Rabbit: A Tool for BDD-Based Verification of Real-Time Systems

Dirk Beyer, Claus Lewerentz, and Andreas Noack

Software Systems Engineering Research Group
Brandenburg Technical University at Cottbus, Germany
`{db|cl|an}@informatik.tu-cottbus.de`

Abstract. This paper gives a short overview of a model checking tool for real-time systems. The modeling language are timed automata extended with concepts for modular modeling. The tool provides reachability analysis and refinement checking, both implemented using the data structure BDD. Good variable orderings for the BDDs are computed from the modular structure of the model and an estimate of the BDD size. This leads to a significant performance improvement compared to the tool RED and the BDD-based version of Kronos.

1 Introduction

Timed automata are a common and theoretically well-founded formalism for real-time systems [1]. Reachability analysis of timed automata has been implemented in several tools, the best-known being Kronos [11] and Uppaal [2]. From our point of view, two major problems are the lack of concepts for modeling large systems and the exploding consumption of time and memory by the verification algorithms. We address these issues with our tool *Rabbit*, which provides the following features:

– **Modular Modeling.** Our modular extension of timed automata is called Cottbus Timed Automata (CTA). The automata are encapsulated by modules. Each module has an explicit *interface*, which declares the variables and synchronization labels used for the communication with other modules. The use of these variables and synchronization labels can be restricted by specifying an explicit access mode (read only, exclusive write, etc.). Replicated subsystems do not have to be multiply defined, but can be instantiated from a common *template module*. Modules can contain other modules to form hierarchical structures. Our formalism provides a *compositional semantics*, i.e. we can define the semantics of a CTA module on the basis of the semantics of its components [7].

– **Reachability Analysis.** The tool provides efficient reachability analysis for timed automata. Sets of configurations are represented by the data structure binary decision diagram (BDD). The modular structure of the model is used to compute BDD variable orderings for an efficient representation of the transition relation and the set of reachable configurations.

– **Refinement Checking.** To make the verification of large systems tractable, detailed modules can be replaced by more abstract modules. To prove the correctness of

such a replacement regarding safety properties, we have to check that two modules have the same behavior with respect to external synchronization labels, i.e. the set of traces of the abstract module has to be a superset of the set of traces of the detailed module. To enable efficient modular proofs, we check the existence of a *simulation relation* in our tool implementation. A simulation relation exists in many practical cases of trace inclusion, especially when stepwise refinement is used in the development process. Details about the implementation of this refinement check are given in [4].

For a comprehensive and detailed explanation of all the concepts and the tool, we refer to [5]. The tool Rabbit, example models and related papers are available from `http://www.software-systemtechnik.de/Rabbit`.

2   BDD-Based Reachability Analysis

Safety properties of timed automata can be verified by reachability analysis. The main problem is the exploding consumption of time for the computation and memory for the representation of the reachable configurations. Therefore the data structure for sets of configurations is of vital importance. Sets of configurations of timed automata consist of locations and associated sets of clock assignments. For the symbolic representation of sets of locations binary decision diagrams (BDDs) are widely used. For a uniform representation of locations and clock assignments as BDDs we defined an **integer semantics** which only considers integer clock assignments. We proved that for timed automata without strict clock constraints (i.e. without $<$ and $>$ in clock constraints), this integer semantics is equivalent to the usual, continuous semantics with respect to the reachable locations [3]. The restriction to non-strict clock constraints is of technical nature, and we did not find examples within our application area of production cell controllers and real-time algorithms for which it is difficult to construct models without strict constraints.

Representing the transition relation as several BDDs and applying these **partial transition relations** sequentially is more efficient than using a monolithic transition relation [9]. Concerning the **order of their application**, always computing the fixed point of the discrete transitions before applying time transitions is a successful strategy to avoid large intermediate BDDs.

We use a heuristic to find good BDD **variable orderings**. This heuristic first computes an initial variable ordering and than improves it by local search. The initial variable ordering is a pre-order linearization of the module hierarchy (i.e. the modules are in the order in which they are reached by a depth-first traversal of the hierarchy). This assigns local components of a module to neighboring positions in the variable ordering, and thus exploits the knowledge of the modeler who usually tries to create cohesive modules. Then local search is applied to improve the ordering with respect to a size estimate for the BDD of the reachability set [3]. This estimate reflects the two most important characteristics for good variable orderings: (1) Communicating components have neighboring positions within the ordering. (2) Components which communicate with many other components precede these other components.

Table 1. Time for the computation of the reachability set of Fischer's protocol

# proc.	4	5	6	7	8	10	12	14	16	32	64	128
Uppaal	0.06	1.44	181	32488								
RED	1.64	6.78	21.7	60.7	168	1400						
Rabbit	0.04	0.08	0.15	0.26	0.50	1.35	1.61	3.81	6.50	61.4	559	5200

Table 2. Time for the computation of the reachability set of the FDDI protocol

# senders	2	4	6	8	10	12	14	16
Uppaal	0.01	0.03	0.16	1.42	18.2	279	4530	
RED	0.02	0.09	0.26	0.61	1.18	2.16	3.62	6.31
Rabbit	0.04	0.25	0.99	4.20	11.4	26.9	49.8	142

3   Performance Results

Performance results are given in seconds of CPU time on a Linux PC with an AMD Athlon processor (1 GHz) for the publicly available tools RED [10] version 3.1, which is based on a BDD-like data structure called CRD, Rabbit [4] version 2.1, which is based on BDDs with automatic variable ordering, and Uppaal2k [2] version 3.2.4 (without approximation), a popular and highly optimized DBM-based tool. An empty table entry means that the analysis needs more than 400 MB memory or more than 7 200 s computation time.

Detailed analytical explanations of the experimental results can be found in [6]. This paper also discusses the sensitivity of the BDD representation to the size of constants in the model, which is a major disadvantage of BDDs compared to CRDs and DBMs.

Fischer's Protocol (Table 1). Fischer's timing-based protocol for mutual exclusion is a protocol for accessing a shared resource. We computed the set of reachable configurations to verify the mutual exclusion property. For the BDD-based version of Kronos (which is not publicly available) a maximum of 14 processes is reported in [8].

The example of Fischer's protocol illustrates the dramatic influence of the variable ordering: The set of all reachable configurations for 16 processes is represented by 2 096 957 BDD nodes using an ordering with the shared variable k at the last position. This ordering violates the second characteristic for good orderings because variable k is used by all processes. A good variable ordering with variable k at the first position reduces the representation to 6 190 BDD nodes.

Token-Ring-FDDI Protocol (Table 2). Fiber Distributed Data Interface (FDDI) is a high speed protocol for local networks based on token ring technology. The automata model was introduced by Yovine [12]. Here RED outperforms Rabbit because the number of reachable locations does not explode with growing number of senders.

CSMA/CD Protocol (Table 3). CSMA/CD (Carrier Sense, Multiple Access with Collision Detection) is a protocol for communication on a broadcast network with a multiple access medium. The timed automata model is taken from [11].

Production Cell. To validate the suitability of our tool for more realistic models, we developed a CTA model of a production cell. This system consists of 20 machines and belts with 44 sensors and 28 motors. We modeled the system as a modular composi-

Table 3. Time for the computation of the reachability set of the CSMA/CD protocol

# senders	2	4	6	8	10	12	14	16	32	64	128	256
Uppaal	0.01	0.03	5.1									
RED	0.05	0.28	1.15	5.88	41.4	516						
Rabbit	0.02	0.08	0.23	0.49	0.82	1.28	1.83	2.69	12.6	62.9	367	2160

tion of several belts, turntables and machines, using 82 timed automata with 44 clocks, 17 discrete variables and 183 synchronization labels. For a model with a simple communication structure like Fischer's protocol good variable orderings might be obvious (at least for experts). The production cell shows the need for automatic variable ordering, and the effectiveness of our automatic estimate-based heuristic: The pre-order linearization of the module hierarchy results in a BDD for the reachability set with 378 229 nodes. This is a good ordering, much better than the vast majority of other permutations, but applying our heuristic improves the representation to 14 895 nodes. Modular proofs using refinement checking simplify the models used in reachability analysis and thus further reduce the BDD size.

References

1. Rajeev Alur and David L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.
2. Tobias Amnell, Gerd Behrmann, Johan Bengtsson, Pedro R. D'Argenio, Alexandre David, Ansgar Fehnker, Thomas Hune, Bertrand Jeannet, Kim G. Larsen, M. Oliver Möller, Paul Pettersson, Carsten Weise, and Wang Yi. Uppaal - now, next, and future. In *Proc. MOVEP'00*, LNCS 2067, pages 99–124. Springer, 2001.
3. Dirk Beyer. Improvements in BDD-based Reachability Analysis of Timed Automata. In *Proc. FME'01*, LNCS 2021, pages 318–343. Springer, 2001.
4. Dirk Beyer. Efficient Reachability Analysis and Refinement Checking of Timed Automata using BDDs. In *Proc. CHARME'01*, LNCS 2144, pages 86–91. Springer, 2001.
5. Dirk Beyer. *Formale Verifikation von Realzeit-Systemen mittels Cottbus Timed Automata*. Verlag Mensch & Buch, Berlin, 2002. Zugl.: Dissertation, BTU Cottbus, 2002.
6. Dirk Beyer and Andreas Noack. A comparative study of decision diagrams for real-time verification. Technical Report I-03/2003, BTU Cottbus, 2003.
7. Dirk Beyer and Heinrich Rust. Cottbus Timed Automata: Formal Definition and Semantics. In *Proc. FSCBS'01*, pages 75–87, 2001.
8. Marius Bozga, Oded Maler, Amir Pnueli, and Sergio Yovine. Some progress on the symbolic verification of timed automata. In *Proc. CAV'97*, LNCS 1254, pages 179–190, 1997.
9. Jerry R. Burch, Edmund M. Clarke, David E. Long, Kenneth L. McMillan, and David L. Dill. Symbolic model checking for sequential circuit verification. *IEEE Transactions on CAD*, 13(4):401–424, April 1994.
10. Farn Wang. Symbolic verification of complex real-time systems with clock-restriction diagram. In *Proc. FORTE'01*, pages 235–250. Kluwer, 2001.
11. Sergio Yovine. Kronos: A verification tool for real-time systems. *Software Tools for Technology Transfer*, 1(1-2):123–133, October 1997.
12. Sergio Yovine. Model checking timed automata. In *Lectures on Embedded Systems*, LNCS 1494, pages 114–152. Springer, 1998.