Publications

Journals

- **[SBK10]**: B. Schlich, J. Brauer and S. Kowalewski, *"Application of Static Analyses for State-Space Reduction to the Microcontroller Binary Code"*, Science of Computer Programming, Special Issue on FMICS 2007 & 2008, Elsevier. To appear. (PDF)

This article describes the application of two abstraction techniques, namely dead variable reduction and path reduction, to microcontroller binary code in order to tackle the state-explosion problem in model checking. These abstraction techniques are based on static analyses, which have to cope with the peculiarities of binary code such as hardware dependencies, interrupts, recursion, and globally accessible memory locations. An interprocedural static analysis framework is presented that handles these peculiarities. Based on this framework, extensions of dead variable reduction and path reduction are detailed. A case study using several microcontroller programs is presented in order to demonstrate the efficiency of the described abstraction techniques.

Refereed Conference/Workshop Papers

- **[BKKN10]**: J. Brauer, V. Kamin, S. Kowalewski, and T. Noll, *"Loop Refinement Using Octagons and Satisfiability"*, 5th International Workshop on Systems Software Verification (SSV 2010), Vancouver, Canada. To appear.

This paper presents a technique for refining the control structure of loops in programs operating over finite bit-vectors. This technique is based on abstract interpretation using octagons and affine equalities in order to identify infeasible sequences of loop iterations. Our approach naturally integrates wrap-around arithmetic during the generation of abstractions. Abstract interpreters operating on a refined control structure then typically derive strengthened program invariants without having to rely on complicated domain constructions such as disjunctive completions.

- **[SBB10]**: S. Biallas, J. Brauer, and S. Kowalewski, *"Counterexample-Guided Abstraction Refinement for PLCs"*, 5th International Workshop on Systems Software Verification (SSV 210), Vancouver, Canada. To appear.

This paper presents a method for model checking programs for programmable logic controllers (PLCs) using the counterexample-guided abstraction refinement (CEGAR) approach. The described technique is tailored to this specific hardware platform by accounting for the cyclic scanning mode that is symptomatic to PLCs. In particular, the hardware model poses the need for on-the-fly abstraction refinement in order to guarantee a deterministic control flow. It also allows to treat refinement phases triggered by input and global variables differently, leading to a more effective implementation. The effectiveness of CEGAR for this domain is shown in a case study, which highlights the verification process for function blocks that implement a specification provided by the industrial consortium PLCopen.

- **[BKK10]**: J. Brauer, A. King, and S. Kowalewski, *"Range Analysis of Microcontroller Code Using Bit-Level Congruences"*, 15th International Workshop on Formal Methods for Industrial Critical Systems (FMICS 2010), Antwerp, Belgium. Volume 6371 of LNCS, pages 82-98, Springer, 2010. To appear. (PDF)

Bitwise instructions, loops and indirect data access pose difficult challenges to the verification of microcontroller programs. In particular, it is necessary to show that an indirect write does not mutate registers, which are indirectly addressable. To prove this property, among others, this paper presents a relational binary-code semantics and details how this can be used to compute program invariants in terms of bit-level congruences. Moreover, it demonstrates how congruences can be com- bined with intervals to derive accurate ranges, as well as information about strided indirect memory accesses.

- **[BK10]**: J. Brauer and A. King, *"Automatic Abstraction for Intervals using Boolean Formulae"*, 17th International Static Analysis Symposium (SAS 2010), Perpignan, France. Volume 6337 of LNCS, pages 167-183, Springer, 2010. To appear. (PDF)

Traditionally, transfer functions have been manually designed for each operation in a program. Recently, however, there has been growing interest in computing transfer functions, motivated by the desire to reason about sequences of operations that constitute basic blocks. This paper focuses on deriving transfer functions for intervals — possibly the most widely used numeric domain — and shows how they can be computed from Boolean formulae which are derived through bit-blasting. This approach is entirely automatic, avoids complicated elimination algorithms, and provides a systematic way of handling wrap-arounds (integer overflows and underflows) which arise in machine arithmetic.

- **[GBK10]**: D. Gueckel, J. Brauer, and S. Kowalewski, *"A System for Synthesizing Abstraction-Enabled Simulators for Binary Code Verification"*, International Symposium on Industrial Embedded Systems (SIES 2010), Trento, Italy, pages 118-127, IEEE.

Formal verification of embedded software is crucial in safety-critical applications, ideally requiring as little human intervention as possible. Binary code model checking based on hardware simulators already comes close to this goal, although with high initial effort for developing a simulator of the respective target platform. In the embedded systems domain with its varieties of different architectures in use, this can severely restrict the applicability of this approach. To remedy this drawback, we describe a system for automatically synthesizing simulators, which are suited for model checking in that they support auto- matic abstraction. We evaluate the practicality of this approach by synthesizing simulators for the ATMEL ATmega16 and Intel MCS-51 microcontrollers.

- **[BNS10]**: J. Brauer, T. Noll, and B. Schlich, *"Interval Analysis of Microcontroller Code using Abstract Interpretation of Hardware and Software"*, 13th International Workshop on Software and Compilers for Embedded Systems (SCOPES 2010), St. Goar, Germany. To appear. (PDF)

Static analysis is often performed on source code where intervals – possibly the most widely used numeric abstract domain – have successfully been used as a program abstraction for decades. Binary code on microcontroller platforms, however, is different from high-level code in that data is frequently altered using bitwise operations and the results of operations often depend on the hardware configuration. We describe a method that combines word- and bit-level interval analysis and integrates a hardware model by means of abstract interpretation in order to handle these peculiarities. Moreover, we show that this method proves powerful enough to derive invariants that could so far only be verified using computationally more expensive techniques such as model checking.

- **[GSBK10]**: D. Gückel, B. Schlich, J. Brauer, and S. Kowalewski *"Synthesizing Simulators for Model Checking Microcontroller Binary Code"*, 13th IEEE International Symposium on Design & Diagnostic of Electronic Circuits & Systems (DDECS 2010), Vienna, Austria, pages 313-316, IEEE.

Model checking of binary code is recognized as a promising tool for the verification of embedded software. Our ap- proach, which is implemented in the [MC]SQUARE model checker, uses tailored simulators to build state spaces for model checking. Previously, these simulators have been

generated by hand in a time-consuming and error-prone process. This paper proposes a method for synthesizing these simulators from a description of the hardware in an architecture description language in order to tackle these drawbacks. The application of this approach to the Atmel ATmega16 microcontroller is detailed in a case study.

- **[RHB+09]**: T. Reinbacher, M. Horauer, B. Schlich, J. Brauer, and F. Scheuer, *"Model Checking Assembly Code of an Industrial Knitting Machine"*, 4th International Conference on Embedded and Multimedia Computing (EM-Com 2009), Jeju, Korea. To appear.

Microcontrollers are used in many embedded systems. The reliability of these embedded systems is of great importance. Model checking is seen as a promising tool for the analysis of the corresponding software. For this purpose, an on-the-fly CTL model checker for microcontroller assembly code called [mc]square was developed at the RWTH Aachen University. This paper describes a case study that was conducted using [mc]square. The aim of the case study was to model check the software of an industrial embedded system used for monitoring a knitting machine without manually modifying the code. Using model checking, we found a bug in the communication protocol of the application that was not revealed during testing.

- **[BSRK09]**: J. Brauer, B. Schlich, T. Reinbacher, and S. Kowalewski, *"Stack Bounds Analysis for Microcontroller Assembly Code"*, 4th Workshop on Embedded Systems Security (WESS 2009), Grenoble, France. To appear.

Abstract:An important criterion for correctness of embedded software is stack safety, which requires that the stack must never overflow. This paper presents a static analysis for assembly code that determines upper and lower bounds of the stack. These bounds serve two purposes. First, they can be used to verify stack safety. Second, they can be used to increase the precision of several other static analyses, which are used in the context of model checking. Interrupts play an important role in embedded software, but they are a major challenge for the static analysis of stack bounds. In different microcontroller architectures, the handling of interrupts varies. In some architectures, interrupt handlers are executed atomically, while in others, they are interruptible. Therefore, we applied this analysis to two different microcontrollers, namely the ATMEL ATmega16 and the Intel MCS-51. In a case study, we show the applicability and efficiency of this analysis.

- **[SNBB09]**: B. Schlich, T. Noll, J. Brauer, and L. Brutschy, *"Reduction of Interrupt Handler Executions for Model Checking Embedded Software"*, Haifa Verification Conference (HVC 2009), Haifa, Israel. To appear. (PDF)

Abstract: Interrupts play an important role in embedded software. Unfortunately, they aggravate the state-explosion problem that model checking is suffering from. Therefore, we propose a new abstraction technique based on partial order reduction that minimizes the number of locations where interrupt handlers need to be executed during model checking. This significantly reduces state spaces while the validity of the verification results is preserved. The paper details the underlying static analysis which is employed to annotate the programs before verification. Moreover, it introduces a formal model which is used to prove that the presented abstraction technique preserves the validity of the branching-time logic CTL*-X by establishing a stutter bisimulation equivalence between the abstract and the concrete transition system. Finally, the effectiveness of this abstraction is demonstrated in a case study.

- **[RBHS09]**: T. Reinbacher, J. Brauer, M. Horauer, and B. Schlich: *"Refining Assembly Code Static Analysis for the Intel MCS-51 Microcontroller"*, IEEE Symposium on Industrial Embedded Systems (SIES 2009), Lausanne, Switzerland. To appear.

Abstract: Embedded systems are ubiquitous and their software is in most cases the elaborate part of the system. The use of formal verification methods such as model checking was proposed to verify these software systems. One disadvantage of model checking is that it suffers from the state-explosion problem. [mc]square combines model checking and static source code analysis at assembly code level to alleviate this downside. This approach allows considering particular features of the targeted microcontroller. In this paper, a novel data-flow analysis termed register bank analysis is presented. This analysis is an extension of a reaching definitions analysis to cope with register bank switching as performed by the Intel MCS-51 target. An informal and a formal description of the register bank analysis is given and an example to highlight the effectiveness of our approach is presented. Moreover, four remaining challenges in assembly code static analysis are pointed out.

- **[BHS09]**: J. Brauer, R. Huuck, and B. Schlich, *"Interprocedural Pointer Analysis in Goanna"*, in Proc. 4th International Workshop on Systems Software Verification (SSV 2009), Aachen, Germany, 2009. To appear.

Abstract: Goanna is an industrial-strength static analysis tool used in academia and industry alike to

find bugs in C/C++ programs. Unlike existing approaches, Goanna uses the off-the-shelf model checker NuSMV as its core analysis engine on a syntactic flow-sensitive program abstraction. The CTL-based model checking approach enables a high degree of flexibility in writing checks and scales to large code bases. In this paper, a new approach to pointer analysis for C is described. It is detailed how this technique is integrated into the model checking approach to perform interprocedural analysis. The performance and precision of this approach are demonstrated using a case study.

- **[SBWK09]**: B. Schlich, J. Brauer, J. Wernerus, and S. Kowalewski, *"Direct Model Checking of PLC Programs in IL"*, in Proc. Dependable Control of Discrete Systems (DCDS 2009), Bari, Italy, 2009. To appear. (PDF)

Abstract: While there are several approaches applying model checking to PLC programs, it is still not used in industry. This is due to the limited applicability of the existing approaches, which all translate PLC programs into the input languages of existing model checkers and thus suffer from certain problems. This paper presents a new approach that applies model checking directly to PLC programs written in IL without using translations. This has some advantages: domain-specific information is available during verification, users can make propositions about all features of the PLC, and counterexamples are given in the same language as the program, thus, simplifying the process of locating errors. In the described approach, a tailored simulator builds the state space for verification. Within this simulator, different abstraction techniques are used to tackle the state-explosion problem. A case study shows the applicability of this approach.

- **[BSK09]**: J. Brauer, B. Schlich, and S. Kowalewski, *"Parallel and Distributed Invariant Checking of Microcontroller Software"*, in Proc. Workshop on Systems Software Verification (SSV 2009), Aachen, Germany, 2009. To appear.

Abstract: Formal verification techniques are recognized as promising tools for the development of embedded systems. One such technique is invariant checking, which can be applied intuitively by developers as it does not require knowledge of temporal logics. State spaces for invariant checking are built using the same methods as used for model checking. They can become large due to the state-explosion problem. In [mc]square, which is a model checker for microcontroller programs, most of the time is spent building state spaces when checking programs. To improve the performance of [mc]square, we have implemented four parallel and one distributed algorithm for invariant checking. Parallel algorithms are especially helpful as they allow to fully utilize multi-core CPUs. This paper describes several parallel and distributed algorithms and presents a case study that compares these algorithms and shows the performance improvements achieved.

- **[FHSB08]**: A. Fehnker, R. Huuck, S. Seefried, and J. Brauer, *"Goanna: Syntactic Software Model Checking"*, 6th International Symposium on Automated Technology for Verification and Analysis (ATVA), October 20-23, 2008 Seoul, Korea.

Abstract: Goanna is an industrial-strength static analysis tool used in academia and industry alike to find bugs in C/C++ programs. Unlike existing approaches Goanna uses the off-the-shelf NuSMV model checker as its core analysis engine on a syntactic flow-sensitive program abstraction. The CTL-based model checking approach enables a high degree of flexibility in writing checks, scales to large number of checks, and can scale to large code bases. Moreover, the tool incorporates techniques from constraint solving, classical data flow analysis and a CEGAR inspired counterexample based path reduction. In this paper we describe Goanna's core technology, its features and the relevant techniques, as well as our experiences of using Goanna on large code bases such as the Firefox web browser.

Theses

- J. Brauer, *"Interprocedural Static Analysis by Model Checking"*, Diploma Thesis, Programming Languages and Compiler Construction Group, University of Kiel, Germany. Supervisors: Priv.-Doz. Dr.rer.nat. Frank Huch and Dr.rer.nat. Ralf Huuck. The work was carried out while visiting NICTA.

From:
https://www.embedded.rwth-aachen.de/ - **Informatik 11 - Embedded Software**

Permanent link:
https://www.embedded.rwth-aachen.de/doku.php?id=en:lehrstuhl:mitarbeiter:brauer:publications

Last update: **2011/11/21 17:27**