

T(O)RMC: A Tool for (ω)-Regular Model Checking (Tool Paper)

Axel Legay

Carnegie Mellon University, Computer Science Department, Pittsburgh, PA,
alegacy@cs.cmu.edu

1 Introduction and Motivations

Within the context of the verification of infinite-state systems, “(ω)-Regular model checking” is the name of a family of techniques in which states are represented by words, sets of states by finite automata on these objects, and transitions by finite automata operating on pairs of state encodings, i.e. finite-state transducers. If the states are encoded by finite words, then sets of (pairs of) states can be represented by finite-word automata. This setting can be used to represent various classes of infinite-state systems [17], including parametric systems, FIFO-queue systems, and systems manipulating integer variables (those defined in Presburger arithmetic). When the states are encoded by infinite words, sets of (pairs of) states are represented by deterministic weak Büchi automata¹. This setting is mainly used to represent systems involving both integer and real variables [4, 6], such as linear hybrid systems with a constant derivative.

It is known [7, 8] that computing the set of reachable states and verifying linear temporal properties in this automata-based framework reduces to solving the (ω)-regular reachability problems. Given a finite-word (resp. deterministic weak) automaton A representing the initial states and a finite-word (resp. deterministic weak) transducer T representing the transition relation, those problems amount to computing the iterative closures T^+ and $T^+(A)$. This can be done either by *specific techniques* that exploit the specific properties and representations of the domain being considered, or by *generic techniques* that consider automata-based representations and provide algorithms that operate directly on these representations.

In [5, 6], we have proposed a generic technique for computing $T^+(A)$ and T^+ for the finite and infinite word cases. Our approach consists in computing the limit of a possibly infinite sequence of automata. This is an undecidable problem to which the computation of T^+ and $T^+(A)$ as well as several others can be reduced. The technique has been evaluated and improved with the help of prototypes, which have been applied on various classes of problems [5, 6, 8, 9,

¹ A weak Büchi automaton is a Büchi automaton whose strongly connected components contain either only accepting or only nonaccepting states. Deterministic weak Büchi automata are easily complementable and admit a unique minimal form.

12]. Those prototypes predate the T(O)RMC toolset which unifies them, and is the subject of this paper².

2 The Underlying Technique from [5, 6, 12]

Given a sequence $S = A_0, A_1, A_2, \dots$ of minimal finite-word (resp. weak) automata, the technique computes a finite-word (resp. weak Büchi) automaton for $\bigcup_{i=0}^{\infty} A^i$. The general idea is to extrapolate one of the finite *sampling sequence* of S , *i.e.* selected automata from one of its finite prefixes. The extrapolation step is done by comparing successive automata in the sampling sequence, trying to identify the difference between these in the form of an *increment* (*i.e.* a difference in their graph structures), and *extrapolating* the repetition of this increment by adding loops to the last automaton of the sequence. After the extrapolation has been built, one has to check whether it corresponds to the limit of the sequence³. If this is the case, then the computation terminates. Otherwise, another sampling sequence has to be chosen.

Testing whether two (or more) automata differ by the addition of increments is decided by a combination of forward and backward language equivalences. Choosing the sampling sequence is a rather tricky issue and there is no guarantee that this can be done in a way that ensures that the extrapolation step can be applied. However, there is a number of heuristics that are very effective for obtaining a sample sequence that can be extrapolated (see Chapter 5 of [12] to know how to choose a “good” sampling strategy for your case study). Finally, checking safety of the extrapolation (does it include the limit?) is simple, but checking preciseness (is it exactly the limit?) is a much more involved problem for which only partial solutions have been developed (see Chapter 7 of [12]). In most of cases, working with an over approximation is practically sufficient.

3 The tool

The T(O)RMC toolset (available at [15]), builds upon the LASH toolset. LASH [11] takes the form of a set of C functions for building and manipulating (union, intersection, complementation, minimization, ...) both finite-word and weak Büchi automata. The tool also provides some domain specific techniques for solving the regular reachability problems. In addition, several compilers can be used to make easier the construction of specific classes of automata (e.g. those that represent solution of Presburger formulas). T(O)RMC improves LASH with three new packages, that are (see [12] and [15] for details):

1. *The transducer package* that provides data structures and algorithms to manipulate transducers. The package also provides several heuristics to improve

² States for Tool for (ω)-Regular Model Checking.

³ In practice, the user may also be satisfied with an extrapolation whose correctness is unknown.

the efficiency of the composition between transducers (sometimes from days to seconds!).

2. *The extrapolation package* for detecting increments in a sequence of automata, and extrapolating a finite sampling sequence. The tool allows the user to precise (1) which sampling strategy has to be used, and (2) how to build the successive elements in the infinite sequence.
3. *The counter-word automata package* that provides data structures and algorithms to check the correctness of the extrapolation for several classes of problems.

4 Summary of the Experiments

The T(O)RMC toolset has been evaluated over more than 100 case studies. Due to space limitations, this section only briefly recaps the classes of problems for which T(O)RMC has been used so far. Details about the experiments (including performances in terms of time and memory, which vary from examples to examples) can be found in Chapters 7 and 13 of [12] and in [8].

We first used T(O)RMC to compute an automata-based representation of the set of reachable states of several infinite-states systems, including parametric systems, FIFO-queue systems, and systems manipulating integer variables. Others experiments concerned the computation of the transitive closure of several arithmetic relations. It is worth mentioning that the disjunctive nature of some relations sometimes prevents the direct use of specific domain-based techniques [10, 3]. We also applied T(O)RMC to the challenging problem of analyzing linear hybrid systems. One of the case studies consisted of computing a precise representation of the set of reachable states of several versions of the *leaking gas burner*. To the best of our knowledge, only the technique in [3] was able to handle the cases we considered. Among the other experiments, we should also mention the computation of the set of reachable states of an augmented version of the IEEE Root Contention Protocol [12], which has been point out to be a hard problem [14]. The ability of T(O)RMC to compute the limit of an infinite sequence of automata has other applications. As an example, the tool has been used in a semi-algorithm to compute the convex hull of a set of integer vectors [9]. T(O)RMC was also used to compute a symbolic simulation over the state-space of an infinite-state system, with the aim of verifying temporal properties [8].

5 Conclusion

We presented T(O)RMC, a tool for computing the limit of an infinite sequence of finite-word or deterministic weak Büchi automata.

T(O)RMC implements a very general automata sequence extrapolation technique. As such it is slower than tools that are specific to the arithmetic domain (FAST [1], LIRA [2], LASH), but is perfectly competitive when handling other regular model checking cases (parametric systems, FIFO systems, ...) [13, 16].

The main goal of the tool is not performance improvement, but to allow experimentation with automata sequence extrapolation in a variety of context that goes beyond (omega-)Regular Model checking problems.

References

1. S. Bardin, J. Leroux, and G. Point. Fast extended release. In *Proc. 18th Int. Conference on Computer Aided Verification (CAV)*, volume 4144 of *Lecture Notes in Computer Science*, pages 63–66. Springer, 2006.
2. B. Becker, C. Dax, J. Eisinger, and F. Klaedtke. LIRA: Handling constraints of linear arithmetics over the integers and the reals. In *Proc. 19th Int. Conference on Computer Aided Verification (CAV)*, volume 4590 of *Lecture Notes in Computer Science*, pages 307–310. Springer-Verlag, 2007.
3. B. Boigelot and F. Herbretreau. The power of hybrid acceleration. In *Proc. 18th Int. CAV*, volume 4144 of *LNCS*, pages 438–451. Springer, 2006.
4. 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.
5. B. Boigelot, A. Legay, and P. Wolper. Iterating transducers in the large (extended abstract). In *Proc. 15th Int. CAV*, LNCS, pages 223–235. Springer, 2003.
6. B. Boigelot, A. Legay, and P. Wolper. Omega-regular model checking. In *Proc. 10th Int. TACAS*, volume 2988 of *LNCS*, pages 561–575. Springer, 2004.
7. A. Bouajjani, B. Jonsson, M. Nilsson, and T. Touili. Regular model checking. In *Proc. 12th Int. CAV*, volume 1855 of *LNCS*, pages 403–418. Springer-Verlag, 2000.
8. A. Bouajjani, A. Legay, and P. Wolper. Handling liveness properties in (omega-)regular model checking. In *Proc. 6th Int. INFINITY*, volume 138(3) of *ENTCS*. Elsevier, 2004.
9. F. Cantin, A. Legay, and P. Wolper. Computing convex hulls by automata iteration. In *Proc. 1th Int. AUTOMATHA*, 2007.
10. A. Finkel and J. Leroux. How to compose presburger-accelerations: Applications to broadcast protocols. In *Proc. 22rd Int. FSTTCS*, volume 2556 of *LNCS*, pages 145–156. Springer, 2002.
11. The Liège Automata-based Symbolic Handler (LASH). Available at <http://www.montefiore.ulg.ac.be/~boigelot/research/lash/>.
12. A. Legay. *Generic Techniques for the Verification of Infinite-state Systems*. Collection des publications de la Faculté des Sciences Appliquées de l’Université de Liège, Liège, Belgium, 2007. available at <http://www.montefiore.ulg.ac.be/~legay/papers/index>.
13. The regular model checking tool (RMC). Available at <http://www.it.uu.se/research/docs/fm/apv/rmc>.
14. D. P. L. Simons and M. Stoelinga. Mechanical verification of the ieee 1394a root contention protocol using uppaal2k. *International Journal STTT*, 3(4):469–485, 2001.
15. The T(O)RMC toolset. Available at <http://www.montefiore.ulg.ac.be/~legay/TORMC/index-tormc.html>.
16. A. Vardhan and M. Viswanathan. Lever: A tool for learning based verification. In *Proc. 18th Int. Conference on Computer Aided Verification (CAV)*, volume 4144 of *Lecture Notes in Computer Science*, pages 471–474. Springer, 2006.
17. P. Wolper and B. Boigelot. Verifying systems with infinite but regular state spaces. In *Proc. 10th Int. CAV*, volume 1427 of *LNCS*, pages 88–97. Springer-Verlag, 1998.