heidelberg, 1999.

[14] TruongKhanh Nguyen, Jun Sun, Yang Liu, JinSong Dong, and Yan Liu. Improved bdd-based discrete analysis of timed systems. In Dimitra Giannakopoulou and Dominique Mry, editors, *FM 2012: Formal Methods*, volume 7436 of *Lecture Notes in Computer Science*, pages 326–340. Springer Berlin Heidelberg, 2012.

[15] Stefano Schivo, Jetse Scholma, Brend Wanders, Ricardo A. Urquidi Camacho, Paul E. van der Vet, Marcel Karperien, Rom Langerak, Jaco van de Pol, and Janine N. Post. Modelling biological pathway dynamics with timed automata. In *12th IEEE International Conference on Bioinformatics & Bioengineering, BIBE 2012, Larnaca, Cyprus, November 11-13, 2012*, pages 447–453, 2012.

[16] A. Srinivasan, T. Ham, S. Malik, and R.K. Brayton. Algorithms for discrete function manipulation. In *Computer-Aided Design, 1990. ICCAD-90. Digest of Technical Papers., 1990 IEEE International Conference on*, pages 92–95, Nov 1990.

[17] Tom van Dijk and Jaco van de Pol. Sylvan: Multi-core decision diagrams. In Christel Baier and Cesare Tinelli, editors, *Tools and Algorithms for the Construction and Analysis of Systems*, volume 9035 of *Lecture Notes in Computer Science*, pages 677–691. Springer Berlin Heidelberg, 2015.

[18] Farn Wang. Efficient verification of timed automata with bdd-like datastructures. In LenoreD. Zuck, PaulC. Attie, Agostino Cortesi, and Supratik Mukhopadhyay, editors, *Verification, Model Checking, and Abstract Interpretation*, volume 2575 of *Lecture Notes in Computer Science*, pages 189–205. Springer Berlin Heidelberg, 2003.

[19] Huiping Zhang, Junwei Du, Ling Cao, and Guixin Zhu. A full symbolic reachability analysis algorithm of timed automata based on bdd. In *Autonomous Decentralized Systems (ISADS), 2015 IEEE Twelfth International Symposium on*, pages 301–304, March 2015.