Automatic Verification of Real-Time Systems with Rich Data: An Overview*

Ernst-Rüdiger Olderog

Department of Computing Science, University of Oldenburg, Germany
olderog@informatik.uni-oldenburg.de

Abstract. We present an overview of the results of the project "Beyond Timed Automata" of the Collaborative Research Center AVACS (Automatic Verification and Analysis of Complex Systems) during the period 2008–2011, which advances the automatic verification of high-level specifications of systems exhibiting the three dimensions of process behavior, complex infinite data, and continuous real-time—beyond the capabilities of Timed Automata.

1 Introduction

Computers are needed to control the behavior of complex systems, for instance in the traffic domain, where assistance systems should guarantee the collision freedom of traffic agents such as cars, trains, and planes. Such applications are safety critical, i.e., a malfunction of the computers is costly and dangerous. These applications necessitate the use of formal models of the overall system and of formal verification for establishing the relevant safety properties. The models must be able to represent various aspects of the systems such as state spaces and their transformation, communication between system components, real-time constraints, interfaces to a continuously evolving physical environment, and dynamically changing system structures. To cope with such models in a manageable way, combined specification techniques have been proposed, integrating well researched specification techniques for individual system aspects. It is a major research challenge to develop methods for the automatic verification and analysis of such combined specifications modeling complex real-life systems.

To address this challenge the research center AVACS (Automatic Verification and Analysis of Complex Systems) was founded in 2004. In this center, researchers of the Universities of Oldenburg, Freiburg and Saarbrücken as well as the Max-Planck-Institute for Informatics in Saarbrücken collaborate. AVACS brings together experts in semantic modeling and specification with experts in verification and analysis techniques.

* This work was partly supported by the German Research Council (DFG) as part of the Transregional Collaborative Research Center "Automatic Verification and Analysis of Complex Systems" (SFB/TR 14 AVACS, http://www.avacs.org/).

M. Agrawal, S.B. Cooper, and A. Li (Eds.): TAMC 2012, LNCS 7287, pp. 84–93, 2012.

In the following we give an overview of the results of one of the projects in the research area R (Real-Time) achieved during Phase 2 of AVACS (2008–2011): the project R1 "Beyond Timed Automata" that advances the automatic verification of high-level specifications of systems exhibiting the three dimensions of process behavior, complex infinite data, and continuous real-time—beyond the capabilities of Timed Automata. To this end, transformation and decomposition techniques are combined with enhanced proof procedures resting on the paradigms of abstraction refinement and local theory extensions. This paper complements the more technical overview of Phase 1 of R1 presented in [32].

2 Overview of the Project R1

The general set-up of the project is as follows. For the specification of real-time systems, the language CSP-OZ-DC (or COD for short) integrating aspects of Communicating Sequential Processes (CSP), Object-Z (OZ), and Duration Calculus (DC) has been developed [23]. We use the automata-theoretic approach for automatic verification of systems against real-time requirements, whereby both the system and the requirement are transformed into a semantically equivalent parallel composition of Phase-Event-Automata (PEA), which allow for complex data in their phases [21]. PEA are the stepping stone for further transformations into the input languages of verification engines developed in the project. These engines are the model checkers ARMC [35] and SLAB [11], and the tool H-PILoT [25] for dealing with complex data; they implement the paradigms of abstraction refinement and local theory extensions. The graphical tool Syspect realizes a tool chain from COD down to these verification engines [15].

The publication [31] reflects the state of automatic verification in R1 *at the start of* Phase 2. Then a subclass of DC—involving *counterexample formulae* and allowing for Boolean combinations of timed phases—could be used as real-time conditions inside CSP-OZ-DC (COD) specifications. In particular, the translation of counterexample formulae into PEA involves a sophisticated power set construction to cope with the nondeterminism arising from overlapping phases. PEA are translated further into Transition Constraint Systems (TCS) serving as input to the model checkers ARMC (Abstraction-Refinement Model Checker) and SLAB (Slicing-Abstraction Model Checker) that use Craig interpolation and decision procedures for data in order to refine their abstractions. Some reductions of the state spaces were achieved by applying a priori slicing techniques to COD specifications [4,5].

The viability of the whole approach has been demonstrated on the case study of *Emergency Messages* between two trains and a radio block center (RBC) in the context of the European Train Control System (ETCS) at Level 3 [31]. Besides continuous real-time, this case study involved infinite scalar data types for the position and speed, communications of these data between trains and the RBC, as well as parameters for the length and target speed of the trains. We could automatically verify real-time requirements of the system with ARMC and SLAB. Collision freedom could not be proven push-button, but required a

manual decomposition into real-time requirements that in turn could be verified automatically [31].

While each of the ETCS real-time requirements depends only on a subset of all the parallel PEA, the requirement of collision freedom depends on the full set of parallel PEA. During Phase 1, the parallel product of PEA needed to be computed before translating the result into TCS because neither ARMC nor SLAB could process a parallel composition directly. Thus for large real-time systems, the state space explosion arising from the parallel product of the PEA limited the applicability of the approach.

In Phase 2, the project R1 advanced automatic verification of real-time systems with complex data in the following directions.

- *Explicit durations.* The class of real-time requirements amenable to automatic verification has been extended to formulae with *explicit durations.*
- *Structural optimization.* Two approaches to counteract the problem of state space explosion arising from the calculation of the parallel composition of PEA have been developed, employing *layered composition* and *verification architectures.*
- *Automating verification.* Novel concepts supporting the automatic verification of systems have been developed, based on *subsequence invariants*, *refinement of trace abstraction*, and *nested interpolants.*
- *Complex data.* The scope of data that can be handled automatically has been extended considerably by the method of *local theory extension.*
- *Verification tools.* The model checker SLAB, pioneered in Phase 1 of R1, has been extended to exploit structural information. The new tool H-PILoT supports hierarchical reasoning in chains of local theory extensions. The graphical tool Syspect builds bridges from COD to these verification tools.
- *Case studies.* We mastered the state space explosion problem in the case study involving ETCS Emergency Messages and succeeded in the automatic verification of parametric specifications with complex railway network topologies.

We now describe these achievements in some more detail.

2.1 Explicit Durations

Explicit durations are a potential source of undecidability in the time dimension of the system specifications, and correspond to the (un-)decidability frontier between Timed Automata (TA), for which location reachability is decidable, and Linear Hybrid Automata (LHA), for which reachability happens to be undecidable. This frontier has been populated by variants of *Priced Timed Automata* (PTA)—a continuous-time variant of *weighted automata*—and *stopwatch automata* (SWA). Whereas PTA augment TA with continuous observer variables that may neither be reset nor be queried in guards and invariants, SWA do allow such resets and queries. As a consequence, some PTA variants admit decidability, while SWA are as expressive as LHA and thus not decidable.

A generalization of the translation of test formulae into PEA to a translation of formulae with explicit durations into PEA with *integrators* (a variant of SWA with possibly infinite data) was pursued. While doing so, we discovered that the basic structure of this translation can be isolated even in the setting of discrete time and formal languages, leading to the concepts of *availability automata* and corresponding *regular availability expressions* [22]. Availability automata are similar to weighted automata. However, the availability counters therein may also be queried and reset as for SWA. Availability automata provide an alternative language-theoretic characterization of request-response scenarios formalized by means of weighted automata in Henzinger's *quantitative languages* [7].

PTA, on the other hand, have only *observer variables* termed as *costs* in addition to the clocks of TA, and thus permit the analysis and optimization of phenomena (such as scheduling) beyond the scope of TA within certain decidable model classes lying at the frontier between TA and LHA. The PTA variants lying at this frontier correspond to explicit durations, owing to earlier AVACS work [16] on a decision procedure (that involves reduction to PTA having multiple positively valued cost variables) for model-checking TA against DC requirements having constraints on *positive* linear combinations of explicit durations with only *upper bound duration constraints*.

2.2 Structural Optimization

To avoid state space explosion, we pursued two different approaches. We *structurally optimized* the system specifications at the *design-level*, *prior to verification*. To this end, the operator of *layered composition* was lifted from the setting of (hierarchical) process graphs in [27] to that of TA extended with data [33]. Layered composition (intermediate between sequential and parallel composition) allows for the transformation of the system from a parallel representation into an *equivalent* layered and finally sequential one, provided certain conditions (concerning the *independence* of transitions wrt. variables accessed or the *precedence* of transitions enforced by timing in guards and actions) hold. The equivalence between the parallel and sequential representations induces an *a priori design-level partial order reduction* of the system's state space, with the preservation of *stutter-invariant* (i.e., next-free) temporal requirements. As an illustrative application, we revisited in [33] the UPPAAL case study of a collision avoidance protocol for an audio/video system by Bang & Olufsen [18]. We could show in [33] a possible a priori design level reduction by a factor of 300 of the number of discrete locations in the composite system representing the collision avoidance protocol of [18].

The concept of *verification architectures* (VA) was introduced in [12] (and elaborated in the PhD thesis [13]). VA have as parameters component processes with data constraints and timing requirements, and offer an abstract behavioral protocol view on complex real-time systems. In combination with COD, the component processes of VA formalize parametric version of CSP-OZ-DC and are represented as *unknown processes* satisfying certain local real-time requirements.

A VA splits system runs into several phases, formalized as unknown processes satisfying local real-time assumptions. Once a desired global requirement for a VA protocol is verified by proof rules of a dedicated dynamic logic, it is also guaranteed by all instances of that protocol satisfying local real-time assumptions. Thus, given a correct VA, we verify global safety requirements of concrete models by combining local analyses: for a concrete model—usually given as a complex specification in a combined language like COD—(1) the protocol structure needs to be an instantiation of that of the VA (entailing a purely syntactic check), and (2) the validity of local real-time assumptions for the corresponding components of the concrete specification needs to be model-checked by ARMC or SLAB. The VA approach thus provides: (1) a formal framework of design patterns for complex, combined real-time specifications, and (2) a decompositional approach that reduces global verification to local proof tasks. In contrast to our behavioral protocol-based VA patterns, previous approaches on formal design patterns either focus on handling standard design patterns that consider static analysis of code and structures in object-oriented languages, e.g., [28], or do not incorporate real-time aspects [37] or infinite data [17]. As a large-scale application of this VA approach, the Phase 1 case study of ETCS Emergency Messages has been revisited in [13].

2.3 Automating Verification

K. Dräger and B. Finkbeiner [10] introduced *subsequence invariants* that characterize the behavior of a concurrent system in terms of the occurrences of synchronization events. Unlike state invariants that refer to the state variables of the system, subsequence invariants are defined over auxiliary counter variables that reflect how often the event sequences from a given set have occurred so far. A subsequence invariant is a linear constraint over the possible counter values that is preserved when a given process is composed with additional processes. Subsequence invariants can therefore be computed individually for each process and then be used to reason about the full system. Subsequence invariants can be computed efficiently by a fixed point iteration. In his PhD thesis, K. Dräger extended the results of [10] to include more general synchronization patterns, and showed how to integrate the fixed point iteration for subsequence invariants with the SLAB refinement loop, making it possible to check the validity of a proposed invariant for an infinite-state system.

M. Heizmann, J. Hoenicke and A. Podelski [19] presented *refinement of trace abstractions* as a method to extend the scalability of automatic verification. A known bottleneck of automatic verification based on the classical CEGAR (counterexample-guided abstraction refinement) approaches is the intensive use of a theorem prover. This bottleneck was addressed using the following two techniques. First, a precise abstraction is obtained given several coarse abstractions. The crux of the technique is that only automata theoretic operations are used, but no theorem proving is needed. Second, a coarse abstraction is obtained given

an infeasibility proof of a spurious counterexample from an interpolating theorem prover. In this construction, a theorem prover is queried only to prove the inductivity of several selected transitions.

M. Heizmann, J. Hoenicke and A. Podelski [20] introduced a novel technique for the verification of a sequential system that consists of several procedures. While constructing an abstraction in a CEGAR based automatic verification, two contrasting requirements arise. On one hand, the refined abstraction should be precise; on the other hand, the refined abstraction should be small. Using the information obtained from interpolants of an unsatisfiability proof for an infeasible counterexample has shown to be useful tradeoff. In [20], a *nested interpolation* scheme was presented, where interpolants are not only tailored to a trace but also local to a procedure. This interpolation scheme allows one to represent the whole system by one abstraction, but analyzes the system in a modular way. Calls of procedures are summarized and reused in the further analysis. Furthermore, the interpolants obtained satisfy an inductiveness property, which allows one to combine this with the abstraction techniques from [19].

2.4 Complex Data

V. Sofronie-Stokkermans, together with S. Jacobs and C. Ihlemann, identified a large number of theories—in particular theories of data structures related to CSP-OZ-DC specifications—for which efficient reasoning procedures exist. For this, they used and extended their results on *local theory extensions* [26]. The locality property of a theory extension allows them to replace universally quantified clauses by a set of ground instances. This makes a reduction to a satisfiability test in the underlying theory possible.

Decidable fragments of theories of data structures were studied before, e.g., a fragment of the theory of arrays [3] and a theory of pointers [30]. Sofronie-Stokkermans et al. [24] presented and generalized these results in a locality framework. In [14] and the PhD thesis of C. Ihlemann, a more general fragment of the theory of pointers was considered, which turned out to be extremely useful for the verification of systems of trains with a complex track topology.

2.5 Verification Tools

To demonstrate applications of R1 techniques, we developed a tool chain. It takes a graphical UML model of a real-time system as input, applies property-preserving translations to COD specifications and via PEA into Transition Constraints Systems, following the R1 verification approach. The resulting transition system is then passed to the verification tools SLAB, H-PILoT, or ARMC.

The SLAB (Slicing Abstraction) model checker [6,11] was completely redesigned during Phase 2. In order to automatically verify the requirements of layered networks of PEA, it now incorporates a specialized abstraction refinement procedure. In contrast to the approach in Phase 1, where the model checker accepted a *product* of system processes, the new version accepts a *structured* description of the analyzed system represented in terms of parallel, sequential and layer composition. The tool initializes the refinement cycle with an

abstraction that accurately represents the control structure of the system but over-approximates its behavior with respect to complex data.

During the abstraction refinement cycle, SLAB inspects the current abstraction to find a counterexample. From the counterexample, the tool constructs a Craig interpolant, and uses it to split the abstraction locally, thus refuting the counterexample. In order to reduce the size of intermediate abstractions, SLAB applies two sets of rules. The first set consists of *slicing* rules [6] applied locally to some process in the abstraction, eliminating its inconsistent or irrelevant parts. The second set consists of *parallel reduction* rules tracing inconsistencies based on the synchronization between parallel processes. The two sets of rules mutually benefit from each other: slicing irrelevant parts in a process reduces its synchronization capabilities, and thus opens the way to apply the parallel reduction rules; and, vice versa, parallel reductions result in additional slicing steps.

The verification tool H-PILoT (Hierarchical Proving by Instantiation in Local Theory Extensions) [25] implements the method for hierarchical reasoning in extensions of logical theories and chains thereof. By this method, the satisfiability of constraints over specific theory extensions identified to be local are reduced to the satisfiability of constraints in a base theory for which a dedicated prover exists. Standard SMT solvers can then be used to check the satisfiability of the formulae of the base theory. With this approach, the invariant checking problem for local theory extensions becomes decidable. H-PILoT has been used to verify requirements of COD specifications with rich data types like arrays or pointer data structures.

We developed the graphical tool Syspect (System Specification Tool) [15] for modeling, specifying, and automatically verifying reactive systems with continuous real-time constraints and complex, possibly infinite data. It represents the R1 tool chain. For modeling these systems, a UML profile comprising component diagrams, protocol state machines, and class diagrams is used; for specifying the formal semantics of these models, the combination CSP-OZ-DC is employed; for verifying requirements of these specifications, translators are provided to the input formats of the model checkers ARMC and SLAB as well as the tool H-PILoT. By this means, Syspect bridges the gap between informal modeling techniques from software engineering and formal analysis of real-time systems.

2.6 Case Studies

We first revisited the case study of ETCS Emergency Messages between *two trains* considered in Phase 1 of R1. While it is still not possible to verify the global requirement of collision freedom entirely in a push-button fashion, we are now able to structure the proof into two formal parts: (1) A *verification architecture* with real-time assumptions on the "unknown processes" describes the abstract protocol of the case study. The global requirement of collision freedom is verified manually using the proof rules of a dedicated dynamic logic. (2) An *instantiation* of that verification architecture yields the full case study. The assumptions made for the "unknown processes" are verified fully automatically using the model checkers ARMC or SLAB. The verification architecture, the

instantiating model, and the automatic verification of the instantiation are realized with the Syspect tool. We then considered a variant of the ETCS case study without communication aspects and with a simplified control structure, but a complex track topology. In this case study, an *arbitrary number of trains* drive along a track network, specified by first-order formulae with data in doubly-linked lists. Invariant requirements like keeping a safe distance could be verified automatically. For this, the high-level COD specification with these data, modeled with Syspect, was at the semantic level of PEA automatically translated into the input format of the tool H-PILoT [14].

3 Conclusion

We presented an overview of the achievements of the AVACS project R1 "Beyond Timed Automata" during the period 2008–2011. While there have been some works outside of AVACS dealing with real-time systems augmented with data, these works do not cover the scope of the specification and verification techniques considered within R1.

- The works in [2,8,29] consider (variants of) timed automata augmented with (possibly unbounded) data structures (such as a push-down stack). However, these works deal predominantly with theoretical *decidability results* and do not present techniques amenable to automated verification.
- The works [1,9,34,36] present techniques for the automated reasoning of continuous real-time systems with data. The techniques in [9,1,36] are compositional and modular, but involve timed automata variants enriched with *finite data*. The techniques in [34] deal with complex (and possibly infinite) data, but involve equational reasoning based on rewriting logics, and are not compositional, and thus not amenable to modular verification.
- Furthermore, the timing requirements that can be handled in each of the above works are much more confined than the (explicit) duration requirements considered within R1.

In the coming third phase of AVACS, the project R1 will emphasize verification "beyond yes/no" by considering *parametric* systems and requirements. We will also pursue the paradigm "design meets verification" to find design styles and transformations for real-time systems that optimize their structure to ease their automatic verification.

Acknowledgements. This paper is a report of the work done in the project "Beyond Timed Automata" within the Transregional Collaborative Research Center AVACS during the period 2008–2011. I would like thank the other members of the project: I. Brückner, K. Dräger, J. Faber, B. Finkbeiner, M. Fränzle, M. Heizmann, J. Hoenicke, C. Ihlemann, S. Jacobs, A. Kupriyanov, R. Meyer, A. Podelski, V. Sofronie-Stokkermans, and M. Swaminathan.

References

1. Arbab, F., Baier, C., de Boer, F.S., Rutten, J.J.M.M.: Models and temporal logical specifications for timed component connectors. Soft. and Syst. Modeling 6(1), 59–82 (2007)
2. Bouajjani, A., Echahed, R., Robbana, R.: On the Automatic Verification of Systems with Continuous Variables and Unbounded Discrete Data Structures. In: Antsaklis, P.J., Kohn, W., Nerode, A., Sastry, S.S. (eds.) HS 1994. LNCS, vol. 999, pp. 64–85. Springer, Heidelberg (1995)
3. Bradley, A.R., Manna, Z., Sipma, H.B.: What's Decidable About Arrays? In: Emerson, E.A., Namjoshi, K.S. (eds.) VMCAI 2006. LNCS, vol. 3855, pp. 427–442. Springer, Heidelberg (2005)
4. Brückner, I.: Slicing Concurrent Real-Time System Specifications for Verification. In: Davies, J., Gibbons, J. (eds.) IFM 2007. LNCS, vol. 4591, pp. 54–74. Springer, Heidelberg (2007)
5. Brückner, I.: Slicing Integrated Formal Specifications for Verification. PhD thesis, Report Nr. 2/08, University of Oldenburg (March 2008)
6. Brückner, I., Dräger, K., Finkbeiner, B., Wehrheim, H.: Slicing abstractions. Fundamenta Informaticae 89(4), 369–392 (2008)
7. Chatterjee, K., Doyen, L., Henzinger, T.A.: Quantitative languages. ACM Trans. Comput. Log. 11(4), Article 23, 38 (2010)
8. Dang, Z.: Pushdown timed automata: a binary reachability characterization and safety verification. Theor. Comput. Sci. 302(1-3), 93–121 (2003)
9. Dong, J.S., Hao, P., Qin, S., Sun, J., Yi, W.: Timed automata patterns. IEEE Trans. Software Eng. 34(6), 844–859 (2008)
10. Dräger, K., Finkbeiner, B.: Subsequence Invariants. In: van Breugel, F., Chechik, M. (eds.) CONCUR 2008. LNCS, vol. 5201, pp. 172–186. Springer, Heidelberg (2008)
11. Dräger, K., Kupriyanov, A., Finkbeiner, B., Wehrheim, H.: SLAB: A Certifying Model Checker for Infinite-State Concurrent Systems. In: Esparza, J., Majumdar, R. (eds.) TACAS 2010. LNCS, vol. 6015, pp. 271–274. Springer, Heidelberg (2010)
12. Faber, J.: Verification Architectures: Compositional Reasoning for Real-Time Systems. In: Méry, D., Merz, S. (eds.) IFM 2010. LNCS, vol. 6396, pp. 136–151. Springer, Heidelberg (2010)
13. Faber, J.: Verification Architecture for Complex Real-Time Systems. PhD thesis, Report Nr. 03/11, University of Oldenburg (August 2011)
14. Faber, J., Ihlemann, C., Jacobs, S., Sofronie-Stokkermans, V.: Automatic Verification of Parametric Specifications with Complex Topologies. In: Méry, D., Merz, S. (eds.) IFM 2010. LNCS, vol. 6396, pp. 152–167. Springer, Heidelberg (2010)
15. Faber, J., Linker, S., Olderog, E.-R., Quesel, J.-D.: Syspect - modelling, specifying, and verifying real-time systems with rich data. International Journal of Software and Informatics 5(1-2), 117–137 (2011)
16. Fränzle, M., Hansen, M.R.: Deciding an Interval Logic with Accumulated Durations. In: Grumberg, O., Huth, M. (eds.) TACAS 2007. LNCS, vol. 4424, pp. 201–215. Springer, Heidelberg (2007)
17. Giese, H., Tichy, M., Burmester, S., Schäfer, W., Flake, S.: Towards the compositional verification of real-time UML designs. In: ESEC/FSE-11, pp. 38–47. ACM (2003)
18. Havelund, K., Skou, A., Larsen, K.G., Lund, K.: Formal modeling and analysis of an audio/video protocol: an industrial case study using UPPAAL. In: IEEE Real-Time Systems Symposium (RTSS), pp. 1–13. IEEE Computer Society (1997)

19. Heizmann, M., Hoenicke, J., Podelski, A.: Refinement of Trace Abstraction. In: Palsberg, J., Su, Z. (eds.) SAS 2009. LNCS, vol. 5673, pp. 69–85. Springer, Heidelberg (2009)
20. Heizmann, M., Hoenicke, J., Podelski, A.: Nested interpolants. In: Hermenegildo, M.V., Palsberg, J. (eds.) Principles of Programming Languages (POPL), pp. 471–482. Association for Computing Machinery. ACM (2010)
21. Hoenicke, J.: Combination of Processes, Data, and Time. PhD thesis, Report Nr. 9/2006, University of Oldenburg (July 2006)
22. Hoenicke, J., Meyer, R., Olderog, E.-R.: Kleene, Rabin, and Scott Are Available. In: Gastin, P., Laroussinie, F. (eds.) CONCUR 2010. LNCS, vol. 6269, pp. 462–477. Springer, Heidelberg (2010)
23. Hoenicke, J., Olderog, E.-R.: CSP-OZ-DC: A combination of specification techniques for processes, data and time. Nordic J. of Comput. 9(4), 301–334 (2002)
24. Ihlemann, C., Jacobs, S., Sofronie-Stokkermans, V.: On Local Reasoning in Verification. In: Ramakrishnan, C.R., Rehof, J. (eds.) TACAS 2008. LNCS, vol. 4963, pp. 265–281. Springer, Heidelberg (2008)
25. Ihlemann, C., Sofronie-Stokkermans, V.: System Description: H-PILoT. In: Schmidt, R.A. (ed.) CADE-22. LNCS (LNAI), vol. 5663, pp. 131–139. Springer, Heidelberg (2009)
26. Ihlemann, C., Sofronie-Stokkermans, V.: On Hierarchical Reasoning in Combinations of Theories. In: Giesl, J., Hähnle, R. (eds.) IJCAR 2010. LNCS (LNAI), vol. 6173, pp. 30–45. Springer, Heidelberg (2010)
27. Janssen, W.: Layered Design of Parallel Systems. PhD thesis, Univ. Twente (1994)
28. Knudsen, J., Ravn, A.P., Skou, A.: Design Verification Patterns. In: Jones, C.B., Liu, Z., Woodcock, J. (eds.) Formal Methods and Hybrid Real-Time Systems. LNCS, vol. 4700, pp. 399–413. Springer, Heidelberg (2007)
29. Lanotte, R., Maggiolo-Schettini, A., Troina, A.: Reachability results for timed automata with unbounded data structures. Acta Informatica 47, 279–311 (2010)
30. McPeak, S., Necula, G.C.: Data Structure Specifications via Local Equality Axioms. In: Etessami, K., Rajamani, S.K. (eds.) CAV 2005. LNCS, vol. 3576, pp. 476–490. Springer, Heidelberg (2005)
31. Meyer, R., Faber, J., Hoenicke, J., Rybalchenko, A.: Model checking duration calculus: A practical approach. Formal Aspects of Comput. 20(4-5), 481–505 (2008)
32. Olderog, E.-R.: Automatic verification of combined specifications. In: Pu, G., Stolz, V. (eds.) Proc. of the 1st Internat. Workshop on Harnessing Theories for Tool Support in Software, Macau. ENTCS, vol. 207, pp. 3–16 (2008)
33. Olderog, E.-R., Swaminathan, M.: Layered Composition for Timed Automata. In: Chatterjee, K., Henzinger, T.A. (eds.) FORMATS 2010. LNCS, vol. 6246, pp. 228–242. Springer, Heidelberg (2010)
34. Ölveczky, P.C., Thorvaldsen, S.: Formal modeling, performance estimation, and model checking of wireless sensor network algorithms in Real-Time Maude. Theor. Comput. Sci. 410, 254–280 (2009)
35. Podelski, A., Rybalchenko, A.: ARMC: The Logical Choice for Software Model Checking with Abstraction Refinement. In: Hanus, M. (ed.) PADL 2007. LNCS, vol. 4354, pp. 245–259. Springer, Heidelberg (2006)
36. Stöcker, J., Lang, F., Garavel, H.: Parallel Processes with Real-Time and Data: The ATLANTIF Intermediate Format. In: Leuschel, M., Wehrheim, H. (eds.) IFM 2009. LNCS, vol. 5423, pp. 88–102. Springer, Heidelberg (2009)
37. Taibi, T. (ed.): Design patterns formalization techniques. IGI Publishing (2007)