MoVES Newsletter

Work Package 6 - Verification Methods & Tools

University in Focus: University of Liège

Moves Newsletter, No. 7, August 2011

Editorial

Dear Reader, this edition of the MoVES newsletter is dedicated to Workpackage 6, focused on verification methods and tools, and presents results obtained by the University of Liège on that topic.

This issue of the newsletter describes three recent results that cover different research topics in the field of verification. The first result, presented on p. 2, introduces an original method for analyzing concurrent programs running on multi-core processors. These processors actually rely on a memory architecture that has particular properties, which can introduce errors in the execution of programs that have not been specifically tailored to this architecture. Using a symbolic approach, we have developed a method for analyzing the properties of multi-core programs, as well as for translating automatically programs from one memory model to another.

The second result, described on p. 3, is more theoretical. These past years, we have pioneered the use of automata as symbolic formalisms for representing the set of configurations handled during state-space exploration of systems. In particular, this approach has been applied to the symbolic representation of sets of numbers and vectors. The contribution is here to provide a complete characterization of the set of vectors that can be represented symbolically by automata-based formalisms.

The automata-based symbolic representations developed in the framework of verification can also be applied to other problems. The third presented result, p. 4, introduces *Implicit Real-Vector Automata*, which are data structures that are able to represent canonically arbitrary polyhedra in n-dimensional space.

Finally, the University of Liège will host next September a summer school on Verification Technology, Systems & Applications. Attendance is free (but registration is mandatory), and a few places have been reserved for MoVES partners, so do not hesitate to contact us if you are interested.

Enjoy reading!

Bernard Boigelot

Upcoming Event & Recent Publications

Event

Summer School on Verification Technology, Systems & Applications, Montefiore Institute, University of Liège, September 19th-23rd, 2011. Detailed information and program on p. 5. MoVES partners are especially welcome.

Recent publications

Alexander Linden, Pierre Wolper: *A Verification-Based Approach to Memory Fence Insertion in Relaxed Memory Systems*, Proc. SPIN 2011, pp. 144–160.

Julien Brusten: *On the Sets of Real Vectors Recognized by Finite Automata in Multiple Bases*, PhD thesis, Université de Liège, 2011.

Bernard Boigelot, Julien Brusten, Jean-François Degbomont: *Implicit Real Vector Automata*, Proc. INFINITY 2010, pp. 63–76.

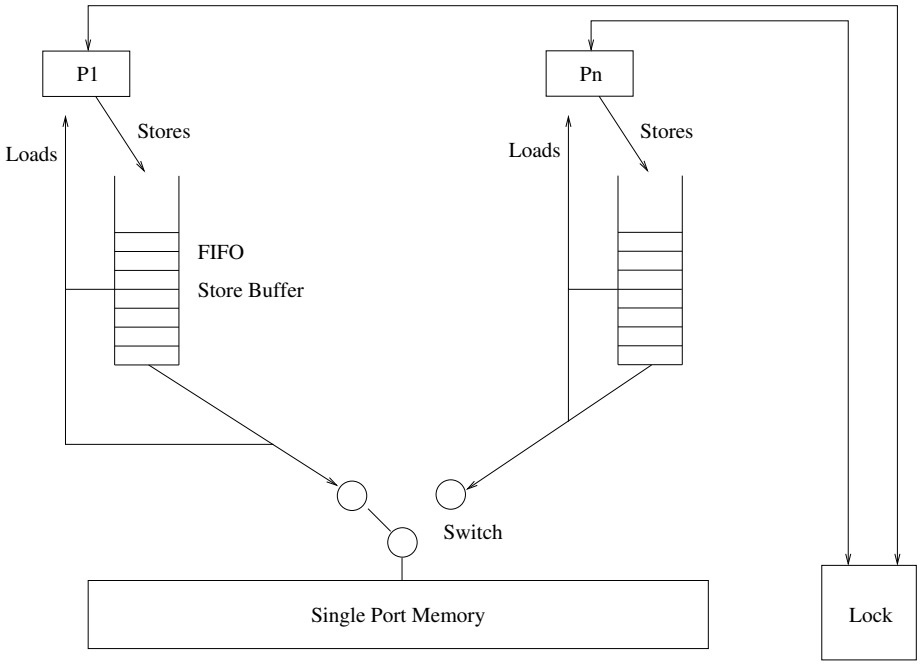
Bernard Boigelot: *Domain-specific Regular Acceleration*, To appear in a special issue of *Software Tools for Technology Transfer* dedicated to Regular Model Checking, 2011.

Bernard Boigelot, Julien Brusten, Véronique Bruyère: *On the Sets of Real Numbers Recognized by Finite Automata in Multiple Bases*, Logical Methods in Computer Science 6(1): pp. 1–17, 2010.

Alexander Linden, Pierre Wolper

Verifying Programs on Relaxed Memory Models

Model-checking tools such as SPIN [Hol93] verify concurrent programs under the traditional *Sequential Consistency* (*SC*) [Lam79] memory model, in which all accesses to the shared memory are immediately visible globally. However, modern multiprocessor architectures implement relaxed memory models, such as *Total Store Order* (*TSO*) [Spa94] (or its extension *x86-TSO* [SSO+10] with locks, illustrated in the picture), which allow many more possible executions and thus can introduce errors that were not present in SC. Of course, one can force a program executed in the context of TSO to behave exactly as in SC by adding synchronization operations after every memory access. But this totally defeats the performance advantage that is precisely the motivation for implementing relaxed memory models, rather than SC. Thus, when moving a program to an architecture implementing a relaxed memory model (which includes most current multi-core processors), it is essential to have tools to help the programmer check if correctness (i.e. a safety property) is preserved and, if not, to minimally introduce the necessary synchronization operations.



The proposed verification approach [LW10] uses an operational store-buffer based semantics of the chosen relaxed memory model and proceeds by using finite automata for symbolically representing the possible contents of the buffers. Store, load, commit and other synchronization operations then correspond to operations on these finite automata.

The advantage of this approach is that it operates on (potentially infinite) sets of buffer contents, rather than on individual buffer configurations, and that it is compatible with partial order reduction techniques. This provides a way to tame the explosion of the number of possible buffer configurations, while preserving the full generality of the analysis. It is thus possible to check even designs that exploit the relaxed memory model in unusual ways, and that may contain cycles.

We have also proposed a memory fence insertion algorithm [LW11] that finds how to preserve the correctness of a program when it is moved from SC to TSO. Its starting point is a program that is correct for the usual sequential consistency memory model (with respect to a given safety property), but that might be incorrect under x86-TSO. This program is then analyzed for this relaxed memory model and when errors are found (a broken safety property), memory fences are inserted in order to avoid these errors. The approach proceeds iteratively and heuristically, inserting memory fences until correctness is obtained, which is guaranteed to happen.

In future work, we will investigate how to adapt our techniques to other common memory models, such as *Partial Store Order* (*PSO*) [Spa94], as well as how to optimally use of the partial order reduction techniques.

References

[Hol93] Gerard J. Holzmann. *The SPIN Model Checker: Primer and Reference Manual*. Addison-Wesley Professional, September 2003.

[Lam79] Leslie Lamport. How to Make a Multiprocessor Computer That Correctly Executes Multiprocess Programs. *IEEE Trans. Computers*, 28(9):690–691, 1979.

[LW10] Alexander Linden and Pierre Wolper. An Automata-Based Symbolic Approach for Verifying Programs on Relaxed Memory Models. In *SPIN'10*, pages 212–226, 2010.

[LW11] Alexander Linden and Pierre Wolper. A Verification-Based Approach to Memory Fence Insertion in Relaxed Memory Systems. In *SPIN'11 (to appear)*, 2011.

[Spa94] CORPORATE SPARC International, Inc. *The SPARC architecture manual (version 9)*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1994.

[SSO+10] Peter Sewell, Susmit Sarkar, Scott Owens, Francesco Zappa Nardelli, and Magnus O. Myreen. x86-TSO: A Rigorous and Usable Programmer's Model for x86 Multiprocessors. *Commun. ACM*, 53:89–97, July 2010.

Further Information: *http://orbi.ulg.ac.be/handle/2268/91581*

Contact: *Alexander.Linden@ulg.ac.be*

Julien Brusten

On the Sets of Real Vectors Recognized by Finite Automata in Multiple Bases

The verification of infinite-state systems, in particular the reachability analysis of systems modeled as finite-state machines extended with unbounded variables, has prompted the development of symbolic data structures for representing the sets of vectors of values that have to be handled during state-space exploration.

A simple representation strategy consists in using finite-state automata: The vectors with components in the considered domain are encoded as words over a given finite alphabet; a set of vectors is thus encoded as a language. If this language is regular, then a finite-state automaton that accepts it forms a representation of the set.

This approach has many advantages: Regular languages are closed under all usual set-theory operators (intersection, union, complement, Cartesian product, projection, ...), and automata are generally easy to manipulate algorithmically. Deterministic automata can also be reduced to a canonical form, which simplifies comparison operations between sets.

The expressive power of automata is also well suited for verification applications. In the case of programs manipulating unbounded integer variables, it is known for a long time that the sets of integer vectors that can be recognized by a finite-state automaton using the positional encoding of numbers in a base $r > 1$ correspond to those definable in an extension of Presburger arithmetic, i.e., the first-order additive theory of the integers $\langle \mathbb{Z}, +, < \rangle$. Furthermore, the well known Cobham's and Semenov's theorems characterizes the sets that are representable by automata in all bases $r > 1$ as being exactly those that are Presburger-definable.

In order to analyze systems relying on integer and real variables, such as timed or hybrid automata, automata-based representations of integer vectors can be generalized to vectors with real components. From a theoretical point of view, this amounts to moving from finite-word to infinite-word automata, which leads to a data structure known as the *Real Vector Automaton (RVA)*. It has been shown that the sets of real vectors that can be recognized by infinite-word automata in a given encoding base are those definable in an extension of the first-order additive theory of real and integers variables $\langle \mathbb{R}, \mathbb{Z}, +, < \rangle$.

In practice though, handling infinite-word automata can be difficult, especially if set complementation needs to be performed. It is however known that, for representing the sets definable in $\langle \mathbb{R}, \mathbb{Z}, +, < \rangle$, the full expressive power of Büchi automata is not required, and that the much simpler subclass of *weak deterministic* automata is sufficient [BJW05]. The advantage is that, from an algorithmic perspective, handling weak automata is similar to manipulating finite-word automata.

A natural question is then to characterize precisely the expressive power of automata representing sets of real vectors.

For weak deterministic automata, we established that the sets of real vectors simultaneously recognizable in two *multiplicatively independent bases* are necessarily definable in the additive theory of reals and integers. For general automata, we showed that the multiplicative independence is not sufficient, and we proved that, in this context, the sets of real vectors that are recognizable in two bases that do not share the same set of prime factors are exactly those definable in the additive theory of reals and integers [BBB10,BBL09].

Since recognizability in two multiplicatively dependent bases is equivalent to recognizability in only one of them, those results lead to a complete characterization of the sets of real vectors that are recognizable in multiple bases, and provide a theoretical justification to the use of weak deterministic automata as symbolic representations of sets, by showing that the sets recognizable by general infinite-word automata in every encoding base are exactly the same as the ones recognizable by weak deterministic automata in every base.

As additional contribution, we also obtained valuable insight into the internal structure of automata recognizing sets of vectors definable in the additive theory of reals and integers [Bru11]. This documentation has been exploited in order to improve the efficiency of the decision procedure for that arithmetic [BBD10].

References

[BBB10] B. Boigelot, J. Brusten, and V. Bruyère. On the sets of real numbers recognized by finite automata in multiple bases. *Logical Methods in Computer Science*, 6(1):1–17, 2010.

[BBD10] B. Boigelot, J. Brusten, and J.-F. Degbomont. Implicit Real Vector Automata. In *Proceedings of the 12th International Workshop on Verification of Infinite-State Systems*, volume 39 of *Electronic Proceedings in Theoretical Computer Science*, pages 63–76, Singapore, September 2010.

[BBL09] B. Boigelot, J. Brusten, and J. Leroux. A generalization of Semenov's theorem to automata over real numbers. In *Proceedings of the 22nd International Conference on Automated Deduction*, volume 5663 of *Lecture Notes in Artificial Intelligence*, pages 469–484. Springer, July 2009.

[BJW05] B. Boigelot, S. Jodogne, and P. Wolper. An effective decision procedure for linear arithmetic over the integers and reals. *ACM Transactions on Computational Logic*, 6(3):614–633, 2005.

[Bru11] J. Brusten. *On the sets of real vectors recognized by finite automata in multiple bases*. PhD thesis, Université de Liège, 2011.

Further Information: *http://orbi.ulg.ac.be/handle/2268/92036*
Contact: *Julien.Brusten@ulg.ac.be*

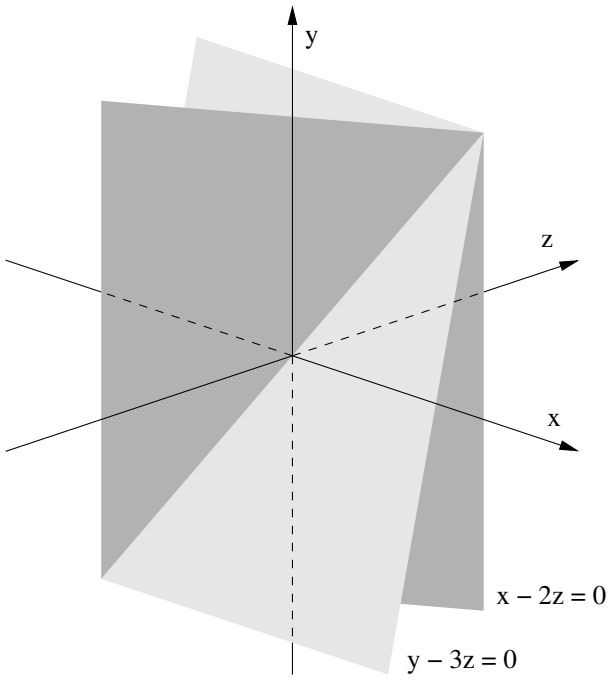
Bernard Boigelot, Julien Brusten, Jean-François Degbomont

Implicit Real-Vector Automata

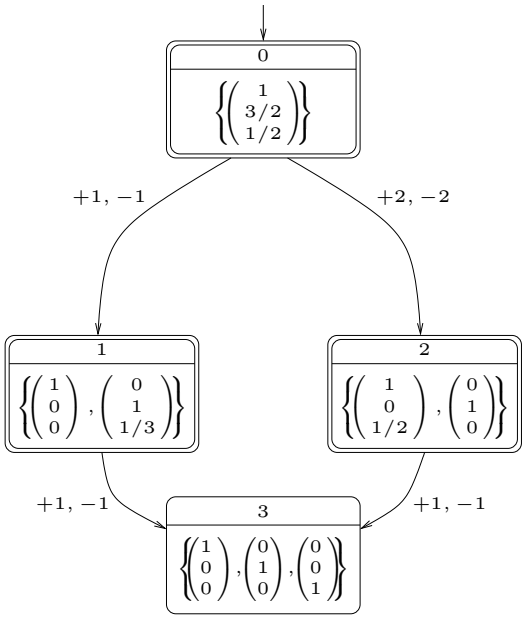
Algorithms and data structures for handling systems of linear constraints are extensively used in many areas of computer science such as computer-aided verification of systems. In this application we need a symbolic representation for representing the sets of the reachable data values computed during the state-space exploration of such systems.
In our work, we consider the sets that can be represented by arbitrary finite Boolean combinations of linear constraints over real vectors. Intuitively, a non-trivial linear constraint in the n-dimensional space describes either a $(n-1)$-plane, or a half-space bounded by such a plane. A Boolean combination of constraints thus defines a region of space delimited by planar boundaries, that is, a *polyhedron*.
Our goal is to develop an efficient data structure for representing arbitrary polyhedra, as well as associated manipulation algorithms. Among the requirements, one should be able to build representations of elementary polyhedra (such as the set of solutions of individual constraints), to apply Boolean operators in order to combine polyhedra, and to test their equality, inclusion, emptiness, and whether a given point belongs or not to a polyhedron.
We have developed an original data structure, the *Implicit Real Vector Automaton* [BBD10], for representing such sets. This representation is based on an implicit and concise encoding of a known structure, the Real Vector Automaton. The resulting formalism provides a canonical representation of polyhedra. This feature is very useful to be able to test efficiently inclusions and equalities between sets. The representation is also closed under Boolean operators, and admits a simple and efficient decision procedure for testing the membership of a vector.

Example

This example shows an IRVA representing the polyhedron $x - 2z = 0 \cup y - 3z = 0$, expressed as the union of two planes in 3-dimensional space. Intuitively, the IRVA consists in an acyclic transition graph, whose nodes are labeled by vector spaces corresponding to the characteristic elements of the polyhedron: its two supporting planes, the intersection line between these planes, and its exterior points. The reachability relation between the nodes describes the incidence relation between the characteristic elements.



(a) The set $x - 2z = 0 \cup y - 3z = 0$



(b) An IRVA representing this set

References

[BBD10] B. Boigelot, J. Brusten, and J.-F. Degbomont. Implicit Real Vector Automata. In *Proceedings of the 12th International Workshop on Verification of Infinite-State Systems*, volume 39 of *Electronic Proceedings in Theoretical Computer Science*, pages 63–76, Singapore, September 2010.

Further Information: *http://orbi.ulg.ac.be/handle/2268/74767*
Contact: *jfdegbomont@ulg.ac.be*

Summer school on Verification Technology, Systems & Applications

The summer school on verification technology, systems & applications will take place at the Montefiore Institute (University of Liège) from September 19th-23rd, 2011. We believe that all three aspects verification technology, systems and applications strongly depend on each other and that progress in the area of formal analysis and verification can only be made if all these aspects are considered as a whole. Our five speakers Alessandro Armando, Franz Baader, Bruno Blanchet, Florent Jacquemard, and Joost-Pieter Katoen stand for this view in that they represent and will present a particular verification technology and its implementation in a system in order to successfully apply the approach to real world verification problems. They will give the following tutorials.

The Rewriting Approach to Decision Procedures, *Alessandro Armando*.

Program analysis and automated verification require decision procedures to reason on theories of data structures. Many problems can be reduced to the satisfiability of sets of ground literals in decidable, first-order theory T. The rewrite approach to decision procedures amounts to using the well-known superposition-based inference system for first-order equational logic for deciding satisfiability in various theories of interest in verification (including the theory of lists, encryption, extensional arrays, extensional finite sets, and their combinations). This tutorial provides an introduction to the rewrite approach to decision procedures and also shows that, contrary to the folklore, a general-purpose prover non only can compete but can also outperform ad-hoc decision procedures with built-in theories, thereby showing that the rewriting approach is not only elegant and conceptually simple, but has important practical implications.

Automatic Symbolic Analysis of Access Control Policies, *Alessandro Armando*.

In computer systems access control policies specify who can access which resources. If on the one hand, policies must guarantee that a minimal set of users can access certain resources (principle of least privilege), on the other hand, they should be flexible enough to to support a wide range of application scenarios. For this reason, the traditional models for access control (such as Role-Based Access Control, RBAC) have been extended in several ways, e.g. with delegation and with rules for policy administration (Administrative RBAC, ARBAC). All these aspects make the analysis of access control policies so complex to make manual inspection of the policy unfeasible. Automated analysis techniques are thus of paramount importance to check security policies against desired security requirements. This tutorial introduces an approach to the automatic analysis of access control policies that leverages recent symbolic model checking for infinite state systems and state-of-the-art theorem proving techniques.

Reasoning in Description Logics, *Franz Baader*.

Description Logics (DLs) are a successful family of logic-based knowledge representation formalisms, which can be used to represent the conceptual knowledge of an application domain in a structured and formally well-understood way. They are employed in various application domains, such as natural language processing, configuration, and databases, but their most notable success so far is the adoption of the DL-based language OWL DL as standard ontology language for the semantic web.

This tutorial concentrates on designing and analyzing reasoning procedures for DLs. After a short introduction and a brief overview of the research of the last 20 years, it will on the one hand present approaches for reasoning in expressive DLs, which are the foundation for reasoning in OWL DL. On the other hand, it will consider reasoning in the more light-weight DLs EL and DL-Lite, which are the foundation for the more tractable OWL 2 profiles OWL 2 EL and OWL 2 QL.

Automatic Verification of Security Protocols: the verifier ProVerif, *Bruno Blanchet*.

After a brief overview of the field of automatic protocol verification, we will focus on the tool ProVerif. This verifier can verify a wide range of security properties of the protocols, such as secrecy, authentication, and limited cases of process equivalences, in a fully automatic way. It supports cryptographic primitives defined by rewrite rules or by certain equations. Furthermore, the obtained security proofs are valid for an unbounded number of sessions of the protocol, in parallel or not. ProVerif relies on an abstract representation of the protocol by a set of Horn clauses, and on a resolution algorithm on these clauses. We will explain this algorithm.

Tree automata techniques for the verification of infinite state-systems, *Florent Jacquemard*.

Tree automata are an appropriate formalism for the verification of safety properties of programs and systems whose states can be represented by tree structures. This the case for instance of networks of communicating processes with a tree architecture, functional programs manipulating tree data values with pattern matching, imperative programs with recursive procedure calls and thread creation, transformations of XML documents...

In this tutorial, we will present classes of tree automata over ranked and unranked trees, and their properties and decision procedures interesting in the context of verification. We will also present how their expressiveness can be extended with some constraints, and by considering tree languages modulo equational theories.

Verification and Abstraction of Continuous-Time Markov Models, *Joost-Pieter Katoen*.

Continuous-time Markov chains (CTMCs) are omnipresent. They are used as semantical backbone of Markovian queueing networks, stochastic Petri nets, stochastic process algebras, and calculi for systems biology.

The lectures will focus on the analysis of CTMCs using model checking. Both temporal-logic based model checking as well as (timed) automata-based model checking will be considered. Equivalences and pre-orders are covered and we'll discuss which logical fragments are preserved by them. Finally, we introduce three-valued abstraction on CTMCs, and apply it to the analysis of examples from systems biology and (large) queueing networks. As outlook, we'll discuss recent progress on the analysis of continuous-time Markov decision processes.

Further Information: *http://www.mpi-inf.mpg.de/VTSA11/*
Contact: *vtsa11@montefiore.ulg.ac.be*

Partners


Universiteit
Antwerpen


FUNDP
NAMUR


UMONS
Université de Mons


MoVES
Modelling,Verification and Evolution of Software


Vrije
Universiteit
Brussel


KATHOLIEKE UNIVERSITEIT
LEUVEN


ULg
UNIVERSITÉ de Liège


ULB


INRIA
centre de recherche
LILLE - NORD EUROPE


Imperial College
London


TU Delft
Delft
University of
Technology