1 Project info

1a) Project Title: Abstraction Refinement for Timed Systems

1b) Project Acronym: ARTS

1c) Principal Investigator: prof.dr. F.W. Vaandrager

2 Summary

2a) English Summary:

Characteristic for embedded systems is that they have to meet a multitude of quantitative constraints. These constraints involve the resources that a system may use (computation resources, power concumption, memory usage, communication bandwidth, costs, etc), assumptions about the environment in which it operates (arrival rates, continuous behavior), and requirements on the services that the system has to provide (timing constraints, QoS, fault tolerance, etc).

Model-Driven Development (MDD) is a new software development technique in which the primary software artifacts are models providing a collection of views. Existing MDD tools for real-time embedded systems are rather sophisticated in handling functional requirements but their treatment of quantitative constraints is still very limited. Hence MDD will not realize its full potential in the embedded systems area unless the ability to handle quantitative properties is drastically improved.

The goal of this project is to automatically construct abstractions of embedded system models (*e.g.* real-time UML) that contain sufficient information to analyze timing behavior of the original models but can still be analyzed using real-time model checkers.

To begin with, we want to develop a prototype tool based on the (publicly available) DBM library that has been developed by the UPPAAL designers. DBMs are efficient data structures to represent clock constraints in timed automata. They are used in UPPAAL as the core data structure to symbolically represent timing information. We will apply our prototype implementation to a number of simple verification problems to assess and fine tune the effectiveness of the abstraction techniques.

Next we intend to add the abstraction refinement technique to the full UPPAAL model checker. If feasible, we intend to combine our DBM based abstraction refinement technique with the Boolean abstraction refinement techniques used in BLAST and SLAM. The extensions will be evaluated experimentally by applying them to a series of existing benchmarks. Also, towards the end of the project, we will evaluate the ability of the tool to analyze real-time UML models.

1b) Nederlandse Samenvatting:

Het gedrag van computers wordt typisch gemodelleerd in termen van eindige automaten met toestanden en discrete overgangen daartussen. Een centraal probleem binnen de informatica is dat het aantal (bereikbare) toestanden van een realistisch systeem al snel uit de

hand loopt: wanneer een programma (of hardwarecomponent) bijvoorbeeld gebruikmaakt van honderd variabelen die ieder een (geheeltallige) waarde van 0 tot en met 9 mogen aannemen, dan is het aantal bereikbare toestanden in potentie googol (10^{100}), veel groter dan het aantal atomen in het universum. De meeste IT-applicaties hebben veel meer dan googol toestanden.

Recent is er door informatici spectaculaire vooruitgang geboekt ten aanzien van het rekenen met grote toestandsruimtes ("model checking"). Een basisidee is dat men de bereikbare toestanden niet een voor een afloopt maar rekent met (compact gerepresenteerde) verzamelingen van toestanden tegelijk. Ook is er veel onderzoek gedaan naar het automatisch construeren van abstracties. Zo kan de computer bijvoorbeeld detecteren dat de precieze waarde van een integervariabele er niet toe doet en dat het voldoende is om te weten of die positief, negatief dan wel nul is. Er zijn nu gereedschappen die abstractie en model-checking combineren om functionele eigenschappen van broncode te analyseren (bijvoorbeeld device drivers van Microsoft met meer dan 100.000 regels C code).

Steeds meer software is ingebouwd (embedded) in apparaten waarvan het de functionaliteit bepaalt (copieermachines, vliegtuigen, autos, treinen). Deze embedded systemen moeten voldoen aan een groot aantal kwantitatieve eisen (geheugengebruik, energieconsumptie, real-time, enz). Alhoewel speciaal voor dit doel ontwikkelde model checkers (zoals de real-time model checker UPPAAL) inmiddels zeer effectief zijn in het doorrekenen van kwantitatieve eigenschappen van abstracte modellen, is er momenteel nog geen goed gereedschap om automatisch goede abstracties te construeren.

Het doel van dit project is om zulk gereedschap te ontwikkelen: programmatuur waarmee het real-time gedrag van modellen en software voor embedded systemen geanalyseerd kan worden. Hiertoe willen we de techniek van *tegenvoorbeeld gedreven abstractie verfijning* ontwikkelen in de setting van real-time systemen. Deze techniek is zeer succesvol gebleken bij het analyseren van functionele eigenschappen van broncode, en is daarnaast met succes toegepast bij het analyseren van hybride (gemengd discreet/continue) systemen. Totnutoe heeft niemand tegenvoorbeeld gedreven abstractie verfijning ontwikkeld voor het analyseren van de bereikbare toestanden van real-time systemen. Dit is ook niet eenvoudig, maar wij denken dat het mogelijk is en verwachten dat het zeer effectief zal zijn. Allereerst willen we een prototype gereedschap ontwikkelen gebruikmakend van een bestaand pakket voor het manipuleren van Difference Bounded Matrices (DBMs), een datastructuur die een essentiële rol speelt in bestaande real-time model checkers. Na evaluatie en fine tuning van het prototype op basis van een aantal standaard benchmarks willen we, in samenwerking met collega's van de universiteit van Aalborg, de techniek toevoegen aan UPPAAL, een state-of-the-art model checker voor real-time systemen. We willen daarbij proberen om de door ons ontwikkelde abstractie verfijning voor real-time gedrag (DBMs) te combineren met de bestaande abstractie verfijning voor discrete toestandsruimtes. Vervolgens willen we de techniek evalueren middels toepassing op industriële real-time UML modellen.

3 Classification

Computer Science. The research is equally relevant for the NOAG-ICT theme Intelligent Systems (subarea Computational Logic, subsubarea model checking) and the NOAG-ICT theme Methods for Design and Construction (subareas model driven design and design environments).

4 Composition of the Research Team

The next table specifies the persons directly involved in this project, which will be carried out within the Informatics for Technical Applications (ITA) group at the Radboud University Nijmegen (see `http://www.ita.cs.ru.nl/`). Prof. F.W. Vaandrager will act as promotor of the prospective PhD student. The research will be carried out in close collaboration with dr A. Fehnker of the National ICT Institute of Australia in Sydney, and the team of prof. K.G. Larsen at the University of Aalborg in Denmark.

Name	Specialism	hrs/week
prof.dr. F.W. Vaandrager	computer aided verification, embedded systems	6
NN (Project PhD student)	formal methods	40
prof.dr. K.G. Larsen (Aalborg)	computer aided verification, embedded systems	PM
dr A. Fehnker (Sydney)	model checking timed and hybrid systems	PM
dr J. Hooman (ITA,ESI)	UML-based development of embedded systems	PM

The last time Vaandrager sent in a proposal in the Open Competititon was in 2001 (the FRAAI proposal with dr Hooman, which was granted). Vaandrager is co-applicant of the FOCUS proposal ARPA by dr Geuvers that was granted last year. However, his involvement in this project is very small.

5 Research School

The research in this project will be carried out in the context of research school IPA (Institute for Programming research and Algorithmics).

6 Description of the Proposed Research

6.1 Problem Statement

Embedded systems are highly specializable, often reactive, sub systems that provide, unnoticed by the user, information processing and control tasks to their embedding system. Embedded systems are omnipresent nowadays and make possible the creation of systems with a functionality that cannot be provided by human beings. Example application areas are consumer electronic products (*e.g.* CD players, microwave ovens), telecommunication

(*e.g.* mobile phones), medical systems (*e.g.* pacemakers), traffic control (*e.g.* intelligent traffic lights), driving and car control (*e.g.* ABS), airborne equipment (*e.g.* fly-by-wire), and plant control (*e.g.* packaging machines, wafer steppers). The term embedded system thus encompasses a broad class of systems, ranging from simple microcontrollers to large and complex multi-processor and distributed systems. The huge economic importance of embedded systems is undisputed.

Some characteristics of embedded systems are:

- Complex interaction with the environment. Embedded systems can only be designed and analyzed if one takes the behavior of their environment into account. Frequently this environment is highly nondeterministic and intrinsically continuous.
- A multitude of *quantitative* constraints. These constraints involve the resources that a system may use (computation resources, power consumption, memory usage, communication bandwidth, costs,..), assumptions about the environment in which it operates (arrival rates, hybrid behavior), and requirements on the services that the system has to provide.
- High dependability requirements. Besides functional constraints many other aspects play a role in the design of embedded systems: timeliness, fault tolerance, availability, security, safety, *etc.*.
- Design and manufacturing costs are very important.

This combination of factors makes the design of embedded systems in general a very complex task.

Models provide (mathematical) abstractions of a physical system that allow engineers to reason about that system by ignoring extraneous details while focusing on relevant ones. All forms of engineering rely on models to understand complex, real-world systems. Models may be developed as a precursor to implementing the physical system, or they may be derived from an existing system or a system in development as an aid to understanding its behavior. In the software engineering world, modeling has a rich tradition, dating back to the earliest days of programming. Boosted by the work of the Object Management Group (OMG) on the Unified Modeling Language (UML) and Model Driven Architecture (MDA), the role of models during application design, implementation, verification and validation has become much more important in recent years, and this is a very positive development. Model-Driven Development (MDD) is a system development technique in which the primary artifact is a model. Ideally, the technique allows engineers to (graphically) model the requirements, behavior and functionality of computer based systems. The model allows all the stakeholders to participate in the development process. The design is iteratively analyzed, validated, and tested throughout the development proces while automatically generated production quality code can be output in a variety of languages. Commercial tools such as Rational Rose, Rhapsody and visualSTATE have gained popularity primarily

because they support automatic code generation from abstract models (various variants of StateChart). These tools support verification of certain functional correctness properties (*e.g.* absence of deadlock, no dead code, *etc.*, and allow the user to manually generate test-cases during simulation of models. However, from the point of view of embedded systems, there is serious lack of support for predicting real-time behaviour, resource-consumption and performance in general of the generated code. Hence MDD will not realize its full potential in the embedded systems area unless the ability to handle quantitative properties is drastically improved.

Model checking is emerging as a practical tool for automated debugging of complex reactive systems such as embedded controllers. In model checking, specifications about the system are expressed as (temporal) logic formulas, and efficient symbolic algorithms are used to traverse the model defined by the system and check if the specification holds or not. Extremely large state-spaces can often be traversed in minutes. Model checkers were initially developed to reason about the logical correctness of discrete state systems (SMV, CADP, SPIN,μCRL,SAL), but have since been extended to deal with real-time (Uppaal), probabilistic systems (PRISM) and limited forms of hybrid systems (Hytech, PHAVER). There have been numerous successful applications of model checking technology to industrial problems (see *e.g.* [CW96, KCB02, MSV06, Vaa06] for pointers). In terms of impact, the main application area is again validation of hardware circuits by companies such as Intel. But also in the field of network and communication protocols model checking has become an indispensible tool. Model checking has been applied successfully to all kinds of scheduling problems in manufacturing, transportation and real-time scheduling. But even though model checkers for quantitative properties have become very powerful and are able to analyize manually constructed verification models, it is typically still very difficult to fully explore models (*e.g.* real-time UML) that are intended for code generation: when you try to do it these models just explode in your hands [HKO$^+$06]. To check large systems, abstraction is therefore a key paradigm: the purpose of an abstract model is to retain those features of a system that are necessary to verify the desired property, and to omit all unnecessary detail. Embedded systems developers don't have the time/resources to manually construct such abstract models. They want push-button technology that can be applied directly to their software or UML models. This brings us to the problem that will be addressed within the project: to automatically construct abstractions of embedded system models (*e.g.* real-time UML) that contain sufficient information to analyze timing behavior of the original models but can still be tackled using state-of-the-art real-time model checkers.

6.2 Approach

Recently a number of breakthroughs have been achieved and we see, for instance, that model-checking techniques are now being applied to validation of functional correctness properties of source-code (in particular C and JAVA), so-called software validation or runtime verification. Notable successes in this area have for instance been obtained by

the SLAM [BR01], BLAST [HJMS02], Bandera [CDH+00] and JAVA-Path-Finder [HP00] projects and tools. A basic technique used by these tools is *counterexample guided abstraction-refinement* [Kur94, CGJ+00, CGJ+03]. In abstraction refinement an initial very course abstraction of a program is computed automatically. In this abstraction, for instance, the only information about an integer variable that is preserved is whether it is zero, positive or negative. Or, alternatively, all valuations of program variables that cannot be distinguished by any of the Boolean guards that occur in the program are deemed equivalent. Next exhaustive state space search (model checking) is used to explore the abstract model. If in the abstract model no "bad" state can be reached then we know by construction that no bad state can be reached by the original program. In this case we have established correctness of the program, and we are done. In case a bad state can be reached in the abstract model then there are two possibilities:

1. either there is a corresponding execution of the original program that leads to a bad state; this means that we have found a bug in the original program,

2. or the bad execution in the abstract model does not correspond to any execution in the original program; in this case we can use the information about the failed correspondence to construct a refinement of the abstraction, that is, a new abstraction that is in between the old abstraction and the program, and we repeat the analysis.

SLAM and Blast have been succesfully applied within the domain of debugging of device drivers (programs with over 100,000 lines of C code). Counterexample guided abstraction refinement techniques has also been developed for the (computationally difficult) analysis of hybrid systems.

The aim of this project is to develop and implement the technique of counterexample guided abstraction for reachability analysis of real-time model checkers.

To begin with, we want to develop a prototype tool based on the (publicly available) DBM library that has been developed by the UPPAAL designers. DBMs are efficient data structures to represent clock constraints in timed automata . They are used in UPPAAL as the core data structure to symbolically represent timing information. The library features all the common operations such as up (delay, or future), down (past), general updates, different extrapolation functions, etc. on DBMs and federations. We will apply the prototype implementation to a number of simple verification problems to assess and fine tune the effectiveness of the abstraction techniques.

Next we intend to add the abstraction refinement technique to the full UPPAAL model checker. If feasible, we intend to combine the DBM based abstraction refinement technique with the Boolean abstraction refinement techniques used in BLAST and SLAM. The extensions will be evaluated experimentally by applying the new tool to a series of existing benchmarks. Also, towards the end of the project, we will evaluate the ability of the tool to analyze real-time UML models. Here we will build upon experience gained within the EU IST project OMEGA (in particular the timed automata semantics of real-time UML), and consider the MARS case studied within that project [HKO+06].

6.3 Importance of Proposed Research

Failure of embedded systems often may have serious consequences (loss of lives, huge financial losses), so correctness and reliability are of vital importance. As a result it is common for more than 75% of embedded software development costs to go into validation and verification. So there is a lot of potential for saving money.

This project will contribute towards bridging the gap between the powerful model checking technology for timed systems that has been developed over the last decade, and the needs of practitioners using Model-Driven Development tools. When successful, the results of the project will not only be implemented in the (widely used) academic toolset UPPAAL but also find their way to commercial tools such as Rational Rose, Rhapsody and visualSTATE.

6.4 Related Work

The research plans described in this proposal fit into the larger area of formal (mathematical) methods for validation and verification. We refer to [CW96, Vaa06] for general overviews.

Inspired by the success of model checking in hardware verification and protocol analysis [CGP99, Hol04], there has been increasing research on developing algorithms and tools for automated verification and analysis of quantitative properties of computer based systems. An important step towards supporting quantitative analysis of real-time aspects is provided by the modelling formalism of timed automata. The potential of timed automata for the modelling and analysis of real-time systems has been documented extensively in the literature. Since their introduction by Alur and Dill [AD94] in 1990, there has been an enormous progress in the field [Wan04b], and several verification tools for timed automata have been developed, which are now applied routinely to industrial-size case studies, *e.g.* UPPAAL [BDL04], KRONOS [DOTY96], and RED [Wan04a]. In our project we will build on the rich collection of algorithms and theoretical results that is available for timed automata.

In the world of program analysis, predicate abstraction has emerged to be a powerful and popular technique for extracting finite-state models from complex, potentially infinite state, discrete systems [CC77, GS97, DDP99, BR00]. verification. Counterexample guided abstraction-refinement [Kur94, CGJ$^+$00, CGJ$^+$03], which extends the basic predicate abstraction scheme, has dramatically increased the performance of software model checkers in recent years, see *e.g.* [BR01, HJMS02]. Counterexample guided abstraction has also been applied for the verification of hybrid systems [CFH$^+$03, ADI06]. Predicate abstraction techniques for timed systems have been studied earlier by [AIKY95] for a model of ω-automata with delays, and by [MRS02] for timed automata and a class of μ calculus formulas.

The success of the counterexample guided abstraction scheme crucially depends on the choice of the abstraction and the data structures that are used. To the best of our knowledge there are no results on applying counterexample guided abstraction refinement

in the setting of DBM based timed automata tools and reachability analysis. Since existing DBMs libraries have been optimized a lot, and form an extremely efficient way to represent clock constraints in timed automata, we expect to obtain a much faster implementation.

6.5 Embedding of the Research

The project will be carried out within the "Informatics for Technical Applications" (ITA) group at the Radboud University Nijmegen.

The research mission of ITA is to carry out fundamental research on formal methods and tools for the specification, design, analysis and testing of computer systems (with focus on embedded systems, distributed algorithms and protocols), and to demonstrate and assess the effectiveness of using these methods and tools in the industrial software development process. Main scientific achievements include the development of the hybrid and timed I/O automata modeling framework [LSV03, KLSV06] (together with the team of Nancy Lynch at MIT), the work on model based testing, contributions to the timed automata model checker UPPAAL (symmetry reduction [HBL+04], guiding and cost optimality [BFH+01b, BFH+01a, Feh02], parametrized analysis [HRSV02], and distributed model checking [BHV00]), and the application of (timed) model checking and theorem proving technology on dozens of compex, industrial problems (see *e.g.* [BGK+96, GVZ06, VdG06, HvdNV06]). In May 2004, an International Review Committee rated the research program of ITA as "Excellent" (in fact ITA was the only Dutch group that received the maximal score of 5), the quality of research was judged to be "Very Good", and the relevance "Excellent". The ITA team has been and is involved in a large number of international research projects and recently acted as coordinator of the EU IST project Advanced Methods for Timed Systems (AMETIST, see `http://ametist.cs.utwente.nl/`). The group has close ties with the Dutch Embedded Systems Institute (ESI) and has been / is involved in larger industrial collaboration projects with companies such as OCE Technologies, ASML, Philips Research, Bosch, Chess and Imtech. From the current ITA research projects, the one that is most closely related to this proposal is the NWO project FRAAI (Fault-tolerant Real-time Algorithms Analyzed Incrementally), which aims at establishing links between different abstraction layers for analysis of distributed algorithms.

The group has a longstanding and very productive collaboration with the group of prof. Kim Larsen in Aalborg centered around the real-time model checker UPPAAL, a collaboration that we would like to continue. In recent years, the model checker UPPAAL has advanced from an academic proof of concept to a tool that is being downloaded by thousands of researchers both in academia and in industry, and that is now being applied routinely to industrial verification problems. The collaboration with the UPPAAL team is a key element in the proposed project.

On the theoretical side we collaborate with Dr Ansgar Fehnker from NICTA, Australia. During his PhD research in Nijmegen, Dr Fehnker became an expert on timed model checking, and as a postdoc in the group of Prof. Ed Clarke at Carneggie Mellon University he was closely involved in the development of counterexample guided abstraction techniques for hybrid systems.

At the end of the project we expect to benefit from the experience of dr Hooman, (member of the ITA group and also affiliated with the Embedded Systems Institute), in the area of UML-based development of embedded systems (see *e.g.* [HKO+06]).

7 Project Planning

Below we provide an approximate planning of the research activities within the project.

Year 1:

(1 - 6) Detailled study of literature on real-time model checking and abstraction refinement.

(7 - 12) Development of DBM based abstraction refinment technique for real-time systems and prototype implementation using existing DBM package.

Year 2:

(13 - 18) Evaluation and fine tuning of the prototype using existing benchmarks from the literature.

(19 - 24) Implementation in UPPAAL and combination with other abstraction techniques, in particular abstraction refinement for discrete states as implemented in tools such as SLAM and Blast.

Year 3:

(25 - 30) Application of the approach to an industrial case study, for instance the UML models for the MARS system studies in the EU IST project OMEGA in [HKO+06].

(31 - 36) Modifying the theory where needed and adapting the proposed tool support.

Year 4:

(37 - 42) Applying the resulting framework some additional industrial cases.

(43 - 48) Completion of PhD thesis.

The training and education of the PhD student will be mostly fulfillled by attending basic courses and spring/fall days organised by the IPA research school. The PhD student will also visit at least one international summer school. Finally, a research visit of one month to the University of Aalborg (aimed at adding the developed abstraction techniques to UPPAAL) will also greatly contribute to the training of the PhD student.

8 Expected Use of Instrumentation

Not applicable.

9 Literature

Below the five most important publications of the applicants are listed.

[KLSV06] D.K. Kaynar, N.A. Lynch, R. Segala and F.W. Vaandrager. The Theory of Timed I/O Automata. Synthesis Lecture on Computer Science, Morgan & Claypool Publishers, 101 pages. 2006. ISBN 159829010X.

[LSV03] N.A. Lynch, R. Segala and F.W. Vaandrager. Hybrid I/O automata. *Information and Computation* 185(1):105–157, August 2003.

[Betal01] G. Behrmann, A. Fehnker, T. Hune K.G. Larsen, Paul Pettersson, J.M.T. Romijn and Frits Vaandrager. Minimum-cost reachability for priced timed automata. In M.D. Di Benedetto and A.L. Sangiovanni-Vincentelli, editors. *Proceedings HSCC'01*, Rome, Italy, March 2001. LNCS 2034, pages 147-161. Springer-Verlag, 2001.

[LV95] N.A. Lynch and F.W. Vaandrager. Forward and backward simulations, I: Untimed systems. *Information and Computation*, 121(2):214–233, September 1995.

[GV92] J.F. Groote and F.W. Vaandrager. Structured operational semantics and bisimulation as a congruence. *Information and Computation*, 100(2):202–260, October 1992.

References

[AD94] R. Alur and D.L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[ADI06] R. Alur, T. Dang, and F. Ivancic. Predicate abstraction for reachability analysis of hybrid systems. *ACM Transactions on Embedded Computing Systems*, 5(1):152–199, February 2006.

[AIKY95] R. Alur, A. Itai, R.P. Kurshan, and M. Yannakakis. Timing verification by successive approximation. *Inf. Comput.*, 118(1):142–157, 1995.

[BDL04] G. Behrmann, A. David, and K.G. Larsen. A tutorial on uppaal. In Marco Bernardo and Flavio Corradini, editors, *Formal Methods for the Design of Real-Time Systems, International School on Formal Methods for the Design of Computer, Communication and Software Systems, SFM-RT 2004, Bertinoro, Italy, September 13-18, 2004, Revised Lectures*, volume 3185 of *Lecture Notes in Computer Science*, pages 200–236. Springer, 2004.

[BFH+01a] G. Behrmann, A. Fehnker, T.S. Hune, K.G. Larsen, P. Pettersson, and J.M.T. Romijn. Efficient guiding towards cost-optimality in UPPAAL. In T. Margaria and W. Yi, editors, *Proceedings of the International Conference on Tools and Algorithms for the Construction and Analysis of Systems,* Genova, Italy, volume 2031 of *Lecture Notes in Computer Science*, pages 174–188. Springer-Verlag, April 2001.

[BFH+01b] G. Behrmann, A. Fehnker, T.S. Hune, K.G. Larsen, P. Pettersson, J.M.T. Romijn, and F.W. Vaandrager. Minimum-cost reachability for priced timed automata. In M.D. Di Benedetto and A.L. Sangiovanni-Vincentelli, editors, *Proceedings Fourth International Workshop on Hybrid Systems: Computation and Control (HSCC'01),* Rome, Italy, volume 2034 of *Lecture Notes in Computer Science*, pages 147–161. Springer-Verlag, March 2001.

[BGK+96] J. Bengtsson, W.O.D. Griffioen, K.J. Kristoffersen, K.G. Larsen, F. Larsson, P. Pettersson, and Wang Yi. Verification of an audio protocol with bus collision using UPPAAL. In R. Alur and T.A. Henzinger, editors, *Proceedings of the 8th International Conference on Computer Aided Verification,* New Brunswick, NJ, USA, volume 1102 of *Lecture Notes in Computer Science*, pages 244–256. Springer-Verlag, July/August 1996.

[BHV00] G. Behrmann, T.S. Hune, and F.W. Vaandrager. Distributed timed model checking — how the search order matters. In E.A. Emerson and A.P. Sistla, editors, *Proceedings of the 12th International Conference on Computer Aided Verification*, volume 1855 of *Lecture Notes in Computer Science*, pages 216–231. Springer-Verlag, 2000.

[BR00] T. Ball and S.K. Rajamani. Bebop: A symbolic model checker for boolean programs. In K. Havelund, J. Penix, and W. Visser, editors, *SPIN Model Checking and Software Verification, 7th International SPIN Workshop, Stanford, CA, USA, August 30 - September 1, 2000, Proceedings*, volume 1885 of *Lecture Notes in Computer Science*, pages 113–130. Springer, 2000.

[BR01] T. Ball and S.K. Rajamani. The SLAM toolkit. In G. Berry, H. Comon, and A. Finkel, editors, *Computer Aided Verification, 13th International Conference, CAV 2001, Paris, France, July 18-22, 2001, Proceedings*, volume 2102 of *Lecture Notes in Computer Science*, pages 260–264. Springer, 2001.

[CC77] P. Cousot and R. Cousot. Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *Proceedings of 4th ACM Symposium on Principles of programming Languages*, pages 238–252, 1977.

[CDH+00] J.C. Corbett, M.B. Dwyer, J. Hatcliff, S. Laubach, C.S. Pasareanu, R., and H. Zheng. Bandera: extracting finite-state models from Java source code. In *ICSE*, pages 439–448, 2000.

[CFH+03] E.M. Clarke, A. Fehnker, Z. Han, B.H. Krogh, J. Ouaknine, O. Stursberg, and M. Theobald. Abstraction and counterexample-guided refinement in model checking of hybrid systems. *Int. J. Found. Comput. Sci.*, 14(4):583–604, 2003.

[CGJ+00] E.M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement. In E.A. Emerson and A.P. Sistla, editors, *Computer Aided Verification, 12th International Conference, CAV 2000, Chicago, IL, USA, July 15-19, 2000, Proceedings*, volume 1855 of *Lecture Notes in Computer Science*, pages 154–169. Springer, 2000.

[CGJ+03] E.M. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Counterexample-guided abstraction refinement for symbolic model checking. *J. ACM*, 50(5):752–794, 2003.

[CGP99] E.M. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, Cambridge, Massachusetts, 1999.

[CW96] E.M. Clarke and J.M. Wing. Formal methods: State of the art and future directions. *ACM Comput. Surv.*, 28(4):626–643, 1996.

[DDP99] Satyaki Das, D.L. Dill, and S. Park. Experience with predicate abstraction. In N. Halbwachs and D. Peled, editors, *Computer Aided Verification, 11th International Conference, CAV '99, Trento, Italy, July 6-10, 1999, Proceedings*, volume 1633 of *Lecture Notes in Computer Science*, pages 160–171. Springer, 1999.

[DOTY96] C. Daws, A. Olivero, S. Tripakis, and S. Yovine. The tool KRONOS. In R. Alur, T.A. Henzinger, and E.D. Sontag, editors, *Hybrid Systems III*, volume 1066 of *Lecture Notes in Computer Science*, pages 208–219. Springer-Verlag, 1996.

[Feh02] A. Fehnker. *Citius, Vilius, Melius: Guiding and Cost-Optimality in Model Checking of Timed and Hybrid Systems*. PhD thesis, University of Nijmegen, April 2002.

[GS97] S. Graf and H. Saïdi. Construction of abstract state graphs with pvs. In O. Grumberg, editor, *Computer Aided Verification, 9th International Conference, CAV '97, Haifa, Israel, June 22-25, 1997, Proceedings*, volume 1254 of *Lecture Notes in Computer Science*, pages 72–83. Springer, 1997.

[GVZ06] B. Gebremichael, F.W. Vaandrager, and M. Zhang. Analysis of the zeroconf protocol using uppaal. In *Proceedings 6th Annual ACM Conference on Embedded Software (EMSOFT 2006),* Seoul, South Korea, October 22-25, 2006, 2006. To appear. Full version available as Technical Report ICIS-R06016, ICIS, Radboud University Nijmegen, 2006.

[HBL+04] M. Hendriks, G. Behrmann, K.G. Larsen, P. Niebert, and F.W. Vaandrager. Adding symmetry reduction to Uppaal. In *Proceedings First International Workshop on Formal Modeling and Analysis of Timed Systems (FORMATS 2003),* September 6-7 2003, Marseille, France, volume 2791 of *Lecture Notes in Computer Science*. Springer-Verlag, 2004.

[HJMS02] T.A. Henzinger, R. Jhala, R. Majumdar, and G. Sutre. Lazy abstraction. In *POPL*, pages 58–70, 2002.

[HKO+06] J. Hooman, H. Kugler, I. Ober, A. Votintseva, and Y. Yushtein. Supporting UML-based development of embedded systems by formal techniques. *Software and Systems Modelling*, 2006. To appear.

[Hol04] G.J. Holzmann. *The SPIN Model Checker: Primer and Reference Manual*. Addison Wesley, 2004.

[HP00] K. Havelund and T. Pressburger. Model checking java programs using java pathfinder. *STTT*, 2(4):366–381, 2000.

[HRSV02] T.S. Hune, J.M.T. Romijn, M.I.A. Stoelinga, and F.W. Vaandrager. Linear parametric model checking of timed automata. *Journal of Logic and Algebraic Programming*, 52-53:183–220, 2002.

[HvdNV06] M. Hendriks, N. J. M. van den Nieuwelaar, and F. W. Vaandrager. Model checker aided design of a controller for a wafer scanner. *Software Tools for Technology Transfer*, pages 1–15, 2006. Special Section on Quantitative Analysis of Real-time Embedded Systems.

[KCB02] D.R. Kuhn, R. Chandramouli, and R.W. Butler. Cost effective use of formal methods in verification and validation, 2002. Paper presented at Workshop on Foundations for Modeling and Simulation (M&S) Verification and Validation (V&V) in the 21st Century (Foundations 02), October 22-24, 2002, Johns Hopkins University Applied Physics Laboratory, Laurel, Maryland (USA).

[KLSV06] D.K. Kaynar, N.A. Lynch, R. Segala, and F.W. Vaandrager. *The Theory of Timed I/O Automata*. Morgan & Claypool Publishers, 2006. Synthesis Lecture on Computer Science, 101pp, ISBN 159829010X.

[Kur94] R.P. Kurshan. *Computer-Aided Verification of Coordinating Processes*. Princeton University Press, 1994.

[LSV03] N.A. Lynch, R. Segala, and F.W. Vaandrager. Hybrid I/O automata. *Information and Computation*, 185(1):105–157, 2003.

[MRS02] M.O. Möller, H. Rueß, and M. Sorea. Predicate abstraction for dense real-time system. *Electr. Notes Theor. Comput. Sci.*, 65(6), 2002.

[MSV06] T. Margaria, B. Schätz, and M. Verhoef. Formal methods going mainstream — cost, benefits and experiences, 2006. Report on the ForTIA Industry Day at FM 2005.

[Vaa06] F.W. Vaandrager. Does it pay off? model-based verification and validation of embedded systems! In F.A. Karelse, editor, *PROGRESS White papers 2006*. STW, the Netherlands, 2006. ISBN-10: 90-73461-00-6, ISBN-13: 978-90-73461-00-0.

[VdG06] F.W. Vaandrager and A.L. de Groot. Analysis of a biphase mark protocol with Uppaal and PVS. *Formal Aspects of Computing Journal*, pages Online first,

DOI 10.1007/s00165–006–0008–1, 2006. Also available as Technical Report NIII-R0445, NIII, Radboud University Nijmegen, 2004.

[Wan04a] F. Wang. Efficient verification of timed automata with bdd-like data structures. *STTT*, 6(1):77–97, 2004.

[Wan04b] F. Wang. Formal verification of timed systems: A survey and perspective. *Proceedings of the IEEE*, 92(8):1283–1305, 2004.

10 Requested Budget

We ask for the standard budget for a PhD student, Euro 172.371, a personal benchfee of Euro 5000 for this studenten, as well as Euro 4000 to enable a research visit of appr. one month to the University of Aalborg. The goal of this research visit will be to add counterexample guided abstraction refinement to the UPPAAL model checker, an essential element of this project. The total budget thus amounts to Euro 181.371.