Symbolic LTL Compilation for Model Checking: Extended Abstract

Kristin Y. Rozier * with Moshe Y. Vardi

Rice University, Houston, Texas 77005,

In Linear Temporal Logic (LTL) model checking, we check LTL formulas representing desired behaviors against a formal model of the system designed to exhibit these behaviors. To accomplish this task, the LTL formulas must be translated into automata [21]. We focus on LTL compilation by investigating LTL satisfiability checking via a reduction to model checking. Having shown that symbolic LTL compilation algorithms are superior to explicit automata construction algorithms for this task [16], we concentrate here on seeking a better symbolic algorithm. We present experimental data comparing algorithmic variations such as normal forms, encoding methods, and variable ordering and examine their effects on performance metrics including processing time and scalability.

Safety critical systems, such as air traffic control, life support systems, hazardous environment controls, and automotive control systems, pervade our daily lives, yet testing and simulation alone cannot adequately verify their reliability [3]. Model checking is a promising approach to formal verification for safety critical systems which involves creating a formal mathematical model of the system and translating desired safety properties into a formal specification for this model. The complement of the specification is then checked against the system model. When the model does not satisfy the specification, model-checking tools accompany this negative answer with a counterexample, which points to an inconsistency between the system and the desired behaviors and aids debugging efforts.

LTL model checkers follow the automata-theoretic approach [21], in which the complemented LTL specification is translated to a Büchi automaton, which is then composed with the model under verification; see also [20]. The model checker then searches for a trace of the model that is accepted by the automaton. Symbolic model checkers, such as CadenceSMV [15], NuSMV [4], or VIS [1], represent the model and analyze it symbolically using binary decision diagrams (BDDs) [2]. All symbolic model checkers use the symbolic translation described in [5] and the analysis algorithm of [8], though CadenceSMV and VIS try to optimize further.

Arguably the most pressing challenge in model checking today is scalability. We must make model checking tools more efficient, in terms of the size of the models they can reason about and the time and space they require to verify a safety property, in order to scale our verification ability to handle real-world safety-critical systems.

A basic observation underlying our work is that LTL satisfiability checking can be reduced to model checking. Consider a formula φ over a set *Prop* of atomic proposi-

* Work contributing to this paper was completed at Rice University, Cambridge University, and NASA Langley Research Center, and was supported in part by the Rice Computational Research Cluster (Ada), funded by NSF under Grant CNS-0421109 and a partnership between Rice University, AMD and Cray.

tions. If a model M is *universal*, that is, it contains all possible traces over $Prop$, then φ is satisfiable precisely when the model M does *not* satisfy $\neg\varphi$. Thus, it is easy to add a satisfiability-checking feature to LTL model-checking tools. Measuring the performance of LTL satisfiability checking enables us to benchmark the performance of LTL model checking tools, and, more specifically, of LTL translation tools.

We have coded our own front-end LTL-to-automaton symbolic translator for NuSMV and CadenceSMV. Our tool compiles an input LTL formula into a symbolic automaton which can be checked against a universal model using either NuSMV or CadenceSMV for the back-end. We investigated numerous novel combinations of algorithmic constructs, including representing the formula specifications in Boolean Normal Form and Negation Normal Form, constructing the automaton using sloppy or fussy encoding, utilizing variable resolution, and applying several variable ordering algorithms from the current literature.

We report here on an experimental investigation of LTL satisfiability checking via a reduction to model checking. By using large LTL formulas, we offer challenging model-checking benchmarks. We tested our front-end LTL-to-automaton translation algorithm against the algorithms of both CadenceSMV and NuSMV, using both tools as a back-end for our translation. We used a wide variety of benchmark formulas, either generated randomly, as in [7], or using a scalable pattern (e.g., $\bigwedge_{i=1}^{n} p_i$). LTL formulas typically used for evaluating LTL translation tools are usually too small to offer challenging benchmarks. Note that real specifications typically consist of many temporal properties, whose conjunction ought to be satisfiable. Thus, studying satisfiability of large LTL formulas is quite appropriate in our goal of extending the scalability of LTL model checking tools.

We have found that the existing literature on LTL to automata translation provides little information on actual algorithm performance. There has been extensive research over the past decade into explicit translation of LTL to automata [6, 7, 9, 10, 11, 14, 12, 13, 18, 17, 19], but we previously demonstrated that symbolic tools have a clear edge over explicit tools with respect to LTL satisfiability checking [16]. Considerably less research has been done on symbolic compilation yet it is very promising. It has already been noted that automata minimization may not result in model checking performance improvement [9] and specific attention has been given to minimizing the size of the product with the model [17]. Still, no previous study of LTL translation has focused on model checking performance, leaving a glaring gap in our understanding of LTL model checking.

References

[1] R.K. Brayton, G.D. Hachtel, A. Sangiovanni-Vincentelli, F. Somenzi, A. Aziz, S.-T. Cheng, S. Edwards, S. Khatri, T. Kukimoto, A. Pardo, S. Qadeer, R.K. Ranjan, S. Sarwary, T.R. Shiple, G. Swamy, and T. Villa. VIS: a system for verification and synthesis. In *CAV, Proc. 8th Int'l Conf*, volume 1102 of *Lecture Notes in Computer Science*, pages 428–432. Springer, 1996.

[2] J.R. Burch, E.M. Clarke, K.L. McMillan, D.L. Dill, and L.J. Hwang. Symbolic model checking: 10^{20} states and beyond. *Information and Computation*, 98(2):142–170, Jun 1992.

[3] R. W. Butler and G. B. Finelli. The infeasibility of quantifying the reliability of life-critical real-time software. *IEEE Trans. Software Eng.*, 19(1):3–12, 1993.

[4] A. Cimatti, E.M. Clarke, F. Giunchiglia, and M. Roveri. NuSMV: a new symbolic model checker. *It'l J. on Software Tools for Tech. Transfer*, 2(4):410–425, 2000.

[5] E. M. Clarke, O. Grumberg, and K. Hamaguchi. Another look at LTL model checking. *Formal Methods in System Design*, 10(1):47–71, 1997.

[6] J-M. Couvreur. On-the-fly verification of linear temporal logic. In *Proc. FM*, pages 253–271, 1999.

[7] N. Daniele, F. Guinchiglia, and M.Y. Vardi. Improved automata generation for linear temporal logic. In *CAV, Proc. 11th Int'l Conf*, volume 1633 of *LNCS*, pages 249–260. Springer, 1999.

[8] E.A. Emerson and C.L. Lei. Efficient model checking in fragments of the propositional μ-calculus. In *LICS, 1st Symp.*, pages 267–278, Cambridge, Jun 1986.

[9] K. Etessami and G.J. Holzmann. Optimizing Büchi automata. In *CONCUR, Proc. 11th Int'l Conf.*, Lecture Notes in CS 1877, pages 153–167. Springer, 2000.

[10] C. Fritz. Constructing Büchi automata from linear temporal logic using simulation relations for alternating büchi automata. In *Proc. 8th Intl. CIAA*, number 2759 in Lecture Notes in Computer Science, pages 35–48. Springer, 2003.

[11] C. Fritz. Concepts of automata construction from LTL. In *LPAR, Proc. 12th Int'l Conf.*, Lecture Notes in Computer Science 3835, pages 728–742. Springer, 2005.

[12] P. Gastin and D. Oddoux. Fast LTL to Büchi automata translation. In *CAV, Proc. 13th Int'l Conf*, volume 2102 of *LNCS*, pages 53–65. Springer, 2001.

[13] R. Gerth, D. Peled, M.Y. Vardi, and P. Wolper. Simple on-the-fly automatic verification of linear temporal logic. In P. Dembiski and M. Sredniawa, editors, *Protocol Specification, Testing, and Verification*, pages 3–18. Chapman & Hall, Aug 1995.

[14] D. Giannakopoulou and F. Lerda. From states to transitions: Improving translation of LTL formulae to Büchi automata. In *FORTE, Proc of 22 IFIP Int'l Conf*, Nov 2002.

[15] K. McMillan. The SMV language. Technical report, Cadence Berkeley Lab, 1999.

[16] K.Y. Rozier and M.Y. Vardi. LTL satisfiability checking. Technical report, Rice University, 2007.

[17] R. Sebastiani and S. Tonetta. "more deterministic" vs. "smaller" büchi automata for efficient LTL model checking. In *CHARME*, pages 126–140. Springer, 2003.

[18] F. Somenzi and R. Bloem. Efficient Büchi automata from LTL formulae. In *CAV, Proc. 12th Int'l Conf*, volume 1855 of *LNCS*, pages 248–263. Springer, 2000.

[19] X. Thirioux. Simple and efficient translation from LTL formulas to Büchi automata. *Electr. Notes Theor. Comput. Sci.*, 66(2), 2002.

[20] M.Y. Vardi. Automata-theoretic model checking revisited. In *Proc. 7th Int'l Conf. on Verification, Model Checking, and Abstract Interpretation*, volume 4349 of *LNCS*, pages 137–150. Springer, 2007.

[21] M.Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. In *Proc. 1st LICS*, pages 332–344, Cambridge, Jun 1986.