it 2/2007

SFB/TR 14 AVACS – Automatic Verification and Analysis of Complex Systems

Der Sonderforschungsbereich/Transregio 14 AVACS – Automatische Verifikation und Analyse komplexer Systeme

Bernd Becker, Andreas Podelski, Universität Freiburg,
Werner Damm, Martin Fränzle, Ernst-Rüdiger Olderog, Universität Oldenburg,
Reinhard Wilhelm, Universität des Saarlandes

Summary The Transregional Collaborative Research Center AVACS integrates the three sites Freiburg, Oldenburg, and Saarbrücken, and addresses the challenge of pushing the borderline for automatic verification and analysis of complex systems. A particular focus of the project is on models of complex transportation systems and their safety requirements. AVACS is organized in ten subprojects, each teaming researchers from all sites, and is funded by the German Science Foundation since January 1, 2004. This article surveys scope, organization, and research directions of AVACS, including pointers to key publications. ►►► **Zusammenfassung** Der SFB-TR AVACS mit den Standorten Oldenburg (Sprecherhochschule), Freiburg und Saarbrücken wird seit dem 1.1.2004 von der Deutschen Forschungsgemeinschaft gefördert. AVACS stellt sich der Herausforderung, Modelle komplexer verkehrstechnischer Systeme in Bezug auf die Einhaltung von Sicherheitseigenschaften zu analysieren, um so frühzeitig mögliche Entwurfsfehler aufzudecken. In insgesamt 10 Teilprojekten werden hierzu neue Verifikationsverfahren entwickelt, welche sowohl quantitativ wie auch qualitativ die Grenzen heutiger Technologien erweitern. Dieser Artikel gibt einen Überblick über Anwendungsdomäne, Organisation und Forschungsfelder von AVACS, einschließlich Verweisen auf weiterführende Literatur.

KEYWORDS J.7 [Computers in other Systems], F.3 [Logics and Meanings of Programs] formal methods, computer aided verification, temporal logic, hybrid systems, real-time systems, systems-of-systems

1 Introduction

Almost all technical artifacts we touch in every day life rely on "embedded systems", i. e., on embedded computing devices, which are not visible from the outside and are generally inaccessible by the user. "*Embedded Systems are everywhere, built into cars, roads, bridges and tunnels, into medical instruments and surgical robots, into homes, offices and factories, into airplanes and airports, into mobile phones and communication and virtual reality glasses, and even into our clothes*"[1]. AVACS focuses on applications of embedded systems in the transportation domain, such as automatic cruise control or car-to-car networking in automotive, aircraft collision avoidance protocols in avionics, or automatic train control applications such as required by the European ETCS/ERTMS standard, including the so-called moving-block-principle allowing increased traffic density while assuring collision avoidance. All such applications share key attributes, to be elaborated below: they are *complex* and *safety critical*.

[1] Quoted from the Artemis Strategic Research Agenda, see `http://www.artemis-office.org`

1.1 Sources of Complexity

Complexity of such embedded applications can be measured in multiple ways. One trend is to simply measure the size of the embed-

ded software. As examples, both the automotive and avionics domain exhibit exponential growth rates in embedded software, with the Audi A8 model in year 2005 containing some 90 MB of embedded code in flash memory. The Airbus A 380 realizes around 100 aircraft functions with the support of 400 MB of embedded software. Complexity of interconnects is a second indicator of overall system complexity, with e. g. the Airbus A 380 comprising of some 600 000 signal interfaces. The latter is an indicator of the increasing trend to realize new functions in *distributed* implementations, where embedded software running on multiple embedded control units interconnected through often hierarchically organized bus systems contribute to implementing *one* function. Examples include Automatic Cruise Control, scaled to the level of systems-of-systems in e. g. distributed protocols for collision avoidance, involving the interaction of trains and so-called radio-block-control centers using wireless communication in ETCS.

1.2 Verifying Safety Critical Systems

A high percentage of embedded systems in transportation is *safety critical*: failures of such systems can endanger human life. All industrial sectors in transportation thus either have established or are about to establish standardized guidelines for the development of such systems, such as DO-178 C for civil avionics, CENELEC 50128 or 50126 for rail applications, and the forthcoming ISO WD 26262 in the automotive domain. Such guidelines pose stringent process requirements, including the establishment of so-called safety cases, and increasingly call for the usage of *automation* for establishing what is called *safety requirements*, such as "trains may never collide". Such top-level safety requirements are refined to safety requirements of individual components, with safety cases providing a justification of how these jointly contribute to maintain the overall safety of, say, an aircraft with extreme high probability as prescribed by the pertinent standards, even when individual components fail. AVACS contributes to the development of safety critical systems a wealth of analysis methods, which allow us to *prove* – in the rigorous, mathematical sense –, that *models* of safety critical systems meet their *safety requirements*. This requires formal definitions of both the systems themselves, hence of mathematical models of safety critical systems, as well as the requirements. While techniques such as model-checking, addressing the verification of a particular class of models against requirements expressed in temporal logic, have been around for more than two decades (see e. g. the textbook by Clarke et al.[2]), the challenges addressed by AVACS stem from the *complexity of models* of safety critical applications: existing methods, while being able today to address well defined classes of applications, with even industry-strength offerings available on the market, cannot cope with multiple dimensions of complexity (such as having both large discrete state spaces and complex continuous control) simultaneously. In the next section, we outline the approach taken in AVACS to attack this complexity barrier.

[2] Edmund M. Clarke, Orna Grumberg, and Doran A. Peled. Model Checking. MIT Press, 1999

1.3 The AVACS Approach to the Verification of Complex Systems

Fig. 1 allows us to explain the AVACS research strategy and focus, as well as the overall approach.

First, Fig. 1 shows four levels of knowledge which are represented in AVACS. The top layer entails know-how on development processes for safety critical embedded systems in the transportation domain, which enables us to address the sources of complexity in real-life applications and their development processes, but which also allows us to exploit their specific characteristics for optimizing our methods.

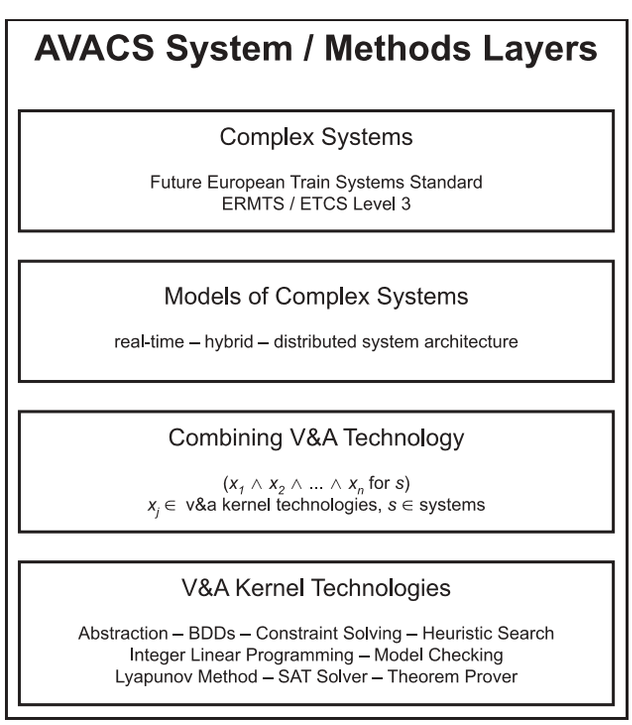

Figure 1 AVACS Knowledge Layers.

This background know-how only sets the stage for the foundational research carried out by AVACS, focussing roughly on three aspects:

- How can we build models of safety critical systems which are both sufficiently expressive as well as mathematically tractable to serve as a basis for formal verification?
- How can we extend core algorithms employed in state-of-the-art state space exploration techniques so as to best exploit the particularities and characteristics of safety critical systems?
- How can a focussed tight integration of such algorithms lead to "super-linear speedup" to solve verification challenges for particular classes of applications, characterized in their mathematical essence by particular classes of mathematical models?

The AVACS organizational structure reflects these three key research dimensions in multiple ways. First, we have divided AVACS into three *research areas* reflecting three classes of models, emphasizing particular foci of our research, notably *timeliness*, *interaction of control and plant*, and *systems-of-systems*. We follow this organizational structure in this article, in that the following three sections present each one such research area, by first explaining the characteristics of models analyzed in this research area, then discussing the application of developed methods with one example, and then highlighting research results achieved in subprojects of this research area. Secondly, we have in each of the ten subprojects made sure to mix competences in all four levels shown in Fig. 1, entailing that every subproject is carried out jointly by a team distributed over two, if not all three sites of AVACS. This organizational principle has proven to be extremely rewarding as it allowed us to tune both the core-algorithms as well as the customized integrations of such algorithms leading to a number of break-through results discussed in the subsequent sections.

2 Real-Time Systems

Real-time systems are systems that interact with their environment in such a way that for certain inputs the corresponding outputs have to occur within given time bounds. Many embedded systems, in particular those in safety critical applications, are of this type. As one case study in AVACS, we consider the handling of *emergency messages* in the European Train Control System (ETCS). In the ETCS level 3 trains communicate wirelessly with radio block centers (RBCs) (see Fig. 2). Controlling the traffic in certain areas, the RBCs grant movement authorities for trains up to a position closely behind the preceding train. In case of an emergency incident of the first train, the RBC has to ensure that this train and all successive trains will stop safely in order to avoid collisions. This safety property depends on the real-time behavior of the trains and the RBC: if the emergency message arrives at the subsequent trains in time, they will be able to stop before colliding with the preceding train.

The analysis and verification of real-time systems is based on computational models of such systems. These models describe different levels of abstraction in the development process. At the specification level they describe the required functionality and real-time behavior of the system as seen by the environment of the system. A state-of-the-art method is to take Timed Automata as a representation of systems at this level and then verify properties with model checkers like UPPAAL. The limitations of this approach are as follows: apart from clocks, it can cope only with finite data types; it suffers from a state space explosion when many clocks are present or many Timed Automata run in parallel.

The AVACS subprojects R1 and R3 present new solutions to these problems. Subproject R1 *Beyond Timed Automata* starts with systems specified in a high-level language CSP-OZ-DC where the three dimensions of concurrency, data, and time are represented by elements from Communicating Sequential Processes (CSP), Object-Z (OZ), and the Duration Calculus (DC). Real-time requirements are represented by certain formulas of the Duration Calculus. The challenge of R1 is to automatically verify real-time requirements of systems with both a continuous time domain (the real numbers) and infinite data types (e. g., arrays of integers). The approach is to transform the CSP-OZ-DC specification via an intermediate representation as a parallel composition of Phase Event Automata down to Transition Constraint Systems. These are the input for the abstraction refinement model checker ARMC [17]. This model checker calls decision procedures for

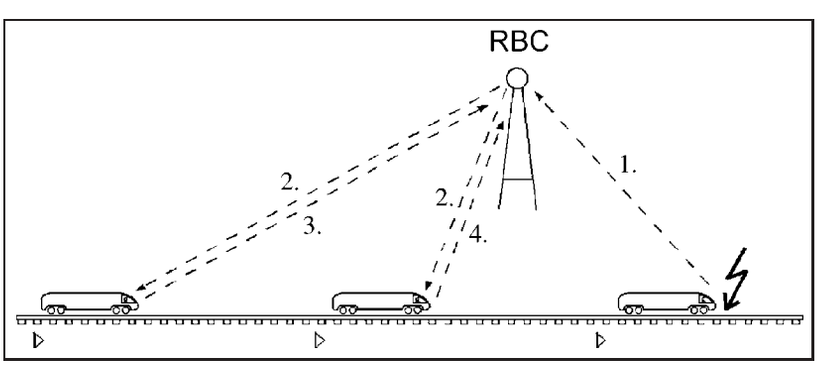

Figure 2 Case study emergency messages.

checking the validity of constraints over the data types of the specification [28]. Using this approach, it can, for example, be automatically verified that in the case study mentioned above the reaction time after an emergency message is small enough [20].

Addressing the complexity of real-time systems with many processes and many clocks is the topic of subproject R3 *Heuristic Search and Abstract Model Checking for Real-Time Systems*. In R3 real-time systems are represented as Timed Automata. The project extends the existing approach of *directed model checking* and uses abstraction techniques from program analysis ("predicate abstraction") and relaxation techniques from AI planning ("delete lists") in order to compute *distance estimates* that are used for guiding a heuristic search in the state space of a timed automaton [10; 18; 19].

At the implementation level, real-time requirements refer to processor execution time. To bridge the gap between the specification and implementation level, the required functionality is realized as a set of tasks. Networks of real-time tasks are studied in subproject R2 *Timing Analysis, Scheduling and Distribution of Real-Time Tasks*. This project develops sound techniques for determining upper bounds on execution times of tasks to be executed on processors with deep pipelines, large caches and speculation. These methods use abstract models of such complex architectures, which need to be either proved correct with respect to the concrete architecture or be formally derived from it. Much effort is therefore invested in the formal derivation and analysis of abstract processor models. R2 also develops optimal and heuristic methods for mapping task networks to distributed architectures involving communication subsystems. The applied scheduling methods have to guarantee that real-time constraints, end-to-end latencies, as well as periodicity constraints are met. The project combines abstract interpretation (to determine bounds on execution times), heuristic incremental scheduling methods (to support the design process), and constraint solving based on Integer Linear Programming (to solve scheduling problems). The link between the two real-time models used in R1 (and R3) and R2, respectively, is formalized by a semantics of task networks in terms of Timed Automata [8; 26; 29].

3 Hybrid Systems

Most embedded systems operate within or entail coupled networks of both discrete and continuous components. Safety assessment amounts to showing that the joint dynamics of the embedded system and its environment is well-behaved, e. g. that it may never reach an undesirable state or that it will converge to a desirable state, regardless of the actual disturbance. A typical example of such a hybrid discrete-continuous system from the transportation domain is depicted in Fig. 3. In this laboratory sample of an automated car platooning maneuver inspired by the California PATH project (http://www.path.berkeley.edu), the goal is to implement safe coordinated driving at low lateral distances, thus improving traffic density and drag coefficients, while at the same time maintaining the benefits of individual transport through offering the driver the safely computer-controlled options of entering and leaving platoons or of overtaking at safe time instants.

Within such systems, interactions between discrete computations and continuous systems occur at a number of different scales: Closed-loop control can, e. g., take the form of mode-switching control, where different continuous controllers are given control authority in alternation based on the decisions of a discrete supervisor. More globally, various task-specific controllers may be sequentially activated and deactivated to implement a multi-step mission (like overtaking) or exception handling (e. g., in case of an emergency). Finally, such missions have to be coordinated between traffic agents.

Within AVACS, four subprojects strive for providing automated verification technology tailored to the different interaction patterns arising, thereby exploiting the different balances between tightness of interaction and size of the subsystems.

Subproject H1 *Deduction and Automata Based Approaches* focuses on techniques for solving first-order constraints over the integers and reals and on their application to analysis of hybrid systems. First-order constraints can describe sets of states and transitions of hybrid systems. Consequently, efficient constraint solvers are enablers for a variety of analysis techniques, ranging from state-space exploration to discharging proof obligations in stability proofs. The project has enhanced different, complementary constraint solving procedures, ranging from Büchi-automata based approaches for constraint solving in $FOL(\mathbb{R}, \mathbb{Z}, +, \leq)$, where optimized

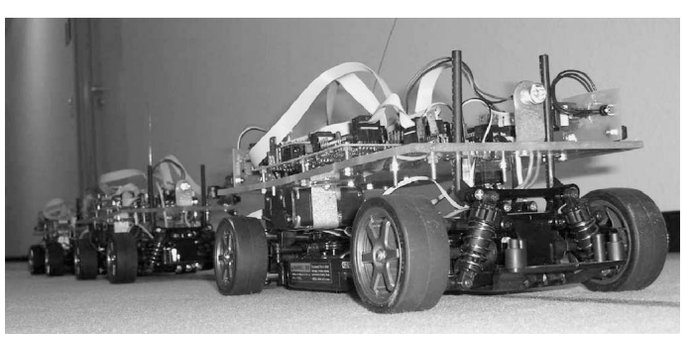

Figure 3 Coordinated driving in a car platoon.

algorithms and data structures have drastically lowered the memory demands [11], to – together with subproject H2 – mixed numeric-symbolic techniques primarily addressing robust constraints in undecidable fragments of arithmetic [16; 25]. *Robustness* here refers to correctness certificates which remain stable under small perturbation of the constants in the system description. Such robustness facilitates arguments based on notions of similarity and metric distance within formal verification [14] as well as the use of mixed symbolic-numeric constraint solving in automatic abstraction methods, thereby reducing the state-explosion problem in abstraction-based verification by computation of particularly concise abstractions [25].

Subproject H2, titled *Bounded Model Checking and Inductive Verification of Hybrid Systems*, deals with satisfiability checkers for many-sorted, quantifier-free logics and their use in optimization for bounded model checking and automatic inductive verification of hybrid systems. It extends modern SAT-solving technology, whose impressive performance gains have been instrumental to the growing industrial acceptance of formal verification technology as a debugging aid, to the hybrid discrete-continuous domain. The approach starts from the by now classical method of lazy theorem proving, yet optimizes this by exploiting the special formula structure arising in bounded model checking for aggressively pruning the search space, thus accelerating the underlying SAT solver [1; 15]. Recently, a fully SAT-based method tackling large (multiple thousands of real and Boolean variables) constraints in the undecidable domain of arithmetic involving transcendental functions has been developed [16].

Subproject H3 *Automatic Abstraction of Hybrid Controllers* investigates decomposition of large-scale hybrid systems, where cooperation principles separate global cooperation from local control, and employs compositional reasoning together with automatic abstraction into discrete-state models for verification. Decomposition thereby exploits design patterns as employed in coordinating autonomous transport vehicles so as to ease the burden in verifying cooperating hybrid systems [7]. Verification is performed by computing finite-state abstractions of non-linear hybrid systems using interval constraint solving, then embedding this into an abstraction refinement loop that is complete for the verification of arbitrary LTL properties on robust discrete-time hybrid systems [9]. Very large systems can be attacked by symbolic methods for representing large discrete state spaces combined with continuous regions by means of And-Inverter-Graphs (AIGs) with first-order constraints [6].

Complementing the aforementioned model-checking approaches, H4 *Automatic Verification of Hybrid System Stability* addresses the challenge of automatically constructing proofs of stability and convergence of hybrid systems. While Lyapunov functions and weight functions have been extensively used as a systematic approach for proving stability and convergence, automation of these procedures is currently lacking, in particular concerning selection of a suitable witness function. H4 attacks this problem with a variety of techniques, and joins them within an integration framework where heuristics select the appropriate technique. Linear matrix inequality methods combined with automatic partitioning of the state space [5] as well as interval based branch-and-relax algorithms [24] can construct Lyapunov-like functions of various polynomial degrees. Alternatively, [22] provides an automated method for showing liveness of a certain class of hybrid systems by means of automatic decomposition and construction of linear weight functions.

4 Systems of Systems

An overall verification methodology must address the global analysis of the interaction between different systems that communicate with each other in (classical or ad-hoc) networks. Typical instances in the traffic domain relate to the interaction between the different components of the on-board electronic subsystems of a vehicle (car, train, plane), or between the vehicles themselves, e. g., for collision avoidance or for traffic warnings or for forming *platoons*.

We illustrate the different aspects of systems of systems with the platoon example. A platoon is formed by cars on a highway; at one instance the cars will form a coordinated group, at another instance they will break out individually or as a group to merge with another group. Each car is an autonomous agent which by itself exhibits a complex behavior and has to possibly be modeled as a real-time system (since it underlies critical time bounds) or a hybrid system (since it is formed by digital controllers with discrete transitions as well as physical components showing a continuous evolution). Each car follows a protocol

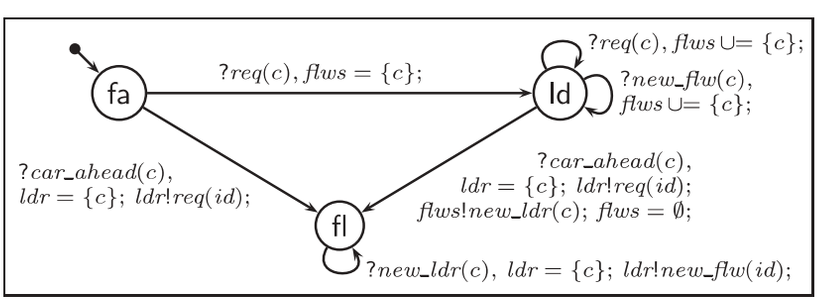

Figure 4 Merge of Car Platoons.

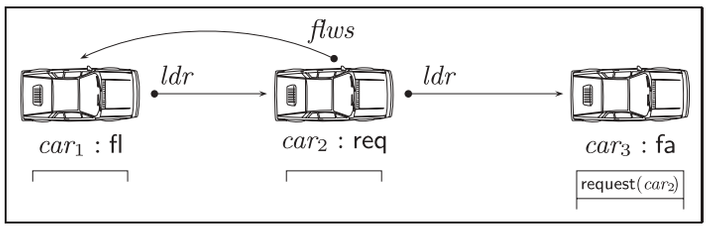

Figure 5 Snapshot for Merge of Car Platoons.

that, for example, regulates the sequence of interactions that leads to the *merger* of two platoons (see Fig. 4). A safety-critical property of the overall system is that within each merge, under all local state changes, the cars form a consistent configuration. The global verification task includes subsystems that can be specified only partially and that appear as black-boxes in the overall design. As a consequence, a *compositional* approach is required to automatically synthesize *assumptions* on subsystems and modularly check whether each subsystem provides the corresponding *guarantees*. At each step during the execution of a protocol (e. g., for merging two platoons), the *communication topology* may change, i. e., communication channels may be added, removed or just redirected. The global verification task thus includes the analysis of the different graph shapes that are characteristic at each point during the execution of a *dynamic communication* system. For certain aspects, the dependability of components and the reliability of communication channels cannot be guaranteed at 100% but only with certain probabilities. Consequently, the global verification task includes, as a third subtask, the *system dependability* taking into account stochastic component failure behaviors.

The three outlined subtasks of global verification are addressed in the three subprojects S1, S2, and S3, respectively. The models used in each of the subprojects reflect the particular complexity dimension of the respective subtask. These models allow one to classify design architectures involving black-box components, express the dynamic creation or removal of subsystems along with dynamic changes in the communication topology, or describe stochastic models of systems with complex failure dependencies.

Subproject S1 *Compositional Approaches to System Verification* investigates automatic verification methods for distributed systems that consist of multiple, statically connected components. For complex distributed systems, a compositional approach to verification is a necessity, because it is too expensive to simultaneously consider the implementation details of all components.

The new verification techniques of the subproject work with partial designs. In a partial design, a subset of the components is identified as "black boxes," for which only the requirements, but not the actual implementation, are considered. The subproject has enhanced the fundamental understanding of this verification problem by identifying the class of system architectures for which the problem is decidable. A uniform verification algorithm [12] provides an exact solution for all decidable cases. Additionally, an approximation technique [21] provides a fast way to produce proofs and counter examples (but is not guaranteed to be complete in all cases).

These verification techniques are supplemented with synthesis techniques for the automatic construction of component requirements [13]. In order to abstract from the implementation details of a component, it is necessary to find an appropriate collection of requirements that characterize what part of the component's behavior is relevant for the global properties under consideration. New techniques, developed in subproject S1, find these requirements through automatic abstraction refinement.

Subproject S2 *Dynamic Communication Structures* (DCS's) investigates systems of systems with dynamic communication. The key of automated verification for such systems is an effective abstraction method. We have extended existing abstraction techniques, in particular:

- symmetry reduction for networks with a parameterized number of participants,
- shape abstraction for imperative or object-oriented programs with dynamic memory management,
- counterexample-guided abstraction refinement,

in Oldenburg, Saarbrücken and Freiburg, respectively, and we are developing a tool chain with a feedback loop that implements each of the above abstraction techniques [3]. Using this tool chain, we have already been able to prove a number of safety-critical properties (both, safety and liveness) for the above-mentioned merge protocol in the platoon example. As a result that enables the automated abstraction refinement loop, we have shown that each new ('refined') abstraction can be computed by a theorem prover [23]. Finally, we have developed an appropriate notion of encapsulation that accounts, e. g., for the implementation of components in a communication system that passes references to data structures to their communication partners [27].

Verification of dependability properties needs to address stochastic phenomena, such as failure rates or message loss probabilities. As a consequence, quantitative guarantees can be given, as for instance *"The probability to hit a safety-critical system configuration within a mission time of 3 hours is at most 0.0001."*

Within subproject S3 *Formal Analysis of Availability Properties* we managed to verify such questions for real dependability problems originating from the ETCS domain. To reach these capabilities, we integrated very recent advances in stochastic model checking into a modelling environment with a stable industrial user group, STATEMATE. The algorithmic workhorse to validate (or refute) such properties is the first implementation of an algorithm [2] which computes the worst-case (or best-case) time bounded reachability probability in a *uniform continuous-time Markov decision process*. To generate such models starting from a STATEMATE specification comprises a nontrivial chain of transformation and compression steps, which are centered around a novel BDD-based implementation of a branching bisimulation compression algorithm [30]. The entire tool-chain is the result of intensive collaboration across the three sites [4]. The tool chain is currently a prototypical one. In the near future we aim at a deeper integration and improved performance. On the long-term horizon we see a high potential for addressing dependability issues on the level of partial designs of dynamically communicating subsystems.

5 Conclusion

We have described how the AVACS project addresses the different phenomena of complexity in the three project areas R, H and S. The results in the first phase of the AVACS project show a promising potential of automated methods for the verification and analysis of complex systems. However, for the time being the results more or less cover the mentioned phenomena in isolation. Later project phases will consider more and more enriched models, allowing us to deal with real-time systems and hybrid systems as components, probabilistic aspects of failure of communication of subsystems, and black-box components in a dynamic communication topology. All this paves the way for the AVACS vision, where an overall verification methodology developed in the project area S unfolds a proof tree for the global problem to a level where verification tasks can be discharged with the verification methods developed in the project area R and H.

References

[1] E. Ábrahám, B. Becker, F. Klaedke, and M. Steffen. Optimizing bounded model checking for linear hybrid systems. In: R. Cousot (Ed.), *Proc. of VMCAI'05 (Verification, Model Checking, and Abstraction)*, vol. 3385 of Lecture Notes in Computer Science, pp. 396–412. Springer, 2005.

[2] C. Baier, H. Hermanns, J.-P. Katoen, and B. Haverkort. Efficient computation of time-bounded reachability probabilities in uniform continuous-time Markov decision processes. *Theoretical Computer Science* 345(1):2–26, 2005

[3] J. Bauer, I. Schaefer, T. Toben, and B. Westphal. Specification and Verification of Dynamic Communication Systems. In: K. Goossens and L. Petrucci (Eds.), *Proc. of the 6th Int'l Conf. on Application of Concurrency to System Design (ACSD 2006)*, Turku, Finland. IEEE, June 2006.

[4] E. Böde, M. Herbstritt, H. Hermanns, S. Johr, T. Peikenkamp, R. Pulungan, R. Wimmer, and B. Becker. Compositional Perfomability Evaluation for Statemate. In: *3rd Int'l Conf. on Quantitative Evaluation of Systems (QEST 2006)*. IEEE Computer Society Press, 2006.

[5] H. Burchardt, J. Oehlerking, and O. Theel. The role of state-space partitioning in automated verification of affine hybrid system stability. In: *Proc. of the 3rd Int'l Conf. on Computing, Communications and Control Technologies*, vol. 1, pp. 187–192. International Institute of Informatics and Systemics, 2005.

[6] W. Damm, S. Disch, H. Hungar, J. Pang, F. Pigorsch, C. Scholl, U. Waldmann, and B. Wirtz. Automatic verification of hybrid systems with large discrete state space. In: S. Graf and W. Zhang (Eds.), *4th Int'l Symp. on Automated Technology for Verification and Analysis (ATVA)*, vol. 4218 of Lecture Notes in Computer Science, pp. 276–291. Springer, 2006.

[7] W. Damm, H. Hungar, and E.-R. Olderog. Verification of cooperating traffic agents. *Int'l Journal of Control* 79(5):395–421, 2006.

[8] W. Damm, A. Metzner, F. Eisenbrand, G. Shmonin, R. Wilhelm, and S. Winkel. Efficient algorithms for feasibility and optimality. In: *Proc. of the 12th IEEE Int'l Conf. on Embedded and Real-Time Computing Systems and Applications RTCSA '06*. 2006.

[9] W. Damm, G. Pinto, and S. Ratschan. Guaranteed termination in the verification of LTL properties of non-linear robust discrete time hybrid systems. In: D. A. Peled and Y.-K. Tsay (Eds.), *Proc. of the Third Int'l Symp. on Automated Technology for Verification and Analysis*, vol. 3707 of Lecture Notes in Computer Science, pp. 99–113. Springer, 2005.

[10] K. Dräger, B. Finkbeiner, and A. Podelski. Directed Model Checking with Distance-Preserving Abstractions. In: Antti Valmari (Ed.), *Model Checking Software, 13th Int'l SPIN Workshop*, vol. 3925 in Lecture Notes in Computer Science, pp. 19–34. Springer, 2006.

[11] J. Eisinger and F. Klaedtke. Don't care words with application to the automata-based approach for real addition. In: T. Ball and R.B. Jones (Eds.), *Proc. of the 18th Int'l Conf. on Computer Aided Verification (CAV'06)*, vol. 4144 of Lecture Notes in Computer Science, pp. 67–80. Springer, 2006.

[12] B. Finkbeiner and S. Schewe. Uniform distributed synthesis. In: *IEEE Sympo. on Logic in Computer Science*, pp. 321–330. June 2005.

[13] B. Finkbeiner, S. Schewe, and M. Brill. Automatic synthesis of assumptions for compositional model checking. In: *26th Int'l Conf. on Formal Methods for Networked and Distributed Systems (FORTE 2006)*, vol. 4229 of Lecture Notes in Computer Science, pp. 143–158. Springer, 2006.

[14] M. Fränzle and M. R. Hansen. A robust interpretation of duration calculus. In: *Proc. of the Int'l Colloquium on Theoretical Aspects of Computing (ICTAC 05)*, vol. 3722 of Lecture Notes

in Computer Science, pp. 257–271. Springer, 2005.

[15] M. Fränzle and C. Herde. Hysat: An efficient proof engine for bounded model checking of hybrid systems. *Formal Methods in System Design*. In print.

[16] M. Fränzle, C. Herde, S. Ratschan, T. Schubert, and T. Teige. Interval constraint solving using propositional SAT solving techniques. In: *CP 2006 Workshop on the Integration of SAT and CP Techniques*, pp. 81–95. Nantes, 2006. Under consideration for publication in JSAT.

[17] J. Hoenicke and P. Maier. Model-checking of specifications integrating processes, data and time. In: J.S. Fitzgerald, I.J. Hayes, and A. Tarlecki (Eds.), *FM 2005: Formal Methods*, vol. 3582 of Lecture Notes in Computer Science, pp. 465–480. Springer, 2005.

[18] J. Hoffmann, J.-G. Smaus, A. Rybalchenko, S. Kupferschmid, and A. Podelski. Using Predicate Abstraction to Generate Heuristic Functions in Uppaal. In: S. Edelkamp and A.R. Lomuscio (Eds.), *Post-Proc. of the 4th Workshop on Model Checking and Artificial Intelligence (MoChArt 2006)*. 2006.

[19] S. Kupferschmid, J. Hoffmann, H. Dierks, and G. Behrmann. Adapting an AI Planning Heuristic for Directed Model Checking. In: Antti Valmari (Ed.), *Model Checking Software, 13th Int'l SPIN Workshop*, vol. 3925 in Lecture Notes in Computer Science, pp. 35–52. Springer, 2006.

[20] R. Meyer, J. Faber, and A. Rybalchenko. Model Checking Duration Calculus: A Practical Approach. In: K. Barkaoui, A. Cavalcanti, and A. Cerone (Eds.), *Proc. Int'l Coll. on Theoret. Aspects of Computing (ICTAC)*, vol. 4281 of Lecture Notes in Computer Science, pp. 332–346. Springer, 2006.

[21] T. Nopper and C. Scholl. Approximate symbolic model checking for incomplete designs. In: A.J. Hu and A.K. Martin (Eds.), *Formal Methods in Computer-Aided Design*, vol. 3312 of Lecture Notes in Computer Science, pp. 290–305. Springer, 2004.

[22] A. Podelski and S. Wagner. Model checking of hybrid systems: From reachability towards stability. In: J. Hesphaha and A. Tiwari (Eds.), *Hybrid Systems: Computation and Control, HSCC'06*, vol. 3927 of Lecture Notes in Computer Science, pp. 507–521. Springer, 2006.

[23] A. Podelski and T. Wies. Boolean heaps. In: C. Hankin and I. Siveroni (Eds.), *Proc. of the 12th Int'l Static Analysis Symp. (SAS 2005)*, vol. 3672 of Lecture Notes in Computer Science, pp. 268–283. Springer, Sep. 2005.

[24] S. Ratschan and Z. She. Providing a basin of attraction to a target region by computation of Lyapunov-like functions. In: *IEEE Int'l Conf. on Computational Cybernetics*. 2006. To appear.

[25] S. Ratschan and Z. She. Safety verification of hybrid systems by constraint propagation based abstraction refinement. *ACM Journal in Embedded Computing Systems*. 2006. To appear.

[26] J. Reineke, B. Wachter, S. Thesing, R. Wilhelm, I. Polian, J. Eisinger, and B. Becker. A definition and classification of timing anomalies. In: F. Mueller (Ed.), *6th Int'l Workshop on Worst-Case Execution Time (WCET) Analysis*. Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, 2006. <http://drops.dagstuhl.de/opus/volltexte/2006/671>.

[27] N. Rinetzky, J. Bauer, T. Reps, M. Sagiv, and R. Wilhelm. A semantics for procedure local heaps and its abstractions. In: *POPL '05: Proc. of the 32nd ACM SIGPLAN-SIGACT sysposium on Principles of programming languages*, pp. 296–309. ACM Press, 2005.

[28] V. Sofronie-Stokkermans. Hierarchic reasoning in local theory extensions. In: R. Nieuwenhuis (Ed.), *Automated deduction – CADE-20: 20th Int'l Conf. on Automated Deduction*, vol. 3632 in *Lecture Notes in Artificial Intelligence*, pp. 219–234. Springer, 2006.

[29] S. Thesing. Modeling a system controller for timing analysis. In: *Embedded Software (EMSOFT) 2006*, pp. 292–300.

[30] R. Wimmer, M. Herbstritt, H. Hermanns, K. Strampp, and B. Becker. Sigref – A Symbolic Bisimulation Tool Box. In: S. Graf and W. Zhang (Eds.), *4th Int'l Symp. on Automated Technology for Verification and Analysis (ATVA)*, vol. 4218 of Lecture Notes in Computer Science, pp. 477–492. Springer, 2006.

1 Prof. Dr. Bernd Becker Professor for Computer Architecture at the University of Freiburg. Co-Speaker of SFB/TR 14 AVACS. Research interests include the verification and test of circuits and systems with a focus on efficient data structures and core algorithms, formal methods for correctness proofs in safety critical systems and defect based testing in nanoelectronics. Address: Albert-Ludwigs-Universität Freiburg, Institut für Informatik, Georges-Köhler-Allee 51, 79110 Freiburg, Germany, Tel.: +49-761-2038141, Fax: +49-761-2038142, E-Mail: becker@informatik.uni-freiburg.de

2 Prof. Dr. Werner Damm Professor for Safety-Critical Systems at the University of Oldenburg. Speaker of SFB/TR 14 AVACS. Scientific Director of the Interdisciplinary Research Center Safety Critical Systems at the University of Oldenburg. Member of the board of the OFFIS Institute for Information Technology, Oldenburg. Research Interests in specification, formal verification and analysis of safety critical embedded systems. Address: Carl von Ossietzky Universität Oldenburg, FK2, Dept. f. Informatik, Ammerländer Heerstr. 114–118, 26111 Oldenburg, Germany, Tel.: +49-441-9722500, Fax: +49-441-9722502, E-Mail: damm@informatik.uni-oldenburg.de

3 Prof. Dr. Martin Fränzle Professor for hybrid discrete-continuous systems at the University of Oldenburg. Research interests in the formal specification, the mathematical modeling, and the automated analysis of embedded systems, especially discrete-continuous ones. Coordinates the research area "Hybrid Systems" of the SFB/TR 14 AVACS.
Address: Carl von Ossietzky Universität Oldenburg, FK2, Dept. f. Informatik, Ammerländer Heerstr. 114–118, 26111 Oldenburg, Germany, Tel.: +49-441-9722566, Fax: +49-441-9722502, E-Mail: fraenzle@informatik.uni-oldenburg.de

4 Prof. Dr. Ernst-Rüdiger Olderog Since 1989 Professor for Theoretical Computer Science at the University of Oldenburg. Research into program verification and correct system design, in particular by combining specification and verification methods. Coordinates the research area "Real-Time Systems" of the SFB/TR 14 AVACS.
Address: Carl von Ossietzky Universität Oldenburg, FK2, Dept. f. Informatik, Ammerländer Heerstr. 114–118, 26111 Oldenburg, Germany, Tel.: +49-441-7982439, Fax: +49-441-7982965, E-Mail: olderog@informatik.uni-oldenburg.de

5 Prof. Dr. Andreas Podelski Since April 2006 Professor for Software Engineering at the University of Freiburg. Previously at Max Planck Institute for Computer Science in Saarbrücken, DIGITAL Paris Research Lab, University of Berkeley. PhD at University of Paris 7. Research in program analysis and verification. Most recent work on software model checking. Coordinates the research area "Systems of Systems" of the SFB/TR 14 AVACS.
Address: Albert-Ludwigs-Universität Freiburg, Institut für Informatik, Georges-Köhler-Allee 52, 79110 Freiburg, Germany, Tel.: +49-761-2038241, Fax: +49-761-2038242, E-Mail: podelski@informatik.uni-freiburg.de

6 Prof. Dr. Reinhard Wilhelm Since 1978 Professor for Informatics at Universität des Saarlandes. Since 1990 scientific director of the international Research and Conference Center for Informatics in Schloss Dagstuhl. Co-Speaker of SFB/TR 14 AVACS. Associate of the spin-off company AbsInt Angewandte Informatik GmbH. Working in abstract interpretation, in particular applied to the problem of timing analysis for hard real-time systems.
Address: Universität des Saarlandes, Fachrichtung Informatik, 66123 Saarbrücken, Germany, Tel.: +49-681-3023434, Fax: +49-681-3023065,
E-Mail: wilhelm@cs.uni-sb.de