

Verics 2004: A Model Checker for Real Time and Multi-agent Systems ^{*}

Wojciech Nabiałek², Artur Niewiadomski², Wojciech Penczek^{1,2},
Agata Pórola³, and Maciej Szreter¹

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

² Institute of Informatics, Podlasie Academy, Sienkiewicza 51, 08-110 Siedlce, Poland
{wojtek,artur}@iis.ap.siedlce.pl

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

Abstract. VerICS is a model checking tool for verification of timed and multi-agent systems. These systems can be represented by timed automata (TA), time Petri nets (TPNs), or given as Estelle and Intermediate Language specifications. VerICS offers three verification methods: Bounded Model Checking (BMC), Unbounded Model Checking (UMC) and Splitting for properties to be specified in subsets of TCTL and CTL_pK. The current version of VerICS uses also a new graphical user interface to design time Petri nets and timed automata.

1 Introduction

Automated verification of finite-state concurrent systems performed by analysis of their models (model checking) is, due to its practical applicability, a very important subject of research. Among these systems, the time-dependent and multi-agent ones belong to the most intensively investigated. Essentially, in this formalism verifying that a property follows from a system specification amounts to checking whether or not a temporal formula is valid on a model representing all possible computations of the system. However, practical applicability of model checking methods is strongly limited by the *state explosion problem*. For real-time systems, the problem occurs with a particular strength, which follows from infinity of the time domain. Therefore, existing verification techniques frequently apply symbolic representations of state spaces of these systems, using either operations on Difference Bound Matrices [11], variations of Boolean Decision Diagrams [5, 26], or SAT-related algorithms. The latter can exploit either a

^{*} In addition to the authors of this paper, the following researchers contributed to the development of VerICS: P. Dembiński, A. Janowska, P. Janowski, M. Kacprzak, B. Woźna, and A. Zbrzezny.

The work is partly supported by the State Committee for Scientific Research under the grant No. 3T11C 004 26

sequence of translations starting from a representation of the system in the form of timed automata and a formula of the logic TCTL, going via (quantified) separation logic to quantified propositional logic and further to propositional logic [4, 16, 24], or a direct translation from timed automata and TCTL to propositional logic [19, 28, 30]. Finite state spaces, preserving properties to be checked, are usually built using detailed region graph approach or (possibly minimal) abstract models based on state classes or regions. Algorithms for generating such models have been defined for time Petri nets as well as for timed automata (see [18] for a survey). Verification methods for timed and multi-agent systems are implemented in many tools [3, 6, 7, 13, 20, 29].

The purpose of this paper is to present a new version of the model checker VerICS, which is now a verification tool for real-time and multi-agent systems. VerICS implements SAT-based verification methods as well as an on-the-fly verification on abstract models. We provide also a short overview of VerICS capabilities by presenting some experimental results.

The rest of the paper is organized as follows. In Section 2 related tools and specification languages are described. Section 3 introduces VerICS, whereas in Section 4 we present the architecture of the system. Section 5 contains case studies and experimental results.

2 Existing Tools

In this section we discuss shortly some existing tools for verifying timed automata, time Petri nets, and multi-agent systems. Concerning **timed automata**, there exist many model checkers, so we mention the most popular ones. **Cospan** is a tool which enables verification of designs written in the industry standard languages VHDL and Verilog, and implements an automata-based approach to model checking, including an on-the-fly enumerative search as well as symbolic search using Binary Decision Diagrams. **Kronos** performs verification of TCTL using forward or backward analysis, and behavioural analysis, which consists in building the smallest finite quotient of a timed model, and then checking whether the minimal model of the system simulates that of the specification. Forward reachability analysis, deadlock detection and verification of properties expressible in a subset of TCTL are available also in **UppAal2k**. It is a tool for modelling, simulation and verification of timed systems which can be described by a collection of non-deterministic processes with finite control structure and real-valued clocks, communicating through channels or shared variables. **HyTech** is an automatic tool for the analysis of embedded systems. Real-time requirements are expressed in the logics TCTL and ICTL, used to specify safety, liveness, time-bounded and duration requirements of hybrid automata. Verification is performed by a successive approximation of the set of states satisfying the formula to be checked, by iterating boolean and weakest-precondition operations. Another model checker is **Spin** with the input given in Promela (PROtocol MEta LAnguage) - a high level language which can be used for describing and specifying communication protocols, and with the model checking module im-

plementing an automata-theoretic approach for LTL. The well-known tools for **Petri nets with time** include **Tina** - a toolbox for analysis of (time) Petri nets, which constructs state class graphs (abstract models) and exploits them for LTL, CTL or reachability verification, **Romeo** - a tool for time Petri nets analysis, which provides several methods for translating TPNs to TA and computation of state class graphs, and **CPN Tools**, which is a software package for modelling and analysis of both timed and untimed Coloured Petri Nets, enabling their simulation, generating occurrence (reachability) graph, and analysis by place invariants. Among tools for **multi-agent systems** one can mention **CASP**, which is a toolkit for modelling the multi-agent systems with logic programming language named AgentSpeak(L). This software allows translation from AgentSpeak(L) to Promela or Java language and further verification with SPIN or Java PathFinder2. **MABLE** is a language for the modelling and automatic verification of multi-agent systems. Its implementation enables translation to Promela, which is then used as an input for SPIN. A tool for BDD-based verification of multi-agent systems is **MCK**, using LTL as a temporal language.

3 Theory Behind Verics

The theoretical background for our implementation has been presented in several papers [10, 12, 15, 19, 28, 30]. In this section we present the main ideas. Our tool accepts an input written as specification in a subset of Estelle, in Intermediate Language, networks of timed automata, time Petri nets, and Petri nets. Estelle is an ISO standard specification language designed for describing communications protocols and distributed systems. Intermediate Language (IL) [12] 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 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 [12]. The next step in the translation process are *timed automata* [1], which are an extension of the standard finite-state automata with constraints on timing behaviour, obtained by augmenting them with sets of real-time variables. A system described in Intermediate Language can be translated either to a set of timed automata, each of which represents a component of the system, or to a global (product) timed automaton (for the description see [12]). The automata obtained are then passed to other components of Verics which are aimed at performing reachability or temporal (epistemic) logic model checking. Our tool offers three complementary methods of verification: SAT-based Bounded Model Checking (BMC), SAT-based Unbounded Model Checking (UMC), and an on-the-fly reachability checking while constructing abstract models of systems.

The BMC-based method combines the well-known forward reachability analysis and the bounded model checking method for timed automata [19, 28, 27].

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. In case when a tested property does not hold, the SAT-based method can be ineffective. Therefore, in parallel to BMC, Verics offers two other verification methods: UMC, consisting in SAT-based encoding of the whole state space [15], and a method of generating finite abstract models for timed automata using a partitioning algorithm combined with on-the-fly reachability analysis [22].

4 Architecture of Verics

A schematic description of the system, showing dependencies between its modules, as well as inputs supported, is presented in Fig. 1. The main modules of Verics include:

- Estelle to Intermediate Language translator,
- Intermediate Language to timed automata translator,
- Bounded Model Checker,
- Unbounded Model Checker,
- Splitter.

Properties of timed systems are expressed in subsets of TCTL, whereas properties of multi-agent systems are specified in CTL_pK . The logic TCTL is an extension of CTL_X obtained by subscribing the modalities with time intervals specifying time restrictions on formulas. The logic CTL_pK is an extension of CTL with past modalities and an epistemic component. TECTL and $ECTL_pK$ denote, respectively, the existential fragments of TCTL and CTL_pK .

Comparing with the versions described in [8, 9], Verics has been extended by several new capabilities. The main novelties are:

- Verification of multi-agent systems,
- New property specification language: CTL_pK ,
- Unbounded Model Checking module,
- Petri nets as a new input,
- New user interface.

In the following subsections, we provide a more detailed description of the main components of Verics.

4.1 Estelle to Intermediate Language Translator

At this moment a subset of Estelle can be translated to Verics Intermediate Language. This translation is rather straightforward - execution models and syntax of those formalisms are similar. Instances of modules are translated to processes. Each local variable of an Estelle module instance becomes a local variable of corresponding Intermediate Language process. Parameters of each interaction which can be sent or received by a module are explicitly defined as local variables. All

Estelle control states are translated into Intermediate Language control states. The structure of an Estelle transition and an Intermediate Language transition is also similar - guards are expressed as formulas depending on channels (buffers) contents and values of local variables. As a body of Intermediate Language transition uses simpler operations than Estelle transitions (there are only three possibilities: input/output statements or assignments), additional states are generated and complex instructions are serialized. Estelle timing conditions, expressed as delays, are translated directly to intermediate language delays.

4.2 Intermediate Language to Timed Automata Translator

The translator builds either a global timed automaton describing the whole system, or a network of local timed automata, each of which represents a component of the system given as an Intermediate Language specification. A description of the system in terms of local components provides more compact representation and can be exploited by model checking methods which take advantage of locality to avoid the state explosion problem. The resulting timed automata are standard formalism to model the behaviour of real-time systems over time proposed in [2]. The only difference concerns a composition method. In our approach sets of clocks do not need to be disjoint. A state of global automaton is defined by values of all variables and contents of all buffers. A transition relation reflects program actions. Clocks are employed to represent delays on transitions. The details about the translation can be found in [12]. The tool can be also exploited in another way. In the case of systems without any time restrictions, the automaton generated from an Intermediate Language specification is a model of a system and it is possible to apply standard model checking algorithms to it.

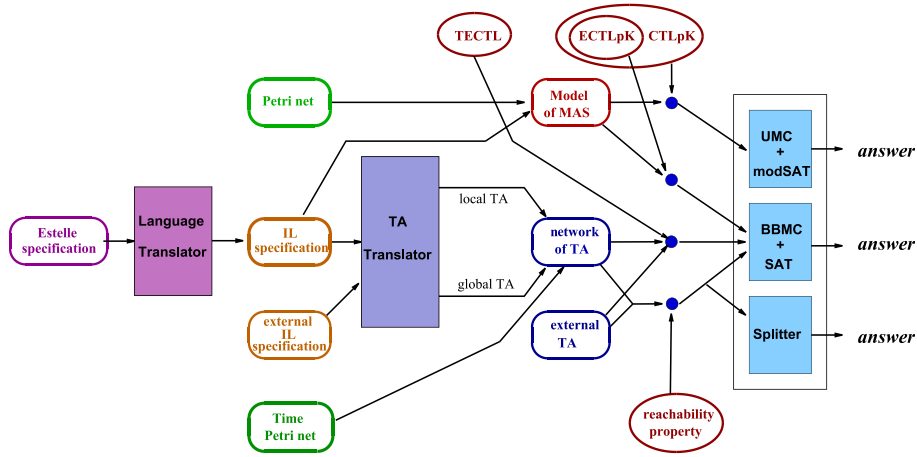


Fig. 1. Architecture of VerICS

4.3 Bounded Model Checking Module

BMC is based on an observation that some properties of systems can be checked without searching the whole state space. In the simplest case of reachability analysis, the approach consists in an iterative encoding of a symbolic path¹ representing a fragment of the model combined with a verified property. The satisfiability of the resulting propositional formula is then checked using an external SAT solver.

Unfortunately, BMC is unable to handle alternation in a specification language, therefore only an existential fragment of the logic TCTL (for real-time systems) or CTL_pK (for MAS) can be verified. Thus, BMC is considered to be a falsification algorithm, that means it works much better when falsifying a property than proving that it holds on the whole model.

4.4 Unbounded Model Checking Module

Contrary to BMC, UMC processes the whole state space. In this approach, sets of states that satisfy the same modal formulas are represented by propositional formulas. These sets are processed using appropriate transformations, e.g. a fixed point computation in case of modal operators. The core of the method is based on an efficient algorithm removing universal quantifiers from Quantified Boolean Formulas exploiting the successful Davis-Logemann-Loveland SAT algorithm.

When the fixpoint computation and quantifier elimination finishes, the resulting formula describes the set of states in which the verified property holds. This formula is conjuncted with the formula representing initial state and checked for satisfiability. If it is satisfiable, the property holds in the model. The detailed definition of encoding for temporal operators can be found in [15]. The current version of Verics supports only UMC for properties in CTL_pK of untimed systems (see [14] for details).

4.5 Splitter Module

The Splitter module enables generating several kinds of abstract models for timed automata. The idea of abstract models consists of combining into classes concrete states that are indistinguishable w.r.t. properties to be tested. For generating bisimulating models, preserving the language of CTL^* , a minimization (partitioning) algorithm is usually used. Some modifications of this algorithm [10, 22, 23] are implemented in the module, enabling building other kinds of models, e.g. simulating ones [17], which preserve the language of $ACTL^*$. In order to test reachability, pseudo-bisimulating [22] and pseudo-simulating [23] models are applied. They are generated in a BFS-like manner, which enables on-the-fly verification. Moreover, pseudo-simulating models are built for a *discrete* semantics, which is sufficient for testing reachability, and in some cases allows to obtain better results (i.e., smaller models). The module cannot be used for verification of multi-agent systems.

¹ Temporal operators usually require more symbolic paths to be encoded

4.6 Other Modules

VerICS includes also other modules such as user interfaces, parsers, translators and generators of case study examples. Currently, two user interfaces are available: a web-based and a new graphical one. The latter is based on Java Swing technology, and in the nearest future will completely replace the previous one.

The new GUI enables a simple design of specifications at the level of automata and Petri nets. Their textual representation is saved as an XML file.

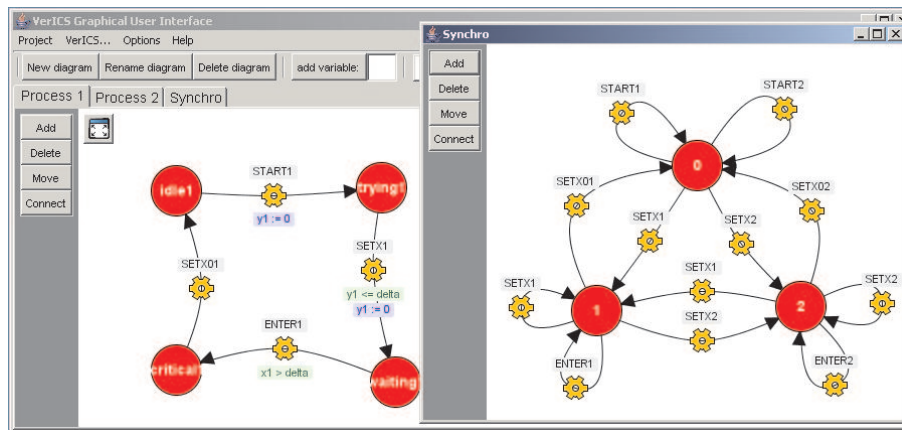


Fig. 2. Timed automata graphical editor

5 Case Studies

This section gives a short overview of the capabilities of VerICS. We provide descriptions and experimental results for two examples: The Train Controller System and Fischer's mutual exclusion algorithm.

5.1 Experimental Results for Verifying Multi-agent Systems

The tests presented below were performed on a workstation equipped with the AMD Athlon XP+ 2800 CPU and 2 GB RAM running under Fedora Linux OS.

We have adopted the Train Controller System scenario from [25] in order to demonstrate how the tool handles scaling of models. There are two kinds of agents participating in the scenario: *the trains* and *the controller*.

Generally, trains occupy different tracks, but sometimes they have to go over a narrow tunnel in the mountains. There is no room for more than one train to be in the tunnel at the same time. There are traffic lights on both sides of

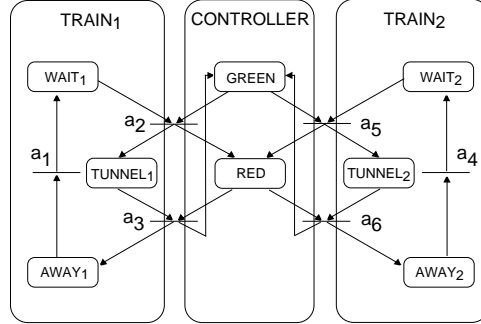


Fig. 3. The Train Controller System

the tunnel, which can either be red or green. The trains are equipped with a signaler, that they use to send a signal when they approach the tunnel. The controller can receive signals from all the trains, and controls the colour of the traffic lights. The task of the controller is to ensure that the trains are never both in the tunnel at the same time. The trains follow the traffic lights signals diligently, i.e., they stop on red.

We model the example above with an interpreted system. Figure 3 shows the models of the agents participating in the scenario for two trains. The example is scaled by increasing the number of trains and extending the model of the controller by adding new transitions.

For the scenario above we have chosen the property which can be scaled together with the model, when we increase number of trains. Our aim was to compare both the UMC and BMC methods on this example. The property states that if train 1 is in the tunnel, then it knows whether there is another train in the tunnel. This property obviously does not hold in our model. The results for UMC and BMC are displayed in Table 1 and Table 2, respectively.

Number of trains	CNF clauses	UMC-mem	UMC-time	Solver time
2	557	2260 KB	0.12 s	0.01 s
4	5214	8376 KB	1.51 s	0.01 s
6	58489	64 MB	46.55 s	0.01 s

Table 1. Performance results of UMC for the model of Train Controller System

Number of trains	depth	Size of formula (vars/clauses)	Solver time
10	1	453/1195	0
	2	811/2199	0
100	1	8958/25270	0.03
	2	16921/48459	2.43
300	1	56858/165770	0.32
	2	110721/325259	60.34

Table 2. Performance results of BMC for the model of Train Controller System

5.2 Experimental Results for Verifying TPNs and TA

In this section we compare experimental results for the four TPNs of Figure 4, and for the TA of Figure 5 modelling Fischer's mutual exclusion algorithm.

Examples of (distributed) time Petri net are shown in Fig. 4. Each of the nets 5a, 5b, and 5c in the left-hand side consists of two disjoint processes with the sets of places $P^1 = \{p_1, p_2, p_3, p_4\}$ and $P^2 = \{p_6, p_7\}$, whereas the net on the right is composed of three communicating processes with the sets of places: $P^i = \{idle_i, trying_i, enter_i, critical_i\}$ with $i = 1, 2$, and $P^3 = \{place0, place1, place2\}$.

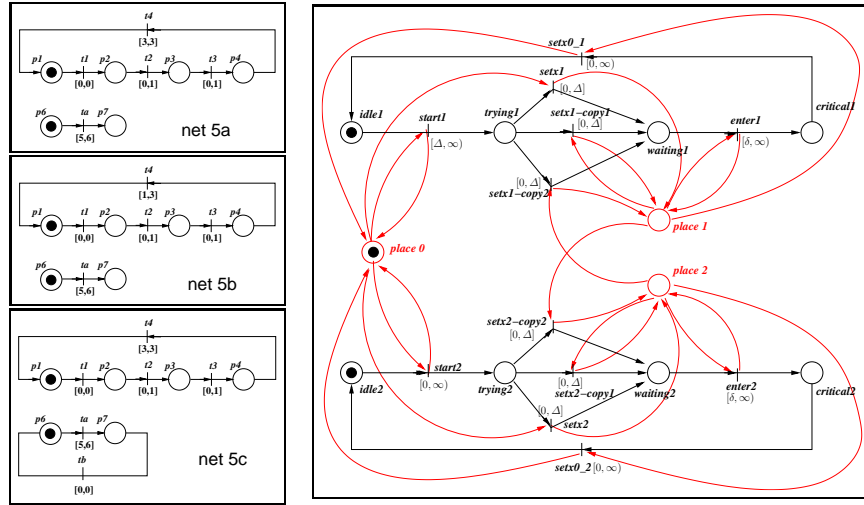


Fig. 4. Distributed time Petri nets and a net for Fischer's mutual exclusion protocol

In Fig. 5, a network of TA for Fischer's mutual exclusion protocol (mutex) with two processes is depicted². The protocol is parameterised by the number of processes involved. In the general case, the network consists of n automata of processes, together with one automaton modelling a global variable X , used to coordinate the processes' access to their critical sections.

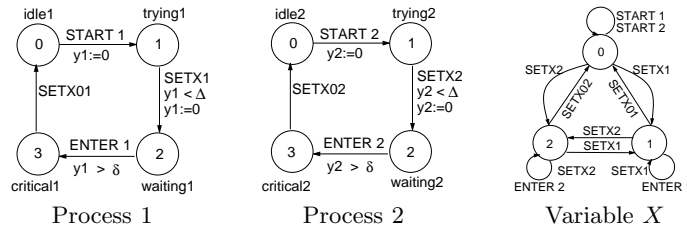


Fig. 5. Fischer's Mutual Exclusion Protocol for two processes

² Two clocks are denoted by y_1 and y_2 , whereas Δ and δ are parameters.

		Net 5a		Net 5b		Net 5c		Mutex $\Delta=1, \delta=2$			Mutex $\Delta=2, \delta=1$		
		states	edges	states	edges	states	edges	noP	states	edges	noP	states	edges
OBTAINED BY TPN \rightarrow TA TRANSLATIONS													
VerICS	bis. dense	54	80	135	230	186	323	3	77	108	3	200	312
VerICS	bis. discr.	26	47	46	135	80	204	3	65	96	3	152	240
VerICS	ps- discr.	21	34	13	22	53	121	3	65	96	3	152	204
Kronos	bis. dense	51	77	134	229	185	321	3	77	108	3	200	312
Kronos	forw -ai -ax	37	42	37	42	26	40	3	214	321	3	613	1084
								5	33451	62223	4	12850	27848

Table 3. Experimental results for the nets in Fig. 4

In Table 3 we give the sizes of several abstract models for all the nets, obtained using minimization algorithms (VerICS, Kronos) applied to translations³ to timed automata as well as directly to the TA of Figure 5. The experiments were performed on a PC (Intel Pentium III 640MHz), with the assumed limit on the execution time (30 min) and memory required (128 MB RAM + 128 MB of swap space under Linux). For the Fischer’s protocol, we give the sizes of models for 3 processes as well as for the maximal number of processes a model could be generated for.

Table 4 displays the results of applying the BMC algorithm (for checking reachability and unreachability) to timed systems modelling Fischer’s mutual exclusion. We verify that either mutual exclusion is violated for $\Delta = 2$ and $\delta = 1$, or is preserved for $\Delta = 1$ and $\delta = 2$. BMC is applied either directly to the TA, or to the timed automaton resulting from the translation of the TPN. For the case of $\Delta = 2$ and $\delta = 1$ we provide the time and memory resources needed to confirm satisfiability of the property on a path of length $k = 17$, whereas for $\Delta = 1$ and $\delta = 2$ the time given is sum of the times required to learn that it is sufficient to test reachability on a path of the length $k = 44$ and then to check satisfiability on the path of that length (see [30] for a description of the method). These experiments were performed on a PC (AMD Athlon XP 1800 - 1544MHz).

Parameters	NoP	TPN \rightarrow TA				TA			
		variables	clauses	sec	MB	variables	clauses	sec	MB
$\Delta = 1, \delta = 2$	8	61530	176319	10890.1	61.31	36461	103228	2326.3	34.5
$\Delta = 2, \delta = 1$	8	22552	64442	8.0	21.9	13357	37666	0.7	20.5
$\Delta = 2, \delta = 1$	10	29918	86002	14.5	23.5	17283	49034	1.1	20.2
$\Delta = 2, \delta = 1$	50	378203	1118763	99.7	100.5	156941	459722	21.8	31.7
$\Delta = 2, \delta = 1$	104	1411156	4200809	1397.6	577.9	528136	1562194	218.8	75.9
$\Delta = 2, \delta = 1$	310	-	-	-	-	3873940	11557290	21723.6	648.3

Table 4. BMC of VerICS for Fischer’s protocol modelled by TPN and TA

³ Processes-as-clocks translations of [21] are used.

Notice that the BMC could verify MUTEX modelled by the TPN or the TA for $\Delta = 1, \delta = 2$ of 8 processes, and respectively of 104 and 310 processes, where the mutual exclusion was violated ($\Delta = 2$ and $\delta = 1$).

References

1. R. Alur and D. Dill. Automata for modelling real-time systems. In *Proc. of the Int. Colloquium on Automata, Languages and Programming (ICALP'90)*, volume 443 of *LNCS*, pages 322–335. Springer-Verlag, 1990.
2. R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
3. R. Alur and R. Kurshan. Timing analysis in COSPAN. In *Hybrid Systems III*, volume 1066 of *LNCS*, pages 220–231. Springer-Verlag, 1996.
4. 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.
5. G. Behrmann, K. Larsen, J. Pearson, C. Weise, and W. Yi. Efficient timed reachability analysis using Clock Difference Diagrams. In *Proc. of the 11th Int. Conf. on Computer Aided Verification (CAV'99)*, volume 1633 of *LNCS*, pages 341–353. Springer-Verlag, 1999.
6. B. Berthomieu. The Tina v2 toolbox. <http://www.laas.fr/tina>, 2001.
7. R. Bordini, M. Fisher, C. Pardavila, W. Visser, and M. Wooldridge. Model checking multi-agent programs with CASP. In *Proc. of the 15th Int. Conf. on Computer Aided Verification (CAV'03)*, volume 2725 of *LNCS*, pages 110–113. Springer-Verlag, 2003.
8. P. Dembiński, A. Janowska, P. Janowski, W. Penczek, A. Pórola, M. Sreter, B. Woźna, and A. Zbrzezny. VerICS: A tool for verifying timed automata and Estelle specifications. In *Proc. of the 9th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'03)*, volume 2619 of *LNCS*, pages 278–283. Springer-Verlag, 2003.
9. P. Dembiński, A. Janowska, P. Janowski, W. Penczek, A. Pórola, M. Sreter, B. Woźna, and A. Zbrzezny. VerICS: weryfikator dla automatów czasowych i specyfikacji zapisanych w języku Estelle. In *Mat. X Konf. Systemy Czasu Rzeczywistego (SCR'03)*, pages 17–26. Instytut Informatyki Politechniki Śląskiej, 2003. In Polish.
10. P. Dembiński, W. Penczek, and A. Pórola. Verification of timed automata based on similarity. *Fundamenta Informaticae*, 51(1-2):59–89, 2002.
11. D. Dill. Timing assumptions and verification of finite state concurrent systems. In *Automatic Verification Methods for Finite-State Systems*, volume 407 of *LNCS*, pages 197 – 212. Springer-Verlag, 1989.
12. 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.
13. T. Henzinger and P. Ho. HyTech: The Cornell hybrid technology tool. In *Hybrid Systems II*, volume 999 of *LNCS*, pages 265–293. Springer-Verlag, 1995.
14. M. Kacprzak, A. Lomuscio, T. Łasica, W. Penczek, and M. Sreter. Verifying multiagent systems via unbounded model checking. In *Proc. of the 3rd NASA Workshop on Formal Approaches to Agent-Based Systems (FAABS III)*, LNCS. Springer-Verlag, 2004. To appear.

15. M. Kacprzak, A. Lomuscio, and W. Penczek. Unbounded model checking for knowledge and time. In L. Czaja, editor, *Proc. of the Int. Workshop on Concurrency, Specification and Programming (CS&P'03)*, volume 1, pages 251–264. Warsaw University, 2003.
16. P. Niebert, M. Mahfoudh, E. Asarin, M. Bozga, O. Maler, and N. Jain. Verification of timed automata via satisfiability checking. In *Proc. of the 7th Int. Symp. on Formal Techniques in Real-Time and Fault Tolerant Systems (FTRTFT'02)*, volume 2469 of *LNCS*, pages 226–243. Springer-Verlag, 2002.
17. W. Penczek. Partial order reductions for checking branching properties of time Petri nets. In *Proc. of the Int. Workshop on Concurrency, Specification and Programming (CS&P'00)*, volume 140 of *Informatik-Berichte*, pages 189–202. Humboldt University, 2000.
18. W. Penczek and A. Pólrola. Specification and model checking of temporal properties in time Petri nets and timed automata. In *Proc. of the 25th Int. Conf. on Applications and Theory of Petri Nets (ICATPN'04)*, volume 3099 of *LNCS*, pages 37–76. Springer-Verlag, 2004.
19. 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.
20. P. Pettersson and K. G. Larsen. UPPAAL2k. *Bulletin of the European Association for Theoretical Computer Science*, 70:40–44, February 2000.
21. A. Pólrola and W. Penczek. Minimization algorithms for time Petri nets. *Fundamenta Informaticae*, 60(1-4):307–331, 2004.
22. A. Pólrola, W. Penczek, and M. Szreter. Reachability analysis for timed automata using partitioning algorithms. *Fundamenta Informaticae*, 55(2):203–221, 2003.
23. A. Pólrola, W. Penczek, and M. Szreter. Towards efficient partition refinement for checking reachability in timed automata. In *Proc. of the 1st Int. Workshop on Formal Analysis and Modeling of Timed Systems (FORMATS'03)*, volume 2791 of *LNCS*, pages 2–17. Springer-Verlag, 2004.
24. S. Seshia and R. Bryant. Unbounded, fully symbolic model checking of timed automata using boolean methods. In *Proc. of the 15th Int. Conf. on Computer Aided Verification (CAV'03)*, volume 2725 of *LNCS*, pages 154–166. Springer-Verlag, 2003.
25. W. van der Hoek and M. Wooldridge. Tractable multiagent planning for epistemic goals. In *Proc. of the 1st Int. Conf. on Autonomous Agents and Multi-Agent Systems (AAMAS'02)*, volume III, pages 1167–1174. ACM, July 2002.
26. F. Wang. Region Encoding Diagram for fully symbolic verification of real-time systems. In *Proc. of the 24th Int. Computer Software and Applications Conf. (COMPSAC'00)*, pages 509–515. IEEE Computer Society, October 2000.
27. 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.
28. B. Woźna, A. Zbrzezny, and W. Penczek. Checking reachability properties for timed automata via SAT. *Fundamenta Informaticae*, 55(2):223–241, 2003.
29. 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.
30. A. Zbrzezny. Improvements in SAT-based reachability analysis for timed automata. *Fundamenta Informaticae*, 60(1-4):417–434, 2004.