New Directions in Symbolic Model Checking for Real-Time Systems

(ARTES Project Proposal)

Parosh Aziz Abdulla
Uppsala University
Dept. of Computer Systems
Box 325
S - 751 05 Uppsala, Sweden
tel: + 46 18 - 471 31 63
fax: + 46 - 18 - 55 02 25
e-mail: parosh@docs.uu.se
URL: http://www.docs.uu.se/~parosh

Summary

Real-time embedded systems are increasingly becoming parts of our everyday life. Examples range from TV's and telephone switches to safety-critical systems such as railway control, air traffic control, and patient monitoring systems. The behaviour of such a system is typically "reactive" in the sense that it reacts to interactions with a physical environment. It is widely recognized that it is extremely difficult to produce error-free implementations of reactive systems. Although the software industry is moving towards a more systematic approach to design, extensive simulation is still common practice to ensure the conformance of a system implementation with its intended specification. However, the complexity of the system in hand makes it usually infeasible to guarantee correctness through the use of simulation. It is therefore important to complement simulation with systematic validation and verification. This allows early detection of flaws, thus reducing the life cycle cost of the system. In fact it is acknowledged by designers that errors can often be traced back to mistakes which should be possible to discover at a much earlier stage of the design process. This project intends to unify several existing techniques and invent new ones for verification of real-time systems.

We plan to develope techniques suitable for the verification of *Programmable Logic Controllers* (PLCs). PLCs is one of the most important platforms for the realization of programmed control. They provide a uniform hardware solution which can be flexibly programmed in software. Formal verification techniques have achieved some notable success for verification of PLC-based control software in railway signalling systems The technology has the potential to shorten development times for control software by an order of magnitude. The aim of the current project is to further develop this approach.

The basic methodology from which we start is that of *model checking*. *Model checking* is a technique which was originally was invented for automatic verification of finite-state systems. A major breakthrough in the application of model checking for industrial-sized applications was achieved through the invention of symbolic methods using *Binary Decision Diagrams (BDDs)* [Bry86, BCM+90]. An important challenge is to extend the applicability of model checking in order to be able to deal with systems with infinite-state spaces, and in particular with the timing aspects (delays, timeouts) found e.g. in PLC programs. We will adopt an alternative approach for symbolic verification using *Stålmarck's method* [Stå] instead of BDDs as a search engine for model checking. Furthermore, we will study the problem of adapting several existing methods for enhancing the performance of model checking

algorithms. For instance, combining compositional analysis and abstraction techniques [BLO98a] allows us often to reduce the difficult problem of verifying a distributed system into the much simpler task of verifying each component separately.

The work of this project is intended to be carried out as a cooperation between Prover Technology AB and the department of computer systems (DoCS) at Uppsala University. Prover Technology has already used Stålmarck's method in numerous industrials application to prove properties of safety-critical systems. We also plan to involve end-users of PLC technology, in order to provide examples for evaluation, and in order to investigate the potential for incorporation into an industrial design process.

An important application for our efforts is to shorten development times for programs for PLC controllers, and other embedded control programs of a related nature. Shorter development times will result from quick feed-back to the designer which reports potential bugs and problems in his design. A formal approach, such as the one proposed this project, could potentially become a basis for component-based synthesis of control programs, even if this latter goal is not part of this particular project.

1 Background and Motivation

Current capabilities in computer technology allow more and more complicated systems to be built. At the same time correctness issues are becoming important in many practical applications, since design faults may lead to unacceptable losses in money, time, human life, etc. It is therefore important to support designers with adequate methods and tools that help to eliminate design errors.

This proposal is concerned with methods for validation and verification of safety-critical systems. Applications such as control of nuclear plants, railway signalling systems, industrial control systems, and many more are to an increasing degree becoming programmed in software which has replaced hard-wired control. Careful procedures must be devised to check the functioning of such control systems before they can be put into operation.

Programmable Logic Controllers (PLCs) is one of the most important platforms for the realization of programmed control. They provide a uniform hardware solution which can be flexibly programmed in software. PLCs can now be found more and more in everyday life, e.g., in traffic control systems, vehicle control systems, household appliances, incustrial contol. Many of these applications are safety-critical, and undetected bugs may cost large sums of money. For example, Aiken et al. [AFS98] describe an approach, in which they detected a known PLC bug which originally costed 750,000 USD in factory down time. Currently, extensive simulation and testing is the best current practice to ensure the conformance of a system implementation with its intended specification. However, the complexity of the system in hand makes it usually infeasible to guarantee correctness through the use of simulation and testing. It is therefore important to complement simulation with systematic validation and verification. This allows early detection of flaws, thus reducing the life cycle cost of the system. In fact it is acknowledged by designers that errors can often be traced back to mistakes which should be possible to discover at a much earlier stage of the design process.

In recent years *formal methods* have increasingly emerged as a possible candidate which would allow us to achieve this goal. Practically applicable formal verification methods will be very valuable to e.g. industries developing complex hardware and software systems. When performing verification, we usually deal with formal descriptions of a systems at two different abstraction levels; one which describes the design of the system, and another which states its intended properties. The task of verification is to answer the question whether the system indeed satisfies the required properties.

Despite the optimism of many formal methods experts, engineers have been reluctant to accept any existing formal methods or tools. One obstacle is that available tools often require a sophisticated knowledge of mathematics and logic which is irrelevant for the rest of the design process. Therefore, much research effort has been devoted to inventing *automatic* methods which allow to hide unnecessary details of the verification problem from the user. One such a method is that of *model checking*. Model checking is a technique for automatic verification of finite-state systems. The implementation of the system is modelled as a finite state-transition graph, while the specification is expressed in a propositional temporal logic. The satisfiability of the specification by the system is checked through performing a search of the graph. Early model checkers were able to verify states with up to tens of thousands of states. A dramatic improvement was achieved after the invention of symbolic model checking which uses *Binary Decision Diagrams (BDDs)* as a data structure [Bry86, BCM+90]. BDDs are canonical representations of propositional formulas which are used to represent sets of states and transitions among them. The introduction of BDDs extended the range of applicability of automatic verification into systems with hundreds of millions of states.

Our aim is to develop formal verification techniques for finding errors in this application area. The aim of the current project is to further develop this approach by attacking some well-defined problems.

While symbolic model checking using BDDs is considered as a remarkable breakthrough for formal verification of hardware structures, there are still many types of systems which are well beyond the capabilities of the current state of the art. For time-dependent systems (real-time systems), one still has not found out how to adapt the BDD based techniques to reasoning about delays. Thus, symbolic state-space exploration methods which deal separately with each particular control state (as e.g., implemented in UPPAAL) is currently the most efficient verification method. Another obstacle is that although BDD technology currently is able to handle routinely designs with several hundred logical gates, there are many types of designs which are inherently difficult for BDD verifiers, and BDD type model checkers cannot currently handle very large designs (with, say, 10^5 logical gates).

A less explored alternative to BDDs is to use more general procedures for checking satisfiability of propositional formulas. Such SAT based techniques have been successfully applied in various domains such as hardware verification [SBSV92, She98], modal logics [GS96], and verification of railway control systems [Bor97b, Bor97a]. Typically, such applications demand solving the satisfiability of propositional formulas with thousands of (boolean) variables. A notable example is *Stålmarck's method* [Stå], which is a patented proof procedure for proposition logic developed by Gunnar Stålmarck. This method has been used on large industrial case-studies involving thousands of variables. An industrial company, Prover Technology, uses this method as a basis for current development of tools to verify embedded systems.

Recently, Ed Clarke and his colleagues have investigated the use of SAT based procedures for symbolic model checking. In two preliminary papers [BCCZ98, BCC+98] they show that SAT based procedures significantly outperforms BDD based approaches on some (chosen) examples. This paper must be viewed as very preliminary, since the evaluation is made on a rather carefully selected set of examples, and since it is rather obvious that some further optimizations must be made to the technology before it can be used on a broader scale.

The purpose of this proposed project is to develop technology for using SAT based procedures for the verification of timed systems. The main issues to be considered are:

- Adapting SAT based procedures with mechanisms for reasoning about time, analogously to the reasoning about time in tools like UPPAAL. There is currently a capability for limited reasoning about arithmetic in Prover's implementation of Stålmarck's method, which may serve as a basis.

- Developing an appropriate way to represent PLC programs in a form that is suitable for mechanical verification. It should be investigated whether (a subset of) the wide-spread IEC 1131 standard for PLC programs can be translated to such a representation.

- Evaluating the approach on a set of benchmarks, some of which are industrial controllers designed with PLC technology. A comparison with standard symbolic state-space exploration should be made.

- Developing a set of optimization techniques that will make the technology usable. A key problem with the approach in [BCCZ98, BCC+98] is that it copes badly with state spaces of great depth. Deep state spaces typically occur in systems with intricate synchronization, such as communication protocols. It may well be that state-space exploration will continue to be the best technology for communication protocols. However, we strongly believe that partial-order methods [God96] and Abstract interpretation will be necessary and highly advantageous when using SAT based procedures for model checking

2 PLC Programs

PLCs are designed to carry out control tasks in an embedded environment, where inputs signals are typically received from sensors and output signals are sent to actuators. PLC programs execute in a nonterminating loop. In each cycle, values of input variables are read, the program computes a new internal state and updates values of the output variables. The program may in addition read and update the values of internal timers. There are several standard programming language notations for PLCs, the most important being IEC 1131-1 [IEC93].

In the project, we aim at finding a representation for PLC programs, including their timing behavior, in a formalism which is suitable for verification. Earlier work on verification includes the approach at Adtranz, which is based on propositional logic. The cyclic paradigm is, we believe, modeled particularly well by languages that exploit the so-called "synchronicity hypothesis". A good example is the language Lustre, but Esterel is also an alternative. Lustre has a very clean logical execution model, but currently does not model timers. in the form that is found in e.g., timed automata and UPPAAL. Esterel is a programming language which also allows to model timing behavior. Our aim is to find a good modeling framework which combines the advantages of the synchronous paradigm with the ability to model explicit timing. We should mention that the approach in [Die97], which models PLC programs by timed automata has the problems that two time-scales are represented in the same way: both the small delays within an execution cycle, and the longer delays enforced by the PLC timers, are modeled by timers in a timed automaton. We feel that this approach will lead to a blow-up in the complexity of verification.

3 Stålmarck's Method

For many years now, most propositional theorem provers have either been based on the resolution principle [Rob65] or semantic tableaux [Bet59, Smu68]. Recently provers based on BDD [Bry86] have also gained popularity. The theorem prover underlying the program Prover^{TM} is a relatively new and patented proof method developed by Gunnar Stålmarck [Stå, Stå89, SS90, WV96]. Stålmarck's method has already been found to be superior to BDDs in some applications [GvVK95]. A key property which we hope to be useful is that the method is not sensitive to the size of the formula, provided that the formula has a low hardness degree (has a shallow proof). The experience gained at Prover Technology from past industrial applications suggest that most systems of interest generate easy formulas.

Stålmarck's method is founded on the notion of *proof depth*. The depth of a (natural deduction) proof is the maximal number of nested assumptions in the proof. A natural strategy is to search for a shallow subformula proof since it seems reasonable to assume that a theorem with only deep proofs is intrinsically hard, at least for automatic theorem provers, and a solution cannot be expected without manual help or hints.

4 Objectives and Results

The goal of the project is to contribute to extending the applicability and efficiency of model checking algorithms. More concretely, we shall implement a prototype design tool to perform model checking using Stålmarck's method. Our aim is to develop techniques so that a procedures like Stålmarcks' will be able to prove large designs of real-time systems in which time plays a critical role.

The project will be carried out along two main complementary tracks: one which concentrates on developing the basic techniques, and another which is mainly devoted to applications through building a tool, and integrating existing trelevant tools such as NP-tools (at Prover), InVeSt [BLO98b] (Verimag, Grenoble), and UPPAAL(Uppsala).

Symbolic Model checking of Timed Systems Currently, the most efficient way to model check timed systems is to use a symbolic variant of state-space exploration. This is the method used in UPPAAL. However, state-space exploration is limited in that it explicitly enumerates the reachable control states of the system. This currently gives an upper bound of (on the order of, say $10^6 - 10^9$) on the number of control states that can be handled.

BDD based symbolic model checking overcomes this limitation in number of control states by encoding them symbolically. In this way, state spaces of size 10^{50} can often be handled. However, there is currently no good method for extending BDDs to reason about timing delays.

In this project, we will adopt the idea of *bounded model checking* to verification of timed systems. In some sense, bounded model checking combines the efficient features of symbolic model checking and state-space exploration.

The idea is very simple: One is given a symbolic model of the system and a functional correctness property to be checked. In bounded model checking one constructs a logical formula which symbolically encodes an execution where the correctness property is violated. Then a SAT based method is used to check whether the formula is satisfiable. If the formula is satisfiable, then a violating execution (called a counterexample) is found.

The advantage of SAT based procedures is that counterexamples are found very fast (if they are displayed in short executions). The disadvantage is that the expression of very long counterexamples are big, and that for systems with intricate synchronization, it is hard to demonstrate the absence of counterexamples (thus establishing that the system satisfies the intended correctness property).

In our project, we intend to adapt bounded model checking to timed systems, and to find cures for the weaknesses of bounded model checking

Adapting Bounded Model checking to Timed systems In principle, this step should be rather straight-forward. Let us consider timed automata, a popular model for timed systems. Timed automata are finite automata extended with clocks that enforce timeouts and delays. In model checking for timed systems, one uses real-valued variables to represent the values of clocks. Passage of time enters in the form of constraints on how delays and timeouts interact with the other events of the system. It should be worked out how this affects the setup for bounded model checking.

Coping with the Weaknesses of Bounded Model Checking The basic idea in Bounded Model Checking is to search for counter-examples symbolically, using theorem-proving techniques. The problem is that one may have to search for very long counter-examples, leading to very long expressions of counterexamples, possibly exceeding the capacity of the prover at hand. Much work will be needed to address this question. Here are some ideas:

Compositional Verification and Abstraction In [BLO98b, BLO98a] a tool is described for computing the abstraction of a system of parallel processes. The method can handle infinite-state systems (real-time systems in our case), and is compositional in the sense that the abstraction of each process can be computed independently. The abstraction is carried out through the evaluation of certain types of predicates on the underlying domain (e.g. the domain of integers or real numbers). Our algorithms seem to be suitable as efficient "predicate evaluators", and consequently can be used for computing abstractions. This part of the project will conducted in cooperation with Verimag, Grenoble.

Partial Order Methods *Partial order methods* [God96, AKP97, AJKP98]. are based on the observation that concurrent actions of parallel processes are often independent and hence their interleavings are equivalent in that they all lead to the same states. Thus, state-space search can be optimized by investigating only some of these independent interleavings. Partial order methods has given substantial reductions for explicitly state-space exploration methods. However, their strength is that they reduce the *number* of counterexamples, not their length. Although partial order methods should certainly be investigated, we cannot be sure that a significant improvement will result for bounded symbolic model checking.

Fixedpoint Acceleration A technique for shortening potential counterexamples is to increase the power of single steps in a counterexample [ABJ98]. For example, a loop which increments a counter may generate a very long (potentially unbounded) counterexample. However, we may be able to precompute the effect of executing the loop an arbitrary number of times and represent this effect by a single execution step. In this way, the long counterexample is reduced to just a single step. Techniques of this form for fixedpoint acceleration are not extensively investigated, and more work needs to be done. Our hope is that this idea will lead to a significant increase in the power of SAT based methods.

5 Industrial Applications

An important objective of the project is to evaluate our efforts on end-user applications provided by the industrial partners of Prover Technology. Also, we plan to have a close cooperation with the the ESPRIT project *Integrating Formal Approaches to Specification, Test Case Generation and Automatic Design Verification (FAST)* in which Prover Technology participate together Ansaldo, CISE, Sextant Avionique, and Volvo. The objective of FAST is to develop new logic-based tools for system specification and design by combining a powerful temporal logical language (TRIO [MMG92, FM94]) with an decision efficient decision procedure (NP-tools). We believe that our symbolic verification methods will be very useful as a complement to the validation techniques based on model generation which are currently being developed within FAST. Furthermore, the industrial-scale verification problems provided by the end users (Ansaldo, Sextant Avionique, and Volvo) will be an important source of benchmarks for our implementation efforts. In the project, we plan to involve end-users of e.g., PLC technology, in order to provide examples for evaluation, and in order to investigate the potential for incorporation into an industrial design process.

6 Budget

The total staff in this project, for which we apply for support from TFR ARTES amounts to one lecturer (Parosh Abdulla) at 20 % and two PhD students.

7 Project Plan

The project will be carried out along two main complementary tracks: one which concentrates on developing the basic techniques, and another which is mainly devoted to applications through building a tool, and integrating existing relevant tools such as NP-tools (at Prover), InVeSt [BLO98b] (Verimag, Grenoble), and UPPAAL(Uppsala). We plan to recruit two PhD students, each working in one of the two above directions.

Project Period 9904 - 0404 (2+2 years research + 1 year teaching).

Milestones

9901 - 9903: Recruitment of two PhD students.

9904 - 9912: Gaining state of the art knowledge in model checking, verification of infinite-state systems, and propositional decision procedures. Working with relevant tools e.g. NP-tools, InVeSt, and UPPAAL. Specifications of PhD theses.

0001 - 0106: Research and publications equivalent to a Licentiate thesis for each student.

0106 - 0306: Refinement of theory. Industrial-sized applications.

0306- 0404: Writing PhD theses.

References

[ABJ98] Parosh Aziz Abdulla, Ahmed Bouajjani, and Bengt Jonsson. On-the-fly analysis of systems with unbounded, lossy fifo channels, 1998. To appear in Proc. CAV '98.

[AFS98] A. Aiken, M. Fändrich, and Z. Su. Detecting races in relay l.adder logic programs. In Bernhard Steffen, editor, *Proc. TACAS '98, 7^{th} Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems*, volume 1384 of *Lecture Notes in Computer Science*, pages 184–200, 1998.

[AJKP98] Parosh Aziz Abdulla, Bengt Jonsson, Mats Kindahl, and Doron Peled. A general approach to partial order reductions in symbolic verification. In Alan J. Hu and Moshe Y. Vardi, editors, *Proc. 10^{th} Int. Conf. on Computer Aided Verification.* Springer Verlag, 1998.

[AKP97] Parosh Aziz Abdulla, Mats Kindahl, and Doron Peled. An improved search strategy for Lossy Channel Systems. In Tadanori Mizuno, Nori Shiratori, Teruo Hegashino, and Atsushi Togashi, editors, *FORTE X / PSTV XVII '97*, pages 251–264. Chapman and Hall, 1997.

[BCC+98] A. Biere, A. Cimatti, E.M. Clarke, M. Fujita, and Y. Zhu. Symbolic model checking using SAT procedures instead of BDDs, 1998. Manuscript.

[BCCZ98] A. Biere, A. Cimatti, E.M. Clarke, and Y. Zhu. Symbolic model checking without BDDs, 1998. Manuscript.

[BCM+90] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic model checking: 10^{20} states and beyond. In *Proc. 5^{th} IEEE Int. Symp. on Logic in Computer Science*, 1990.

[Bet59] E.W. Beth. *The Foundations of Mathematics*. North Holland, 1959.

[BLO98a] S. Bensalem, Y. Lakhnech, and S. Owre. Computing abstractions of infinite state systems automatically and compositionally. In Alan J. Hu and Moshe Y. Vardi, editors, *Computer Aided Verification*, volume 1427 of *Lecture Notes in Computer Science*, pages 319–331. Springer-Verlag, 1998.

[BLO98b] S. Bensalem, Y. Lakhnech, and S. Owre. Invest : A tool for the verification of invariants. In Alan J. Hu and Moshe Y. Vardi, editors, *Computer Aided Verification*, volume 1427 of *Lecture Notes in Computer Science*, pages 505–510. Springer-Verlag, 1998.

[Bor97a] Arne Borälv. A fully automated approach for proving safety properties in interlocking software using automatic theorem-proving. In *Proceedings of the Second International ERCIM Workshop on Formal Methods for Industrial Critical Systems*, pages 39–62, 1997.

[Bor97b] Arne Borälv. The industrial success of verification tools based on stålmarck's method. In *Proc. 9^{th} Int. Conf. on Computer Aided Verification*, number 1254 in Lecture Notes in Computer Science, pages 7–10, 1997.

[Bry86] R.E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Trans. on Computers*, C-35(8):677–691, Aug. 1986.

[Die97] H. Dierks. PLC-automata: A new class of implementable real-time automata. In Larsen M. Bertran and T. Rus, editors, *ARTS'97, Transformation-Based REactive Systems Development*, volume 1231 of *Lecture Notes in Computer Science*, pages 111–125. Springer Verlag, 1997.

[FM94] M. Felder and A. Morzenti. Validating real-time systems by history-checking trio specifications. *ACM TOSEM*, 3(4), Oct. 1994.

[God96] P. Godefroid. *Partial-Order Methods for the Verification of Concurrent Systems: An Approach to the State-Explosion Problem.* PhD thesis, University of Liège, 1996. Vol. 1032 of Lecture notes in Computer Science, Springer Verlag.

[GS96] F. Giunchiglia and R. Stebastiani. Building decision procedures for modal logics from propositional decion procedures - the case study of modal k. In *Proc. 13th Conference on Automated Deduction*, volume 1104 of *Lecture Notes in Artificial Intelligence.* Springer Verlag, 1996.

[GvVK95] J.F. Groote, S.F.M. van Vlijmen, and J.W.C. Koorn. The safety guaranteeing system at station hoorn-kersenboogerd. In *COMPASS 95*, 1995.

[IEC93] IEC. IEC international standard 1131-3, programmable controllers, prt 3, programming languages. Technical report, IEC, 1993.

[MMG92] A. Morzenti, D. Mandrioli, and C. Ghezzi. A model-parametric real-time logic. *ACM TOPLAS*, 14(4):521–573, Oct. 1992.

[Rob65] J.A. Robinson. A machine-oriented logic based on the resolution principle. *Journal of the ACM*, 12(1):23–41, 1965.

[SBSV92] P. R. Stephan, R. K. Brayton, and A. L. Sangiovanni-Vincentelli. Combinational test generation using satisfiability. Technical Report M92/112, Department of Electrical Engineering and Computer Science, University of California at Berkley, October 1992.

[She98] Mary Sheeran. How to prove properties of recursively defiend circuits using stlmarck's method. In *Workshop on Formal Methods for Hardware and Hardware-like systems, Marstrand*, 1998.

[Smu68] R.M. Smullyan. *First Order Logic*. Springer Verlag, 1968.

[SS90] G. Stålmark and M. Säflund. Modelling and verifying systems and software in propositional logic. In *SAFECOMP '90*, pages 31–36. Pergamon Press, 1990.

[Stå] G. Stålmark. A system for determining propositional logic theorems by applying values and rules to triplets that are generated from a formula. Swedish Patent No. 467 076 (approved 1992), US patent No. 5 276 897 (1994), European Patent No. 0403 454 (1995).

[Stå89] G. Stålmark. En metod och anordning fr tautologicheckning, 1989. Svenskt patent nr. 467 076.

[WV96] M. J. Wieczorek and J. Vytopil. Stålmarck's notion of n-saturationåra t. Technical Report NP-K-FW-200, Logikkonsult NP AB, 1996.