Vrije Competitie NWO Exacte Wetenschappen
Project Proposal

1a) Project title : Efficient Multi-Core Model Checking

1b)Project acronym : $E = (mc)^2$

1c) Principal investigator : Dragan Bošnački
BMT, WH 3.101
Eindhoven University of Technology
PO Box 513
5600 MB Eindhoven
The Netherlands

1d) Renewed application: Yes.

This is an improved version of the proposal that was submitted in the round of January 15, 2008, file number 600.065.100.08N003. All the reviewers were quite positive above the proposal and the only negative remarks in the final evaluation were about the project planning and lack of more details in some parts of the text. Therefore in the new version of the proposal we made some substantial changes in the project planning and the description of the approach and methodology.

2a) Summary: Model checking is one of the most successful formal techniques for the verification of software and hardware systems. Developed in the beginning of the eighties, nowadays it is used by major companies, like Microsoft and Intel, to improve the quality of their products. Multi-core processors of affordable prices that emerged on the market in 2005 promise to bring low-cost parallel computers on our desktops which will denote a turning point in the area of parallel computing and computer science and engineering in general.

The aim of this project is to develop new algorithms for model checking (including probabilistic model checking) that can fully exploit the parallelism of the multi-core machines. The emphasis will be put on efficient algorithms for the so called liveness properties expressed in temporal logics.

One of the main application areas of model checking are concurrent systems. Since parallel machines will trigger a growing need of software that exploits

concurrency and parallelism, one can expect that the emergence of the multi-core technology will significantly increase the importance of model checking in industry. With the increasing complexity of the software the verification can obviously benefit from an efficient fast mulit-core model checking.

We plan to develop prototype implementations of the new algorithms in model checkers, like Spin and its extensions, as well as the probabilistic model checker MRMC. The prototype implementations will be validated on case studies including models of biological systems.

2b) Abstract for laymen in Dutch: Correctheid en prestatie van software en hardware zijn van wezenlijk belang. In het bijzonder geldt dit voor zogenaamde kritische systemen zoals medische aparatuur, vliegverkeerssystemen, kerncentrales, telefoonnetwerken, etc.

Model checking is een van de meest prominente formele technieken voor de verificatie van software- en hardware-systemen en wordt gebruikt door grote bedrijven, zoals Microsoft en Intel, om de kwaliteit van hun produkten te verbeteren.

Commerciële multi-core processoren verschenen op de markt in 2005 en tegenwoordig zijn de processoren van bijna alle nieuwe computers dual-core. In feite betekent multi-core dat men meerdere processoren (cores) in één processor heeft. Deze processoren kunnen parallel werken en hebben toegang tot hetzelfde geheugen. In de toekomst zullen de verbeteringen in de prestaties van de programma's afhangen van een effectief gebruik van dit parallelisme. Dit betekent een belangrijk keerpunt op het gebied van parallelle computing en informatica in algemeen.

Het doel van dit project is parallelle algoritmen te ontwikkelen voor model checking. Deze algoritmen moeten zoveel mogelijk het parallellisme van de nieuwe multi-core processoren gebruiken ter verbetering van de efficiëntie van model checking.

In het project ontwikkelde parallelle technieken voor model checking zullen in software tools geimplementeerd worden. De implementaties zullen geevalueerd worden in verschillende toepassingen, onder andere op biomedische systemen.

3) Classification:

- Discipline: Informatica (Computer Science)
- Computer Science (sub)disciplines: 1.1 Parallell systemen; 1.2 Gedistribueerde systemen; 3.4 Testmethoden (formele verificatie); 6.5 Formele methoden.
- Relevant themes from NOAG-i 2005-2010: De data-explosie; De genetwerkte wereld; Methoden voor ontwerpen en bouwen.

4) Composition of the research team:
(EUT = Eindhoven University of Technology, TU = Twente University of

Name	Affiliation	Expertise
Prof. Dr. M.G.J. van den Brand	EUT	CS, SE
Prof. Dr. P.A.J. Hilbers	EUT	CS, PA, BMBI
Prof. Dr. J.-P. Katoen	TU and AU	CS, VPS
Dr. D. Bošnački	EUT	CS, MC, BMBI
one Ph.D. student	EUT, TU, AU	CS
one postdoc	EUT	CS, MC

Technology, AU = RWTH Aachen University; CS = Computer Science, MC = Model Checking, BMBI = Biomodeling and Bioinformatics, PA = Parallel algorithms, SE = Software Engineering, VPS = Verification of Probabilistic Systems;)

Prof. van den Brand and Prof. Katoen will be promotors (advisers) of the Ph.D. student.

5) Research school: Instituut voor Programmatuur en Algoritmiek/Institute for Programming and Algorithmics (IPA)

6a) Description of the proposed research:

Introduction.

Model checking [24, 49, 21, 1] is a formal automated technique that is used to verify the correctness of parallel and distributed algorithms for hardware and software. The effectiveness of any verification method, including model checking, is limited by the complexity of the system that it is verified. Thanks to a continuous improvement of the algorithm designs, but also because of the increasing powers of the CPUs, model checking tools have been able to successfully cope with complexity and tackle a wide range of problems. Unfortunately it seems that we cannot rely anymore on the Moore's law [44] that predicts doubling of the efficiency of the hardware each 18 months. At the time of writing of this proposal the fastest PCs run at 3.8 GHz, while Moore's curve would have predicted 6.6 GHz.

The answer of the manufacturers to the Moore's law failure is focusing on further development of multi-core CPU systems that essentially contain multiple processors in one. Dual-core processors appeared on the market for the first time in 2001 but were quite expensive. Only in 2005 the first affordable price versions arrived [43]. Nowadays virtually all new PCs have dual-core processors and quadricore are also widely available. Manufacturers predict that in five years we can have processors with more than 80 cores. Machines with 256 cores (128 dual-core processors) are already available on the market.

The potential parallelism of the multi-core processors offers a unique opportunity to improve the efficiency of the model checking tools. Ironically enough,

model checking, that was mainly introduced for the verification of highly parallel systems, in the past has mostly relied on sequential algorithms.

Parallel model checking algorithms have been designed before the arrival of the multi-core systems (e.g., [51, 41, 4]). They were implemented on distributed memory systems - clusters of CPUs. However this did not have any major impact in practice - none of the widely used model checking tools has a cluster version that preserves its full range of capabilities.

The aim of this project is to develop efficient parallel algorithms for multi-core shared memory systems. We focus on explicit state model checking with applications to the verification of software and model checking for probabilistic systems. However, one could expect that the ideas behind the algorithms that we intend to develop can be reused in the symbolic model checking that has been employed successfully for hardware verification.

In the sequel we call the distributed (cluster-based) systems multi-CPU systems to distinguish them from multi-core systems.

Multi-core versus multi-CPU (clusters). Despite the many similarities, there exists several important differences between multi-CPU and multi-core systems. For instance, multi-core systems provide all CPUs with access to fast shared memory, making inter-CPU data transfer much more efficient compared to cluster computers. Distributed systems communicate via message passing which can introduce significant delays which is one of the main sources of inefficiency. On the other hand in the multi-core systems we can use queues that reside in shared memory. Those incur much less overhead, but the drawback is that one has to take into account the coherency of the data. Sharing the memory requires efficient mutually exclusive access to the data. This is not such an important issue in multi-CPUs where each CPU has its own memory. (For a comprehensive overview on these differences see [3].) Thus, multi-core systems pose new research questions and require new approaches to the parallel model checking algorithms.

Main research problems/questions and expected results.

Speed and scalability. With the modern 64-bit CPUs there is no practical limit to the amount of shared memory that can be addressed. Thus, using parallel systems, like clusters, to increase the memory of the system is not a primary goal anymore. Instead, the objective shifts toward runtime reductions. Multi-core model checking aims at exploiting the high-throughput of the multi-core CPUs that increases with the number of cores. Therefore, two natural principal common requirements for our algorithms will be speed and scalability, expressed as follows:

- A successful multi-core algorithm must be faster than its sequential counterparts and in most cases than its distributed analogues.
- The efficiency (both in memory and runtime) of the algorithm should steadily improve as the number of cores is increased.

Compatibility. One of the major issues in model checking is the so called state space explosion (e.g. [54]). There exist numerous state space reduction

techniques to combat this problem, like *partial order reduction (POR)* [52, 53, 25, 48], *reduction based on symmetry* [32, 22], as well as approximative verification techniques like *bitstate hashing* [26] and *hash-compact* [55]. Besides the memory requirements these state space reduction strategies also reduce the run time of the algorithms. Since they are crucial for successfully applying model checking in practice we will strive toward making the multi-core model checking algorithms compatible with them.

Unfortunately, reconciling efficient model checking algorithms with reduction techniques is quite often far from trivial even in the sequential case (e.g., [12]). The compatibility constraint will very likely dictate new types of algorithms compared to the existing sequential versions. Considering the above mentioned difficulties with the various reduction techniques, we will define as our research goal only compatibility of the multi-core algorithms with partial order reduction. The compatibility with the other techniques remains as optional research problem which will be tackled depending on the overall project progress.

Load balancing and safety properties. In general, a parallel program can be performed most efficiently if its subtasks can be distributed as evenly as possible between the computing nodes. In parallel model checking algorithms such a workload balance is achieved with state space partitioning functions that map states to processing entities (CPUs or cores). Versions of these functions used in multi-CPU model checking algorithms do not perform that good in a multi-core setting, causing unnecessary communication overhead [3, 27, 28]. In [29, 27, 28] an algorithm for safety properties (roughly: properties that can be disproved with a finite counterexample) was reported that showed quite nice scalability. However, this algorithm is by no means optimal and it is an interesting research question how to further improve the partitioning function.

Liveness properties. Probably the most important problem at the moment in multi-core model checking is to design an efficient - linear in the size of the state space - algorithm for liveness properties (properties that can be disproved with infinite counterexamples). In this project we aim at developing an efficient algorithm for model checking with Linear Temporal Logic (LTL) [21]. Main sequential liveness algorithms are based on depth-first search (DFS) which unfortunately is inherently sequential [50]. Recently published algorithms that are adaptations of their multi-CPU counterparts are in the worst case cubic in the size of the state space (number of states) [3, 4, 5, 15, 17]. Besides that, it is not clear how those algorithms will scale with the number of cores.

Probabilistic model checking. An interesting branch of model checking is the model checking of probabilistic systems(cf. [1, 2, 38]). Unlike in classical model checking in probabilistic model checking the properties are not required to hold for all cases, but instead with certain probability. Probabilistic model checking is even more vulnerable than its classical counterpart to the state space explosion [2, 38, 36]. Thus it can benefit even more from a prospective speed up by the multi-core algorithms.

There are several publications on probabilistic systems in distributed environment [45, 19, 7]. They are mostly limited to analysis of Markov chains which form the basis for probabilistic model checking. To the best of our knowledge

there are no multi-core versions of model checking algorithms for timed and probabilistic models, thus, it would be interesting to explore the possibilities in that direction.

To summarize, we will focus on the following research topics:

- efficient load-balancing and state space partitioning functions;
- efficient multi-core algorithms for the verification of safety and liveness properties;
- multi-core algorithms for probabilistic/stochastic systems;
- speed, scalability and compatibility with state space reduction techniques of the multi-core algorithms.

Prototype implementations of the algorithms will be developed and tested on realistic case studies. The results of the project will be published in highly recognized conferences and journals.

Research approach and methodology.

Load balancing and safety properties. For the verification of safety properties we will use as a starting point the algorithm reported in [27, 28, 29]. Unlike the vast majority of parallel algorithms that use hash functions in order to achieve an even load balance, this multi-core algorithm uses a *sliced-stack* method to partition the state space. In this method the cores communicate via bounded queues. The search is essentially depth-first, only each search is limited by a maximal depth L. When a core c_1 generates a state s beyond the depth limit it hands s off to another core c_2, by putting it into the queue of c_2. (Each core has its own queue that it can exclusively read.) Core c_2 fetches s from its queue and starts a new DFS using s as a root. Again, if a state which is in a distance from s that it is greater than the depth limit L is generated, it is handed off to another core, and so on. To avoid deadlocks, if no free core is available the hand-off does not happen and the DFS continues as usual.

In principle this algorithm scales well for N cores which has been confirmed also by the experiments reported in [27]. It would be interesting however to investigate how those two approaches (hash-function and sliced stack) compare exactly to one another and to get further insights into their strengths and weaknesses. It is worthwhile to check if this could be used further to improve the safety algorithms.

Liveness properties and compatibility. In the most efficient sequential algorithms for LTL model checking, like Nested Depth-First Search [23] and the Strongly Connected Component based ones(e.g. [54]), DFS plays an essential role. Since DFS is inherently sequential [50] a simple adaptation for parallel systems of these algorithms seems impossible. Thus, a parallel algorithm for LTL that has a linear complexity in the number of states is still an open problem.

Also, because of compatibility with reduction techniques algorithms quite different from the sequential ones will be needed. An indicative example is

partial order reduction. The problem is that the most efficient algorithms for POR are based on DFS [52, 25, 48]. Some important features, like the so-called cycle proviso [52] are tailored with this assumption in mind.

One possibility could be to design algorithms which would use alternative search orders. Obviously, a new search order would require new versions of the cycle proviso and the other DFS dependent components of the algorithm. In that direction we intend to build up on [10, 11, 9], where cycle provisos for breadth-first search and general exploration algorithms, respectively, were presented.

Trying to reconcile the new algorithms with symmetry reduction and the approximative techniques (bit-state and hash-compact) also will pose non-trivial questions. For instance, such kind of obstacles are not easy to resolve also in the sequential case [8]. Regarding the symmetry reduction issues in the parallel context we intend to expand on the results in [8]. We emphasize again that primarily we will be trying to solve only the problem with compatibility with POR, while the other reduction techniques remain optional.

Probabilistic model checking. Standard and probabilistic multi-core model checking can share several common aspects. For instance schemes to parallelize algorithms for binary decision diagrams (BDDs) can be applied in both cases. Thus, we will focus on parallelizing BDD schemes.

Another direction will be to develop parallel versions of numerical algorithms for solving systems linear equations. Many of the probabilistic algorithms are based on matrix/vector multiplications, which are inherently parallelizable.

Experimental evaluation. It is of utmost importance to evaluate model checking algorithms on various case studies. Therefore most of the new algorithms will be implemented. The implementations will be compared with analogous sequential and multi-CPU algorithms. To this end we will use implementations of these algorithms for computer clusters (e.g. [3]) and the BEEM bench-mark set [47].

Besides that, since this project is focused on software model checking most of our cases will be on verification of parallel software. We intend to analyze realistic multi-threaded systems. (The experimental results from [27] seem to suggest that multi-core model checking shows its advantages on such examples with huge state spaces.) This will be done preferably in collaboration with industry.

Considering the expertise and background of the project team, we will also deal with case studies inspired by biological systems. In the last years several successful applications of model checking in biology were reported(e.g. [20, 42, 13]). Since biological systems tend to be even more complex than their hardware and software counterparts, they will certainly benefit from the enhancements by the multi-core approach. In particular, our experience [14] shows that probabilistic model checking can be indispensable for analyzing biological processes.

The results of all case studies will be compared with the analysis of those systems with analogous sequential algorithms and algorithms on clusters.

The results on standard model checking will be implemented mostly on top of the model checker Spin [26]. Spin is one of the most widely used model

checkers. The principal investigator has significant experience with the tool and has been contributing various extensions and improvements of it, some of which are included in the official distribution.

For implementations of the probabilistic model checking algorithms we will use the probabilistic model checker MRMC [35]. The latter has been developed in the group of J.-P. Katoen and it is one of the best tools in its branch [34].

Scientific interest and urgency.

A vast majority of the 500 most powerful supercomputers are characterized as clusters (www.top500.org) [43]. Unfortunately this fact has not had much impact on the popularity of parallel computing. For decades parallel computers have been used almost exclusively in scientific areas. With the emergence of the multi-core technology they are ready to move to everyday life. Low cost parallel computers will soon entail simple to use parallel programming environments [43, 46]. The programs on which we base our businesses and research will possibly have to be adjusted rewritten from scratch, often significantly.

It is important that computer science and model-checking in particular will be ready for that. If we want to stay relevant with industry, we have to keep the pace with the new technologies.

Relation of the proposed research with similar research that is done elsewhere.

There exists extensive literature that deals with parallel algorithms for model checking, but it is almost exclusively based on the distributed memory cluster based model (e.g.[3, 4, 5, 6, 15, 16, 17, 18, 33, 37, 40, 41, 51]).

To the best of our knowledge, the papers [27, 28] co-authored by the principal investigator of this project ware the first publications that dealt with multi-core algorithms and model checking. There the safety algorithm with sliced-stack partition was presented. Also an algorithm for LTL properties for dual-core processors was given. Although in many cases it showed a significant runtime improvements compared to the sequential implementation, the main drawback of this algorithm is that it is difficult to see how it can be scaled for more than two cores.

Probably the first results on shared memory model checking were presented in [31, 30]. The paper [30] deals with CTL^* model checking and it is based on hesitant alternating automata. Because of that it is quite difficult to make a meaningful comparison to the cycle detection algorithms used for LTL. Besides that, the verification runs that are giving stop at the generation of the first counter-example.

A recent paper [3] presented an LTL model checking algorithm. However this algorithm is in the worst case cubic in the number of states and because of that its practical application could be quite limited.

Fitting of the research in the groups of the investigators.

Most of the work within the project will be done in the groups of Mark van den Brand in Eindhoven and Joost-Pieter Katoen in Twente and Aachen.

Joost-Pieter Katoen and his group of Software Modeling and Verification have internationally recognized competence in the field of formal specification and verification of probabilistic systems.

The Software Engineering and Technology group of Mark van den Brand is focused on maintaining the consistency between models and code. The experience in software development of the group will provide the knowledge that is needed for the implementations of the multi-core algorithms and extraction of models from the source code of realistic case studies.

Peter Hilbers is an expert in parallel algorithms and has an outstanding record in applications of parallel computing. The Biomodeling and bioinformatics group that he leads at the Biomedical Engineering Department in Eindhoven can provide expertise and interesting case studies of biological systems.

Dragan Bošnački among others has coauthored pioneering papers on multi-core model checking. Besides the theoretical work he has a broad experience in tool implementations and applications of model checking. He also has expertise in modeling of biological systems.

We envisage close collaboration with the groups of Jos Baeten and Jan Friso Groote in Eindhoven, which have experts in model checking, and the group of Jaco van de Pol at Twente which has built up significant experience in distributed algorithms for model checking.

On international plan we expect collaboration with the groups of Gerard Holzmann at NASA JPL, Marta Kwiatkowska at Oxford, and Luboš Brim in Brno.

6b) Application perspective: Computer industry has been using software with parallel features on sequential machines for decades. Operating systems like Windows and Unix/Linux are multi-threaded and languages like Java support multi-threaded programming. With the new low-cost parallel machines and the parallel programming models like OpenMP [46], parallel technology can finally fulfill its promises by offering full exploitation of the gains by multi-core throughput.

Therefore, it is a plausible assumption that the importance of parallel systems will significantly increase. Parallel programs are notoriously more difficult to reason about than the sequential ones. As a result, one can expect that the need for verification of commercial programs will increase in the coming years. Together with program analysis, model checking is the most applied formal verification technique in industry. As model checking mostly targets parallel systems, industry and model checking can have mutual benefit from these trends. Since the model checker Spin is one of the most popular model checkers one can expect that the project results will also have an impact on the verification of realistic systems.

7) Project Planning:

In general, we expect that both the postdoc and the Ph.D. student will work on the theoretical parts as well as on the implementations and experimental

evaluations. Table 1 contains a tentative schedule of their tasks. The roles of the senior members of the research team will be distributed according to their expertises as described above.

During the project the Ph.D. student will follow the regular courses that are organized by the IPA research school. Both the Ph.D. student and the postdoc researcher will attend the events that are organized by IPA (e.g. IPA fall and spring days) and will attend at least one summer school.

The Ph.D. student will be hosted half of the time in the group of J.-P. Katoen regarding the needed expertise in probabilistic model checking. Also the Ph.D. student will stay for 3 months research visit in one of the leading groups in parallel and/or probabilistic model checking (e.g. the groups of Luboš Brim in Brno or Marta Kwiatkowska at Oxford). The postdoc will be working in the group of M. van den Brand.

The fact that we have done already some substantial work on multi-core model checking with encouraging results which are published in journals and conferences and were well received by the community, gives us confidence that the project has high chance of success. Probably the greatest challenge will be finding an efficient livenenss algorithm. Therefore we intend to tackle this topic immediately from the beginning of the project. In case some additional time for this task is needed it should be possible to allocate it without jeopardizing the other work packages. We estimate that the rest of the project subjects are relatively low-risk which should guarantee a successful project and deliverables (e.g., a good quality Ph.D. thesis by the Ph.D. student).

8) Expected use of instrumentation:

- desktop computers with multi-core processors (purchased specially for the project);
- computer clusters (already available in Eindhoven).

9) Literature:

References

[1] C. Baier, J-P. Katoen, *Principles of Model Checking*, MIT Press, 950 pp, 2008.

[2] C. Baier, B. R. Haverkort, H. Hermanns, J.-P. Katoen, *Model-checking Algorithms for Continuous-Time Markov Chains*, IEEE Transactions on Software Engineering, **29**(6), 2003.

[3] J. Barnat, L. Brim, P. Ročkai, *Scalable Multi-core Model-Checking*, Model Checking Software, 14th International SPIN Workshop, SPIN 07 LNCS 4595, pp. 187-203, Springer, 2007.

Table 1: Planning

task/work package	time	who	deliverables
survey of the existing literature on parallel model checking	month 5	postdoc, Ph.D. student	technical report
comparative studies and improvements of the multi-core model checking algorithm for safety properties	m12	Ph.D. student, postdoc	refereed publications, implementation of the safety algorithm on top of Spin, case studies
multi-core algorithm(s) for LTL model checking	m21	Ph.D. student, postdoc	ref. publ., implementation on top of Spin, case studies
multi-core algorithms for probabilistic model checking	m24	postdoc	ref. publ., impl. on top of MRMC, case studies
improvement of multi-core probabilistic model checking algorithms	m33	postdoc	prototype impl., case studies, ref. publications
improvement of the LTL algorithms - possibly adding compatibility with approximative reduction techniques and symmetry reduction	m36	Ph.D. student	prototype implementations, case studies, ref. publications
some improvements of both standard and probabilistic multi-core algorithms	m42	Ph.D. student	tech. reports, ref. publication
working on Ph.D. thesis	m48	Ph.D. student	Ph.D. thesis

[4] J. Barnat, L. Brim, J. Stríbrná, *Distributed LTL Model Checking in SPIN*, Proc. of the 8th Intl. Spin Workshop on Model Checking of Software, SPIN 2001, LNCS 2057, pp. 200-216, Springer, 2001.

[5] J. Barnat, L. Brim, J. Chaloupka, *Parallel Breadth-First Search LTL Model-Checking*, 18th IEEE International Conference on Automated Software Engineering (ASE 2003), pp. 106-115, IEEE Computer Society, 2003.

[6] G. Behrmann, T. Hune, F. Vaandrager, *Distributed Timed Model Checking - How the Search Order Matters*, Computer Aided Verification, 12th International Conference, CAV 2000, LNCS 1855, pp. 216-231, Springer, 2000.

[7] A. Bell, B.R. Haverkort, *Distributed disk-based algorithms for model checking very large Markov chains*, Formal Methods in System Design, **29**(2), pp.177-196, 2006.

[8] D. Bošnački, D. Dams, L. Holenderski, *Symmetric Spin*, International Journal on Software Tools for Technology Transfer, 4(1), pp. 65-80, 2002.

[9] D. Bošnački, E. Elkind, B. Genest, D. Peled, *On Commutativity Based Edge Lean Search*, Automata, Languages and Programming, 34th International Colloquium, ICALP 2007, Wroclaw, Poland, July 9-13, 2007, Proceedings. LNCS 4596, pp. 158-170, Springer 2007.

[10] D. Bošnački, G. J. Holzmann, *Improving Spin's Partial-Order Reduction for Breadth-First Search*, Model Checking Software, 12th International SPIN Workshop, Lecture Notes in Computer Science 3639, pp. 91-105, Springer, 2005.

[11] D. Bošnački, S. Leue, A. Lluch Lafuente, *Partial-Order Reduction for General State Exploring Algorithms*, Model Checking Software: 13th International SPIN Workshop, Lecture Notes in Computer Science 3925, pp.271-287, Springer, 2006. (*to appear* in International Journal on Software Tools for Technology Transfer)

[12] D. Bošnački, *Partial Order Reduction in Presence of Rendez-vous Communications with Unless Constructs and Weak Fairness*, Theoretical and Practical Aspects of SPIN Model Checking, 5th and 6th International SPIN Workshops, LNCS 1680, pp. 40–56, Springer-Verlag, 1999,

[13] D. Bošnački, *Using Black Box Checking for Analysis of Metabolic Networks*, Computational Methods in Systems Biology: International Conference CMSB 2004, Revised Selected Papers, LNCS 3082, pp. 225-230, Springer, 2005.

[14] D. Bošnački, H.M.M. ten Eikelder, M.N. Steijaert, E.P. de Vink, *Stochastic Analysis of Amino Acid Substitution in Protein Synthesis*, Computational Methods in Systems Biology: International Conference CMSB 2008, LNBI 5307, pp. 367-386, Springer, 2008.

[15] L. Brim, I. Černá, P. Moravec, J. Šimša, *Accepting Predecessors are Better than Back Edges in Distributed LTL Model-Checking*, FMCAD 2004, LNCS 3312, pp. 352-366, Springer, 2004.

[16] L. Brim, I. Černá, P. Moravec, J. Šimša, *Distributed Partial Order Reductin of State Spaces*, Electronic Notes of Theoretical Computer Science, vol. 128, pp. 63-74, 2005.

[17] L. Brim, I. Černá, P. Moravec, J. Šimša, *How to Order Vertices for Distributed LTL Model-Checking Based on Accepting Predecessors*, Electronic Notes of Theoretical Computer Science, vol. 135, pp. 3-18, 2006.

[18] I. Černá, R. Pelánek, *Distributed Explicit Fair Cycle Detection (Set Based Approach)*, Model Checking Software, LNCS 2648, pp. 49-73, Springer 2003.

[19] G. Ciardo, *Distributed and Structured Analysis Approaches to Study Large and Complex Systems*, European Educational Forum: School on Formal Methods and Performance Analysis 2000: 344-374, 2000.

[20] N. Chabrier, F. Fages, *Symbolic Model Checking of Biochemical Networks*, Computational Methods in Systems Biology, First International Workshop, CMSB 2003, LNCS 2602, pp.149-162, Springer, 2003.

[21] E. Clarke, O. Grumber, D. Peled, *Model Checking*, MIT Press, 1999.

[22] E. M. Clarke, R. Enders, T. Filkorn, S. Jha, *Exploiting symmetry in temporal logic model checking*, Form. Methods Syst. Des.,**9**, pp.77-104, 1-2, Kluwer Academic Publishers, 1996.

[23] C. Courcoubetis, M.Y. Vardi, P. Wolper, M. Yannakakis, *Memory-Efiifcient Algorithms for the Verification of Temporal Properties*, Formal Methods in System Design 1, pp. 275-288, 1992.

[24] E.A. Emerson, E.M. Clarke, *Characterizing Correctness Properties of Parallel Programs Using Fixpoints*, Automata, Languages and Programming, pp. 169-181, LNCS 85, Springer, 1980.

[25] P. Godefroid, *Partial Order Methods for the Verification of Concurrent Systems: An Approach to the State Space Explosion*, LNCS 1032, Springer, 1996.

[26] G.J. Holzmann, *The SPIN Model Checker: Primer and Reference Manual*, Addison Wesley, 2003.

[27] G.J. Holzmann, D. Bošnački, *The Design of a multi-core extension of the Spin Model Checker* IEEE Trans. on Software Engineering, **33** (10), pp. 659-674, October 2007. (first presented at: Formal Methods in Computer Aided Design (FMCAD), San Jose, November 2006.)

[28] G.J. Holzmann, D. Bošnački, *Multi-core Model Checking with Spin,* Proc. Parallel and Distributed Processing Symposium, IPDPS 2007, pp. 1-8, IEEE International, 2007.

[29] G.J. Holzmann, *A Stack-Slicing Algorithm for Multi-Core Model Checking*, Proc. 6th Intnl. Workshop on Parallel and Distributed Methods in Verification, July, 2007.

[30] C.P. Inggs, H. Barringer, *CTL* Model Checking on a Shared Memory Architecture*, Electronic Notes in Theoretical Computer Science, **128** (4), pp. 107-123, 2005.

[31] C.P. Inggs, H. Barringer, *Effective State Exploration for Model Checking on a Shared Memory Architecture*, Electronic Notes in Theoretical Computer Science, **68** (4), 2002.

[32] C.N. Ip, David L. Dill, *Better Verification Through Symmetry*, Formal Methods in System Design, **9** 1/2, 1996.

[33] S. Jabbar, S. Edelkamp, *Parallel External Directed Model Checker with Linear I/O*, Proc. 7th, Intl. Conf. Verification, Model Checking and Abstract Interpretation, VMCAI 2006, LNCS 3855, pp. 237-251, Springer, 2006.

[34] D.N. Jansen, J.-P. Katoen, M. Oldenkamp, M. Stoelinga, I. Zapreev, *How Fast and Fat is Your Probabilistic Model Checker? An Experimental Performance Comparison*, Proc. HVC 2007, LNCS 4899, pp.69-85, Springer, 2008.

[35] J.-P. Katoen, M. Khattri, I.S. Zapreev, *A Markov Reward Model Checker*, Second International Conference on the Quantitative Evaluaiton of Systems (QEST 2005), pp.243-244, IEEE Computer Society, 2005.

[36] J.-P. Katoen, D. Klink, M. Leucker, V. Wolf, *Three-Valued Abstraction for Continuous-Time Markov Chains*, Proc. of the 19th International Conference on Computer Aided Verification (CAV 07). p. 311. LNCS 4590, Springer, 2007.

[37] R. Kumar, E.G. Mercer, *Load Balancing Parallel Explicit State Model Checking*, Proc. 3rd Intl. Workshop on Parallel and Distributed Model Checking, 2004.

[38] M.Z. Kwiatkowska, G. Norman, D. Parker, *PRISM: Probabilistic Symbolic Model Checker*, Computer Performance Evaluation, Modelling Techniques and Tools 12th International Conference, TOOLS 2002, LNCS 2324, pp.200-204, Springer, 2005.

[39] M. Kwiatkowska, G. Norman, D. Parker. *Stochastic Model Checking*, Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation, LNCS 4486, pp. 220-270, Springer, 2007.

[40] A. Lluch Lafuente, *Simplified Distributed LTL Model Checking by Localizing Cycles*, Technical Report 00176, Institut für Informatik, Univ. Freiburg, Germany, 2002.

[41] F. Lerda, R. Sisto, *Distributed Model Checking in SPIN*, Theoretical and Practical Aspects of SPIN Model Checking, 5th and 6th International SPIN Workshops, LNCS 1680, pp. 22-39, Springer, 1999.

[42] R. Mardare, C. Priami, P. Quaglia, O. Vagin, *Model Checking Biological Systems Described Using Ambient Calculus*, Computational Methods in Systems Biology, International Conference CMSB 2004, LNCS 3082, pp. 85-103, Springer, 2005.

[43] A. Marowka, *Parallel Computing on Any Desktop*, Comm. of the ACM, 50 (9), pp. 75-78, September, 2007.

[44] G.E. Moore, *Cramming More Components onto Integrated Circuits*, Electonics, vol. 38, pp. 114-117, 1965.

[45] , D.M. Nicol, P. Heidelberger, *Optimistic Parallel Simulation of Continuous Time Markov Chains Using Uniformization*, J. Parallel Distrib. Comput. **18**(4), pp.395-410, 1993.

[46] OpenMP Architecture Review Board, *OpenMP Application Program Interface*, Version 2.5, (May 2005); www.openm.org/.

[47] R. Pelánek, *BEEM: BEnchmarks for Explicit Model checkers*, http://anna.fi.muni.cz/models/index.html

[48] D.A. Peled, *Combining Partial Order Reductions with On-the-Fly Model Checking*, Formal Methods on Systems Design, 8: 39-64, 1996. A previous version appeared in Computer Aided Verification 1994, LCNS 818, pp. 377-390, 1994.

[49] J.P. Quelle, J. Sifakis, *Specification and Verification of Concurrent Systems in CESAR*, 5th International Symposium on Programming, pp. 337-350, 1981.

[50] J.H. Reif, *Depth First Search os Inherently Sequential*, Information Processing Letters, **20** (5), pp. 229-234, June 1985.

[51] U. Stern, D. Dill, *Parallelizing the Murϕ Verifier*, Proc. 9th Intl. Conf. Computer Aided Verification CAV 07, LNCS 1254, pp. 256-278, Springer, 1997.

[52] A. Valmari, *Eliminating Redundant Interleavings during Concurrent Program Verification*, Proc. of Parallel Architectures and Languages Europe '89, vol. 2, LNCS 366, pp. 89-103, Springer, 1989.

[53] A. Valmari, *A Stubborn Attack on State Explosion*, in Advances in Petri Nets, LNCS 531, pp. 156-165, Springer, 1991.

[54] A. Valmari, *The State Explosion Problem*, Lectures on Petri Nets I: Basic Models, LNCS Tutorials, LNCS 1491, pp. 429-528, Springer, 1998.

[55] P. Wolper, D. Leroy, *Reliable hashing without collision detection*, Proc. 5th Int. Conference on Computer Aided Verification, CAV '93, LNCS 697, pp. 59-70, 1993.

Key publications of the research team:

1. G.J. Holzmann, D. Bošnački, *The Design of a multi-core extension of the Spin Model Checker* IEEE Trans. on Software Engineering, **33** (10), pp. 659-674, October 2007. (first presented at: Formal Methods in Computer Aided Design (FMCAD), San Jose, November 2006.)

2. C. Baier, B. R. Haverkort, H. Hermanns, J.-P. Katoen, *Model-checking Algorithms for Continuous-Time Markov Chains*, IEEE Transactions on Software Engineering, **29**(6), 2003.

3. C. Baier, J-P. Katoen, *Principles of Model Checking*, MIT Press, 950 pp, 2008.

4. D. Bošnački, D. Dams, L. Holenderski, *Symmetric Spin*, International Journal on Software Tools for Technology Transfer, **4** (1), pp. 65-80, 2002.

5. P.A.J. Hilbers, *Processor Networks and Aspects of the Mapping Problem*, Cambridge University Press, 1991.

10) Requested budget:

(1) PhD student (1 fte for 4 years)		
a) appointment	=	177,495
b) personal benchfee	=	5,000
c) additional traveling budget	=	3,000
d) Project related apparatus/software	=	5,000
Subtotal PhD student	=	190,495

(2) postdoc (1 fte for 3 years)		
a) appointment	=	174,911
b) personal benchfee	=	5,000
c) additional traveling budget	=	
d) Project related apparatus/software	=	5,000
Subtotal postdoc	=	184,911

The additional traveling budget for the PhD student is motivated by her or his stay abroad in an internationally leading group in the areas that are relevant for the project (see Section 7, Project Planning).

The project related apparatus/software item is motivated by a purchase of new multi-core computer(s) (see Section 8, Expected use of instrumentation). For illustration: the current price of an 8-core machine – dual quad-core with 32 GB of memory – is about 10,000 euros. Although this budget item is divided over the Ph.D. student and the postdoc, most likely one machine will be purchased that will be shared between them.

Total requested budget: 375,406 euro