

✓erics: A Tool for Verifying Timed Automata and Estelle Specifications*

Piotr Dembiński¹, Agata Janowska², Paweł Janowski², Wojciech Penczek¹,
Agata Pólrola³, Maciej Szreter¹, Bożena Woźna⁴, Andrzej Zbrzezny⁴

¹ Institute of Computer Science, PAS, Ordonia 21, 01-237 Warsaw, Poland
{piotrd,penczek,mszreter}@ipipan.waw.pl

² Institute of Informatics, Warsaw University, Banacha 2, 02-097 Warsaw, Poland
{janowska,janowski}@mimuw.edu.pl

³ Faculty of Mathematics, University of Lodz, Banacha 22, 90-238 Lodz, Poland
polrola@math.uni.lodz.pl

⁴ Institute of Mathematics and Computer Science, Pedagogical University of
Częstochowa, Armii Krajowej 13/15, 42-200 Częstochowa, Poland
{b.wozna,a.zbrzezny}@wsp.czyst.pl

Abstract. The paper presents a new tool for automated verification of Timed Automata as well as protocols written in the specification language Estelle. The current version offers an automatic translation from Estelle specifications to timed automata, and two complementary methods of reachability analysis, the first of which is based on Bounded Model Checking (BMC), while the second one is an on-the-fly verification on an abstract model of the system.

1 Introduction

We present a new tool for automated verification of Timed Automata as well as protocols written in the specification language Estelle. The main novelty of our tool consists in combining the translation from a subset of Estelle to Timed Automata [7] with the translation of the reachability problem for Timed Automata to the satisfiability problem of propositional formulas (SAT-problem) [20]. The latter problem is very efficiently solved by the SAT-solver ZChaff [14] that is exploited in our tool. Since the above approach is mainly applicable for finding errors, i.e., disproving safety properties, we extend our tool ✓erics with another module, which is used when the correctness is to be proved. This module is based on our original method [17] of building a pseudo-bisimulating model for a timed automaton, which preserves the reachability properties. Reachability over a pseudo-bisimulating model is checked on-the-fly, i.e., while building the model. The architecture of ✓erics is composed of the following modules (see also Fig. 1):

- **Language Translator** from the fragment of Estelle to the intermediate language,

* Partly supported by the State Committee for Scientific Research under the grant No. 8T11C 01419 .

- **TA Translator** from the intermediate language to timed automata,
- **BBMC Reachability Analyser** that verifies reachability properties over timed automata,
- **Splitter** that generates pseudo-bisimulating model for timed automata,
- **Verifier** that verifies reachability properties over pseudo-bisimulating models.

2 Related Tools

The high-level modelling languages, among which Estelle [11], SDL [12], LOTOS [10] and Promela [9] belong to most widely used, were created to describe logical circuits, distributed systems and communication protocols. Many tools for design in these languages were produced, but they usually lacked any support of formal verification. On the other hand, for a long time the model-checking tools were designed for testing new scientific ideas, without paying much attention to their applicability for verification of real-world complex concurrent systems.

The first kind of model-checking tools were explicit state-space checkers, suffering the *state explosion problem*. Model checkers based on an explicit state space representation are still developed (SPIN [9], Kronos [21], UppAal [3] etc.), exploiting various methods of overcoming the above drawback. Another methodology is *symbolic model checking*, in which an explicit representation of states is replaced by a symbolic one (usually of a form of a decision diagram). The above approach is exploited in many packages, like Rabbit [4] and SMV [13]. The next branch of tools, represented by NuSMV [6] and MATHSAT [18], is based on Bounded Model Checking. Verification problems are here transformed to checking satisfiability of boolean formulas, solved by a SAT-prover.

Several solutions for connecting the above languages and tools were proposed. IF [5] uses an intermediate language IF, to which higher-level specifications can be translated, enabling further verification using one of the above tools. Another example of a development environment is CADP [8], which allows to transform a LOTOS system description to the formats accepted by model checkers.

3 Theory Behind $\sqrt{\text{erics}}$

The theoretical background for our implementation has been presented in several papers [7, 15, 20]. In this section we sketch the main ideas.

Our tool accepts three kinds of input: specifications written in a subset of Estelle or in the intermediate language, or timed automata. **Estelle** [11] is an ISO standard specification language designed for describing communication protocols and distributed systems. The **intermediate language (IL)** [7] allows for describing a system as a set of processes, which exchange information by message passing (via bounded or unbounded channels) or memory sharing (using global variables). A process is described in terms of states and transitions similarly like in Estelle.

The translation from the subset of Estelle to the intermediate language is quite straightforward, as the execution models and the syntax of these formalisms are

similar, although some Estelle language constructions require special and careful treatment. The details about the translation can be found in [7].

A system described in the intermediate language can be further translated either to a set of **timed automata** [1], each of which represents a component of the system, or to a global (product) **timed automaton** (for the description see [7]). The automata that are obtained are then passed to other components of $\sqrt{\text{erics}}$, which are aimed at performing reachability model checking.

Given a property p , reachability model checking consists in an exploration of the state space of the system, testing whether there exists a reachable state where p holds. Our tool offers two complementary methods of reachability analysis, the first of which is based on Bounded Model Checking (BMC) and a SAT-solver, while the second one is an on-the-fly verification on an abstract model of the system. The BMC-based method combines the well-know *forward reachability* analysis and the bounded model checking method for Timed Automata [15, 16, 19, 20]. The forward reachability algorithm searches the state space by moving from a state to its successors in the breadth-first mode, whereas BMC performs a verification on a part of the model exploiting a SAT-solver. The detailed description of the above method can be found in [19, 20]. In case when any state satisfying a tested property is not reachable, the SAT-based method can be ineffective. Therefore, in parallel to the BMC-based reachability analysis, $\sqrt{\text{erics}}$ offers a verification method consisting in generating finite *abstract models* for Timed Automata (the pseudo-bisimulating ones [17]), using a partitioning algorithm, and performing an on-the-fly reachability verification. The detailed description of the above algorithm can be found in [17].

4 Tool Overview

As it has been already stated, $\sqrt{\text{erics}}$ allows for an input in Estelle. Moreover, the system to be verified can be given in the intermediate language or in the form of timed automata in the Kronos-like format. Below, we present a short description of the case, where an Estelle specification is given as an input.

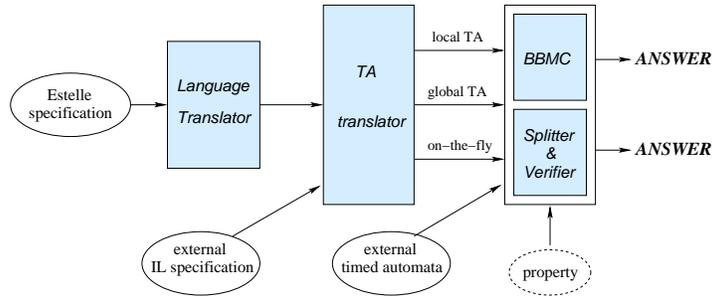


Fig. 1. The modules of $\sqrt{\text{erics}}$

An Estelle specification is automatically translated to a description in the intermediate language (by the Language Translator). The obtained specification usually requires additional (manual) modifications, aimed at adding properties to states or bounds on sizes of buffers (all these features are not handled in the Estelle standard). Then, the enriched specification is passed to the TA Translator, which generates either a set of timed automata corresponding to the components of the system, or a global timed automaton. The automata are returned in the Kronos-like format. Then, they are passed to another component of $\sqrt{\text{erics}}$ aimed at reachability model checking, i.e., BBMC or Splitter. The connection between the modules of $\sqrt{\text{erics}}$ is presented in Fig. 1.

5 Case Studies

We provide experimental results for three well-known examples: the Alternating Bit Protocol, Fisher’s Mutual Exclusion Protocol and Railroad Crossing System (RCS). As the input for the first example we have used an Estelle specification. Two properties have been tested: “whenever the sender receives an acknowledgement and its bit is set to 0, then the receiver’s bit is set to 0 as well” (false), and “whenever the sender receives an acknowledgement, its bit is set to 0 and the receiver managed to change its bit, then the receiver’s bit is set to 0” (true). In

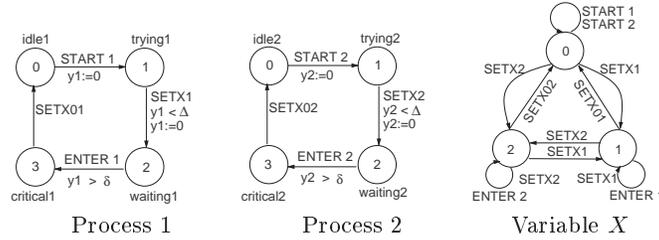


Fig. 2. Fischer’s Mutual Exclusion Protocol for two processes

the case of the Mutual Exclusion Protocol we have used an input in the intermediate language. The timed automata returned by the $\sqrt{\text{erics}}$ translation of this specification are presented in Fig. 2. We have tested them for various values of the parameters, as only if $\Delta < \delta$, then the mutual exclusion property is ensured. In case of the system RCS, the three system automata of the train, gate and controller, and the automaton for a property (see Fig. 3) have been used as the input. The property automaton describes that “whenever the gate is down it is moved back up within K seconds”, which is satisfied for $K < 700$.

The experimental results are presented in Fig. 4. In all the examples, the BBMC Reachability Analyser has been used to show that a state satisfying a tested property is reachable, while the Splitter module has been applied to generate the whole model of the system, i.e., to show that no state satisfying the property

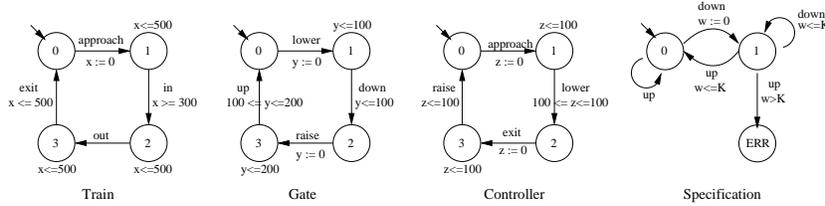


Fig. 3. The automata for the railroad crossing system and a specification

	Alternating Bit				Mutex				RCS							
	ARG		ZChaff		ARG		ZChaff		ARG		ZChaff					
	min	MB	min	MB	NP	δ	Δ	min	MB	min	MB	K	min	MB	min	MB
BBMC	0.21	1.38	0.16	5.62	39	1	2	34.30	382.10	168.3	684.8	500	0.67	5.94	0.08	10.62
	states		edges		NP	δ	Δ	states	edges	K	states	edges				
Splitter	2332		8267		5	2	1	9849	38545	800	18	24				

Fig. 4. Experimental results

can be reached. We provide the times and the amounts of memory needed for BBMC Reachability Analyser to prove reachability of the state, and the sizes of pseudo-bisimulating models generated by Splitter. For the Mutual Exclusion Protocol, NP denotes the number of processes which have been tested. Among all the examples we considered, the Mutual Exclusion Protocol is most widely covered in the literature. Unfortunately, we have not been able to compare our results with ones of other SAT-based model checkers, due to the lack of such experiments in the literature for the property we have tested. However, for the same protocol and another property we obtained better results than the ones of [2] (34 processes instead of 19 only). This allows to expect better results also in other cases. On the other hand, our pseudo-bisimulating models are usually smaller than forward-reachability and bisimulating models generated by Kronos. A web page of our tool is available at <http://www.ipipan.waw.pl/~penczek/verics>.

References

1. R. Alur and D. Dill. Automata for modelling real-time systems. In *Proc. of the International Colloquium on Automata, Languages and Programming (ICALP'90)*, volume 443 of *LNCS*, pages 322–335. Springer-Verlag, 1990.
2. G. Audemard, A. Cimatti, A. Kornilowicz, and R. Sebastiani. Bounded model checking for timed systems. In *Proc. of the 22nd Int. Conf. on Formal Techniques for Networked and Distributed Systems (FORTE'02)*, volume 2529 of *LNCS*, pages 243–259. Springer-Verlag, 2002.
3. J. Bengtsson, K. Larsen, F. Larsson, P. Pettersson, W. Yi, and C. Weise. New generation of UPPAAL. In *Proc. of the Int. Workshop on Software Tools for Technology Transfer*, 1998.
4. D. Beyer. Rabbit: Verification of real-time systems. In *Proc. of the Workshop on Real-Time Tools (RT-TOOLS'01)*, pages 13–21, 2001.

5. M. Bozga, J-C. Fernandez, L. Ghirvu, S. Graf, J.P. Krimm, L. Mounier, and J. Sifakis. IF: An intermediate representation for SDL and its applications. In *Proc. of SDL Forum'99*, pages 423–440, 1999.
6. A. Cimatti, E. M. Clarke, F. Giunchiglia, F. Giunchiglia, M. Pistore, M. Roveri, R. Sebastiani, and A. Tacchella. NuSMV 2: An open-source tool for symbolic model checking. In *Proc. of CAV'02*, volume 2404 of *LNCS*, pages 359–364. Springer-Verlag, 2002.
7. A. Doroś, A. Janowska, and P. Janowski. From specification languages to Timed Automata. In *Proc. of the Int. Workshop on Concurrency, Specification and Programming (CS&P'02)*, volume 161(1) of *Informatik-Berichte*, pages 117–128. Humboldt University, 2002.
8. H. Garavel, F. Lang, and R. Mateescu. An overview of CADP 2001. Technical Report RT-254, INRIA Rhône-Alpes, 655, avenue de l'Europe, 38330 Montbonnot-St-Martin, December 2001.
9. G. J. Holzmann. The model checker SPIN. *IEEE Trans. on Software Eng.*, 23(5):279–295, 1997.
10. ISO 8807 - information processing systems - Open System Interconnection. LOTOS - a formal description technique based on the temporal ordering of observational behaviour, 1989.
11. ISO/IEC 9074(E), Estelle - a formal description technique based on an extended state-transition model. International Standards Organization, 1997.
12. Languages for telecommunication applications - Specification and Description Language, 1999.
13. K. McMillan. The SMV system. Technical Report CMU-CS-92-131, Carnegie-Mellon University, February 1992.
14. M. Moskewicz, C. Madigan, Y. Zhao, L. Zhang, and S. Malik. Chaff: Engineering an efficient SAT solver. In *Proc. of the 38th Design Automation Conference (DAC'01)*, pages 530–535, June 2001.
15. W. Penczek, B. Woźna, and A. Zbrzezny. SAT-based bounded model checking for the universal fragment of TCTL. Technical Report 947, ICS PAS, Ordonia 21, 01-237 Warsaw, August 2002.
16. W. Penczek, B. Woźna, and A. Zbrzezny. Towards bounded model checking for the universal fragment of TCTL. In *Proc. of the 7th Int. Symp. on Formal Techniques in Real-Time and Fault Tolerant Systems (FTRTFT'02)*, volume 2469 of *LNCS*, pages 265–288. Springer-Verlag, 2002.
17. A. Pólrola, W. Penczek, and M. Szreter. Reachability analysis for Timed Automata based on partitioning. Technical report, ICS PAS, Ordonia 21, 01-237 Warsaw, 2003. to appear.
18. R. Sebastiani. Integrating SAT solvers with math reasoners: Foundations and basic algorithms. Technical Report 0111-22, ITC-IRST, Sommarive 16, 38050 Povo, Trento, Italy, November 2001.
19. B. Woźna, W. Penczek, and A. Zbrzezny. Checking reachability properties for Timed Automata via SAT. Technical Report 949, ICS PAS, Ordonia 21, 01 - 237 Warsaw, October 2002.
20. B. Woźna, W. Penczek, and A. Zbrzezny. Reachability for timed systems based on SAT-solvers. In *Proc. of the Int. Workshop on Concurrency, Specification and Programming (CS&P'02)*, volume 161(2) of *Informatik-Berichte*, pages 380–395. Humboldt University, 2002.
21. S. Yovine. KRONOS: A verification tool for real-time systems. *Springer International Journal of Software Tools for Technology Transfer*, 1(1/2):123–133, 1997.