Centre Fédéré en Vérification

Technical Report number 2005.43

On the use of Automata-based Techniques in Symbolic Model Checking

Axel Legay, Pierre Wolper

This work was partially supported by a FRFC grant: 2.4530.02

http://www.ulb.ac.be/di/ssd/cfv

On the use of Automata-based Techniques in Symbolic Model Checking

Axel Legay [1,3] Pierre Wolper [2]

Institut Montefiore
Université de Liège,
Liège, Belgium

At the heart of all the techniques that have been proposed for exploring infinite state spaces, is a symbolic representation that can finitely represent infinite sets of states. In early work on the subject, this representation was domain specific, for example linear constraints for sets of real vectors. For several years now, the idea that a generic finite-automaton based representation could be used in many settings has gained ground, starting with systems manipulating queues and integers [8,11,9,13], then moving to parametric systems [6], and, recently, reaching systems using real variables [10,2].

For exploring an infinite state space, one does not only need a finite representation of infinite sets, but also techniques for finitely computing the effect of an unbounded number of transitions. Such techniques can be domain specific or generic. Domain specific techniques exploit the specific properties and representations of the domain being considered and were, for instance, obtained for queues in [15,14], for integers and reals in [17,22,12], for pushdown system in [18,16], and for lossy channels in [19]. Generic techniques consider finite-automata representations and provide algorithms that operate directly on this representation, mostly disregarding the domain for which it is used.

Generic techniques appeared first in the context of the verification of systems whose states can be encoded by *finite words*, such as parametric systems. The idea used there is that a configuration being a finite word, a transition relation is a relation on finite words, or equivalently a language of pairs of finite words. If this language is regular, it can be represented by a finite state automaton, more specifically a finite-state *transducer*, and the problem then becomes the one of iterating such a transducer. Finite state transducers are

[1] Email: legay@montefiore.ulg.ac.be
[2] Email: pw@montefiore.ulg.ac.be
[3] Axel Legay is supported by a F.R.I.A grant.

This is a preliminary version. The final version will be published in Electronic Notes in Theoretical Computer Science
URL: www.elsevier.nl/locate/entcs

quite powerful (the transition relation of a Turing machine can be modeled by a finite-state transducer), the flip side of the coin being that the iteration of such a transducer is neither always computable, nor regular. Nevertheless, there are a number of practically relevant cases in which the iteration of finite-state transducers can be computed and remains finite-state. Identifying such cases and developing (partial) algorithms for iterating finite-state transducers has been the topic, referred to as "regular model checking", of a series of recent papers [1,20,21,5,23].

The question that initiated the work [3] presented in this talk is, whether the generic techniques for iterating transducers could be fruitfully applied in cases in which domain specific techniques had been exclusively used so far. In particular, one of our goals was to iterate finite-state transducers representing arithmetic relations (see [22] for a survey). Beyond mere curiosity, the motivation was to be able to iterate relations that are not in the form required by the domain specific results, for instance disjunctive relations. Initial results were very disappointing: the transducer for an arithmetic relation as simple as $(x, x+1)$ could not be iterated by existing generic techniques. However, looking for the roots of this impossibility through a mix of experiments and theoretical work, and taking a pragmatic approach to solving the problems discovered, we were able to develop an approach to iterating transducers that easily handles arithmetic relations, as well as many other cases. Interestingly, it is by using a tool for manipulating automata (LASH [24]), looking at examples beyond the reach of manual simulation, and testing various algorithms that the right intuitions, later to be validated by theoretical arguments, were developed. Implementation was thus not an afterthought, but a central part of our research process.

The general approach that has been taken is similar to the one of [21] in the sense that, starting with a transducer T, we compute powers T^i of T and attempt to generalize the sequence of transducers obtained in order to capture its infinite union. This is done by comparing successive powers of T and attempting to characterize the difference between powers of T as a set of states and transitions that are added. If this set of added states, or *increment*, is always the same, it can be inserted into a loop in order to capture all powers of T. However, for arithmetic transducers comparing T^i with T^{i+1} did not yield an increment that could be repeated, though comparing T^{2^i} with $T^{2^{i+1}}$ did. So, a first idea we used is not to always compare T^i and T^{i+1}, but to extract a sequence of samples from the sequence of powers of the transducer, and work with this sequence of samples. Given the binary encoding used for representing arithmetic relations, sampling at powers of 2 works well in this case, but the sampling approach is general and different sample sequences can be used in other cases [4]. Now, if we only consider sample powers T^{i_k} of the transducers and compute $\bigcup_k T^{i_k}$, this is not necessarily equivalent to

[4] as an example, it is often linear when considering parametric systems

computing $\bigcup_i T^i$. Fortunately, this problem is easily solved by considering the reflexive transducer, i.e. $T_0 = T \cup T_I$ where T_I is the identity transducer, in which case working with an infinite subsequence of samples is sufficient.

Once the automata in the sequence being considered are constructed and compared, and that an increment corresponding to the difference between successive elements has been identified, the next step is to allow this increment to be repeated an arbitrary number of times by incorporating it into a loop. There are some technical issues about how to do this, but no major difficulty. Once the resulting "extrapolated" transducer has been obtained, one still needs to check that the applied extrapolation is safe (contains all elements of the sequence) and is precise (contains no more). An easy to check sufficient condition for the extrapolation to be safe is that it remains unchanged when being composed with itself. Checking preciseness is more delicate, but we have developed a procedure that embodies a sufficient criterion for doing so. The idea is to check that any behavior of the transducer with a given number k of copies of the increment, can be obtained by composing transducers with less than k copies of the increment. This is done by augmenting the transducers to be checked with counters and proving that one can restrict theses counters to a finite range, hence allowing finite-state techniques to be used.

Taking advantage of the fact that our extrapolation technique works on automata, not just on transducers, we consider computing reachable states both by computing the closure of the transducer representing the transition relation, and by repeatedly applying the transducer to a set of initial states. The first approach yields a more general object and is essential if one wishes to extend the method to the verification of liveness properties ([1,25]), but the second is often less demanding from a computational point of view and can handle cases that are out of reach for the first. Preciseness is not always possible to check when working with state sets rather than transducers, but this just amounts to saying that what is computed is possibly an overapproximation of the set of reachable states, a situation which is known to be pragmatically unproblematic.

Going further, the problem of using regular model checking technique for systems whose states are represented by infinite (omega) words has been addressed. This makes the representation of sets of reals possible as described in [2,12]. To avoid the hard to implement algorithms needed for some operations on infinite-word automata, only omega-regular sets that can be defined by weak deterministic Büchi automata [7] are considered. This is of course restrictive, but as is shown in [2], it is sufficient to handle sets of reals defined in the first-order theory of linear constraints. Moreover using such a representation leads to algorithms that are very similar to the ones used in the finite word case, and allows us to work with reduced deterministic automata as a normal form. Due to these advantages and properties, one can show that the technique developed for the finite word case can directly be adapted to weak deterministic Büchi automata up to algorithmic modifications [4].

Our technique has been implemented in a prototype that relies in part on the LASH package. This prototype has been tested on several case studies coming from different horizons. In our experiments, we were able to iterate a variety of arithmetic (integer or real) transducers. We were also successful on disjunctive relations that could not be handled by earlier specific techniques such as [17,12]. The technique was also successfully applied to examples of parametric systems and to the analysis of Petri nets. Moreover models of hybrid systems, including a leaking gas burner and an alternating bit protocol with timers were also considered.

Attempting to verify infinite-state systems while working exclusively with automata-theoretic representations and algorithms can appear as a somewhat quixotic endeavor. However, practical results clearly shown their interest, and are thus a motivation for new developments [27,26,28].

References

[1] A. Bouajjani and B. Jonsson and M. Nilsson and Tayssir Touili, *Regular Model Checking*, Proceedings of the 12th International Conference on Computer-Aided Verification (CAV'00), Lecture Notes in Computer Science, volume 1855, year 2000, pages 403–418.

[2] B. Boigelot and S. Jodogne and P. Wolper, *On the Use of Weak Automata for Deciding Linear Arithmetic with Integer and Real Variables*, Proc. International Joint Conference on Automated Reasoning (IJCAR), Lecture Notes in Computer Science, volume 2083, year 200, pages 611–625.

[3] B. Boigelot and A. Legay and P. Wolper, *Iterating Transducers in the Large*, Proc. 15th Int. Conf. on Computer Aided Verification, Boulder, USA, Lecture Notes in Computer Science, volume 2725, year 2003, pages 223–235.

[4] B. Boigelot and A. Legay and P. Wolper, *Omega-Regular Model Checking*, Proc. 10th Int. Conf. on Tools and and Algorithms for the Construction and Analysis of Systems, Barcelona, Spain, Lecture Notes in Computer Science, volume 2988, year 2004, pages 561–575.

[5] D. Dams and Y. Lakhnech and M. Steffen, *Iterating Transducers*, Proceedings 13th International Conference on Computer Aided Verification (CAV), Lecture Notes in Computer Science, volume 2102, year 2001, pages 286–297.

[6] Y. Kesten and O. Maler and M. Marcus and A. Pnueli and E. Shahar, *Symbolic Model Checking with Rich Assertional Languages*, Proceedings of 9th International Conference on Computer-Aided Verification (CAV'97), Lecture Notes in Computer Science, volume 1254.

[7] D. E. Muller and A. Saoudi and P. E. Schupp, *Alternating automata, the weak monadic theory of the tree and its complexity*, Proc. 13th Int. Colloquium on Automata, Languages and Programming, year 1986.

[8] P. Wolper and B. Boigelot, *An Automata-Theoritic Approach to Presburger Arithmetic Constraints*, Proc. Static Analysis Symposium, Glasgow, Lecture Notes in Computer Science, volume 983, year 1995, 21-32.

[9] P. Wolper and B. Boigelot, *Verifying Systems with Infinite but Regular State Spaces*,Proc. 10th Int. Conf. on Computer Aided Verification, Vancouver, Lecture Notes in Computer Science, volume 1427, year 1998, 88-97.

[10] B. Boigelot and L. Bronne and S. Rassart, *An Improved Reachability Analysis Method for Strongly Linear Hybrid Systems*,Proc. 9th Int. Conf. on Computer Aided Verification, Haifa, Lecture Notes in Computer Science, volume 1254, year 1997, 167-177.

[11] B. Boigelot and P. Godefroid and B. Willems and P. Wolper, *The Power of QDDs*, Proc. of Int. Static Analysis Symposium, Paris, Lecture Notes in Computer Science, volume 1302, year 1997, 172-186.

[12] B. Boigelot and F. Herbreteau and S. Jodogne, *Hybrid Acceleration Using Real Vector Automata*,Proc. 15th Int. Conf. on Computer Aided Verification, Boulder, Lecture Notes in Computer Science, volume 2725, year 2003, 193-205.

[13] B. Boigelot and S. Rassart and P. Wolper, *On the Expressiveness of Real Integer Arithmetic Automata*,Proc. 25th ICALP, Lecture Notes in Computer Science, volume 1443, year 1998, 152-163.

[14] A. Bouajjani and P. Habermehl, *Symbolic Reachability analysis of FIFO channel systems with nonregular sets of configurations*, Proc. 24th ICALP, Bologna, Lecture Notes in Computer Science, volume 1256, year 1997, 560-570.

[15] B. Boigelot and P. Godefroid, *Symbolic Verification of Communication Protocols with Infinite State Spaces using QDDs*, Proc. 8th Int. Conf. on Computer Aided Verification, New-Brunswick, Lecture Notes in Computer Sciences, volume 1102, year 1996, 1-12.

[16] A. Bouajjani and J. Esparza and O. Maler, *Reachability Analysis of Pushdown Automata: Application to Model-Checking*, Proc. 8th Int. Conf. of Concurrency Theory, Warsaw, Lecture Notes in Computer Science, volume 1243, year 1997, 135-150.

[17] B. Boigelot, *Symbolic Methods for Exploring Infinite State Spaces*, Collection des Publications de la la Faculté des Sciences Appliquées de l'Université de Liège, Liège, Belgium, year 1999.

[18] A. Finkel and B. Willems and P. Wolper, *A Direct Symbolic Approach to Model checking Pushdown Systems*, Proc 2th Int. Workshop on Verification of Infinite State Systems, Bologna, Electronic Notes in Theoritical Computer Science, volume 9, year 1997.

[19] P.A. Abdulla and B. Jonsson, *Verifying Programs with Unreliable Channels*, Information and Computation, volume 127(2), year 1996, 91-101.

[20] B. Jonsson and M. Nilson, *Transitive closures of regular relations for verifying infinite-state systems*, Proc 6th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems, Lecture Notes in Computer Science, volume 1875, year 2000, 220-234.

[21] T. Touili, *Regular model checking using widening techniques*, Proc 1st. of Workshop on Verification of Parametrized Systems, Electronic Notes in Theoritical Computer Science, year 2001.

[22] B. Boigelot and P. Wolper, *Representing Arithmetic Constraints with Finite Automata: An Overview*, Proc. Int. Conf. on Logic Programming, Copenhagen, Lecture Notes in Computer Science, volume 2401, year 2002, 1–19.

[23] P. A. Abdulla and B. Jonsson and M. Nilsson and J. d'Orso, *Regular Model Checking Made Simple and Efficient*, Proc. 15th Int. Conf. on Computer Aided Verification, Boulder, USA, Lecture Notes in Computer Science, volume 2725, year 2003, pages 237-248.

[24] The Liège Automata-based Symbolic Handler (LASH). Available at `http://www.montefiore.ulg.ac.be/~boigelot/research/lash/`.

[25] A. Bouajjani and A. Legay and P. Wolper, *Handling Liveness properties in (ω-)Regular Model Checking*, To appear in ENTCS.

[26] P. A. Abdulla and A. Legay and J. D'Orso and A. Rezine, *Simulation-Based Iteration of Tree Transducers*, Proc. 11th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems, Edinburgh, U-K, Lecture Notes in Computer Science, volume 3440, year 2005, pages 30-44.

[27] A. Bouajjani and P. Habermehl and P. Moro and T. Vojnar, *Verifying Programs with Dynamic 1-Selector-Linked Structures in Regular Model Checking*, Proc. 11th Int. Conf. on Tools and and Algorithms for the Construction and Analysis of Systems, Edinburgh, U-K, Lecture Notes in Computer Science, volume 3440, year 2005, pages 13-29.

[28] A. Vardan and K. Sen and M. Viswanathan and G. Agha, *Using Language Inference to Verify Omega-Regular Properties*, Proc. 11th Int. Conf. on Tools and Algorithms for the Construction and Analysis of Systems, Edinburgh, U-K, Lecture Notes in Computer Science, volume 3440, year 2005, pages 30-44.