GOAL Extended: Towards a Research Tool for Omega Automata and Temporal Logic*

Yih-Kuen Tsay, Yu-Fang Chen, Ming-Hsien Tsai, Wen-Chin Chan, and Chi-Jian Luo

Department of Information Management, National Taiwan University, Taiwan

Abstract. This paper reports extensions to the GOAL tool that enable it to become a research tool for omega automata and temporal logic. The extensions include an expanded collection of translation, simplification, and complementation algorithms, a command-line mode which makes GOAL functions accessible by programs, and utility functions for such common tasks as file format conversion, random formulae generation, and statistics collection.

1 Introduction

GOAL (`http://goal.im.ntu.edu.tw`) is a graphical interactive tool for defining and manipulating ω-automata, in particular Büchi automata, and temporal logic formulae. It was first formally introduced in [20]. Two most useful and distinctive functions of GOAL are (1) translation of quantified propositional linear temporal logic (QPTL) formulae into equivalent Büchi automata and (2) equivalence test between two Büchi automata or between a Büchi automaton and a QPTL formula. With these and other utility functions, the GOAL tool may be used for educational purposes and for supplementing automata-theoretic model checkers such as SPIN [8]. For example, the user may use GOAL to prepare a Büchi automaton diagram that is checked to be correct in that it is equivalent to another larger reference Büchi automaton or some easier-to-understand QPTL formula.

In this paper, we report extensions to GOAL that enable it to become a research tool for ω-automata and temporal logic. We have at present focused on Büchi automata and PTL (which subsumes LTL). The extensions and their usage for supporting research are described in the next section. Table 1 summarizes the major algorithms implemented in GOAL. Though the number of supported functions does not actually increase, a larger collection of algorithms are very useful for various research purposes. In addition, several utility functions have been implemented for common tasks in experimentation such as (1) collecting statistic data and (2) generating random automata and temporal formulae. These functions allow researchers to test correctness of their translation

* This work was partially supported by the iCAST project sponsored by the National Science Council, Taiwan, under the Grant No. NSC96-3114-P-001-002-Y.

C.R. Ramakrishnan and J. Rehof (Eds.): TACAS 2008, LNCS 4963, pp. 346–350, 2008.

Table 1. Major algorithms in GOAL. An * indicates that the algorithm had already been implemented in earlier versions of GOAL [20].

Translation LTL formula to Büchi automaton	**Complementation**
Tableau*, Inc. Tableau [9], Temp. Tester [10], GPVW [5], GPVW+ [5], LTL2AUT [1], LTL2AUT+ [19], LTL2BA [3], PLTL2BA [4]	Safra*, WAPA [18], WAA [11], Piterman [13]
	Simplification
	Simulation*, Pruning fair sets [15]

algorithm, collect comparison data, and, with GOAL's graphical interface, visually observe and manipulate automata generated from their algorithm. The extensions also enhance the original roles of GOAL as a learning/teaching tool and as a supplementary model-checking tool.

2 The Extensions for Supporting Research

In this section, we detail the extensions to GOAL and explain how they may be used for supporting research.

LTL formula to Büchi automaton

- **Translation Algorithms:** In addition to the Tableau algorithm [12], we have now implemented eight translation algorithms. Four (Tableau, Incremental Tableau, Temporal Tester, and PLTL2BA) of the nine algorithms originally support past operators. We have extended three more (GPVW, LTL2AUT, and LTL2AUT+) to allow past operators. All these nine algorithms are further extended to support quantification on propositions.

  As an illustration of usage, Table 2 summarizes the results of translating the following two equivalent formulae into generalized Büchi automata by seven algorithms:

  1. $\neg(\Diamond p \rightarrow (\neg p \,\mathcal{U}\, (q \wedge \neg p \wedge \bigcirc(\neg p \,\mathcal{U}\, r))))$
  2. $\neg\Box(p \rightarrow (\Diamond\!\!\!-(r \wedge \ominus \Diamond\!\!\!- q)))$

  Each formula is the negation of a formula stating that p must be triggered by q and r with q occurring strictly before r. The first formula with only future operators is taken from the Spec Patterns respository [16].
- **Complementation and Simplification Algorithms:** In addition to Safra's construction, GOAL now has another three complementation algorithms, including complementation via weak alternating parity automata (WAPA) [18], complementation via weak alternating automata (WAA) [11], and Piterman's construction [13]. Cross-checking greatly increases our confidence in the correctness of the different complementation algorithms, which are difficult, and hence the correctness of the language containment and equivalence tests. GOAL applies simplification algorithms to the input automata before an equivalence test, and this substantially enhances the performance. Besides the simulation-based method in [15], we have also implemented simplification heuristics based on pruning fair sets in the same work.

Table 2. Comparison of seven translation algorithms without and with simplification. The column acc indicates the number of acceptance sets.

No.	Tableau			Extended GPVW			GPVW+			Extended LTL2AUT			Extended LTL2AUT+			LTL2BA			PLTL2BA		
	st	tran	acc	st	tran	acc	st	tran	acc	st	tran	acc	st	tran	acc	st	tran	acc	st	tran	acc
1.	65	550	3	72	456	1	49	288	1	22	84	1	21	105	1	8	30	1	33	118	1
2.	49	396	1	13	55	1	-	-	-	9	23	1	8	17	1	-	-	-	15	38	1
1.	14	68	3	14	58	1	13	52	1	13	46	1	8	27	1	8	30	1	14	41	1
2.	10	26	1	7	21	1	-	-	-	5	10	1	5	10	1	-	-	-	10	23	1

- **The Command-Line Mode:** This mode makes most of the GOAL functions accessible by programs or shell scripts. It therefore provides an interface between GOAL and external tools. Sample shell scripts that compare translation algorithms and output the results as text files are provided. They can be easily adapted to handle other different tasks.
- **Utility Functions:** Utility functions are available for collecting statistic data (numbers of states, transitions, and acceptance sets) and for generating random automata and random temporal formulae. Outputs from external automata tools MoDeLLa [14] and LTL2Buchi [6] may also be converted to the GOAL File Format (GFF, which is an XML file format designed to cover all ω-automata) for further processing by GOAL.

We now describe a typical use case for the above functions, namely **checking correctness of a translation algorithm**. This task can be performed with high confidence by comparing the results of the algorithm under test with (1) those of a large number of different translation algorithms, (2) those of a reference algorithm, or (3) a set of reference answers, consisting of pairs of formulae and their equivalent Büchi automata.

We assume a reference algorithm. To carry out the correctness checking process, generate an adequate number of random temporal formulae and then apply the following procedure repeatedly for each formula f:

1. Use the reference algorithm to generate two automata A_f and $A_{\neg f}$ that are equivalent to f and $\neg f$ respectively.
2. Use the algorithm under test to translate f into an automaton B.
3. Test if both $A_{\neg f} \cap B$ and $A_f \cap \overline{B}$ are empty.

If all the emptiness tests succeed, then the algorithm should be correct. Otherwise, GOAL will produce counterexamples which can be run interactively on the automata to "see" what the problem might be. We developed our translation algorithms in this manner. GOAL helped us to find some subtle bugs and possible room for improvement.

3 Performance Evaluation and an Example Experiment

We present an experiment that, on the one hand, evaluates the performance of GOAL and, on the other, demonstrates experimental comparative studies

Table 3. Comparison of complementation algorithms without and with simplification. Only successful runs are accounted in the accumulated States, Transitions, and Time.

	States	Transitions	Time	Timeout
$\neg f$ to BA	1629	6906	$51.0s$	0
Safra	2461	11721	$175.7s$	6
Simplification+Safra	2077	9707	$22.1s + 114.8s$	5
WAPA	89196	4902278	$6346.3s$	51
Simplification+WAPA	8828	425248	$14.0s + 202.9s$	27
WAA	2920	27870	$3629.4s$	51
Simplification+WAA	1886	17740	$14.1s + 167.3s$	27
Piterman	1816	8314	$224.5s$	5
Simplification+Piterman	1531	6916	$23.4s + 442.4s$	3

that may be conducted with GOAL. The experiment was run on an Intel Xeon 3.2GHz machine with 2GB of memory allocated to the Java Virtual Machine.

In the experiment, we compared the four complementation algorithms implemented in GOAL without and with simplification. Note that the performance of a complementation algorithm dictates the performance of the equivalence test function it supports. We generated 300 random PTL formulae with a length of 5 and translated them into Büchi automata as inputs using the LTL2AUT algorithm. None of the 300 formulae are valid or unsatisfiable. The average size of the input automata is about 5.4. We set a timeout of 10 minutes. From this experiment, we found that (1) Safra's and Piterman's algorithms perform much better than complementation via WAPA and complementation via WAA and (2) simplification can significantly speed up the complementation task, especially complementation via WAPA and complementation via WAA.

In this way, can we produce automata that are neither universal nor empty?

4 Remarks

The extension of GOAL will continue to include a few more complementation algorithms, for example [2]. Another effort will be to include even more translation algorithms, in particular those that utilize intermediary automata with acceptance conditions on transitions such as [17] and those that do simplification while constructing automata on-the-fly [7]. The fact that Safra's and Piterman's algorithms in average work better than complementation via WAPA and complementation via WAA is also worthy of further investigation.

Acknowledgment. We thank Susan H. Rodger at Duke University for granting us the permission to use and modify the JFLAP source code.

References

1. Daniele, M., Giunchiglia, F., Vardi, M.Y.: Improved automata generation for linear temporal logic. In: Halbwachs, N., Peled, D.A. (eds.) CAV 1999. LNCS, vol. 1633, pp. 249–260. Springer, Heidelberg (1999)

2. Friedgut, E., Kupferman, O., Vardi, M.Y.: Büchi complementation made tighter. In: Wang, F. (ed.) ATVA 2004. LNCS, vol. 3299, pp. 64–78. Springer, Heidelberg (2004)
3. Gastin, P., Oddoux, D.: Fast LTL to Büchi automata translations. In: Berry, G., Comon, H., Finkel, A. (eds.) CAV 2001. LNCS, vol. 2102, pp. 53–65. Springer, Heidelberg (2001)
4. Gastin, P., Oddoux, D.: LTL with past and two-way very-weak alternating automata. In: Rovan, B., Vojtáš, P. (eds.) MFCS 2003. LNCS, vol. 2747, pp. 439–448. Springer, Heidelberg (2003)
5. Gerth, R., Peled, D., Vardi, M.Y., Wolper, P.: Simple on-the-fly automatic verification of linear temporal logic. In: PSTV 1995, pp. 3–18. Chapman & Hall, Boca Raton (1995)
6. Giannakopoulou, D., Lerda, F.: From states to transitions: Improving translation of LTL formulae to Büchi automata. In: Peled, D.A., Vardi, M.Y. (eds.) FORTE 2002. LNCS, vol. 2529, pp. 308–326. Springer, Heidelberg (2002)
7. Hammer, M., Knapp, A., Merz, S.: Truly on-the-fly LTL model checking. In: Halbwachs, N., Zuck, L.D. (eds.) TACAS 2005. LNCS, vol. 3440, pp. 191–205. Springer, Heidelberg (2005)
8. Holzmann, G.J.: The SPIN Model Checker: Primer and Reference Manual. Addison-Wesley, Reading (2003)
9. Kesten, Y., Manna, Z., McGuire, H., Pnueli, A.: A decision algorithm for full propositional temporal logic. In: Courcoubetis, C. (ed.) CAV 1993. LNCS, vol. 697, pp. 97–109. Springer, Heidelberg (1993)
10. Kesten, Y., Pnueli, A.: Verification by augmented finitary abstraction. In: Information and Computation, vol. 163, pp. 203–243 (2000)
11. Kupferman, O., Vardi, M.Y.: Weak alternating automata are not that weak. ACM Transactions on Computational Logic 2(3), 408–429 (2001)
12. Manna, Z., Pnueli, A.: Temporal Verification of Reactive Systems: Safty. Springer, Heidelberg (1995)
13. Piterman, N.: From nondeterministic Büchi and Streett automata to deterministic parity automata. In: LICS 2006, pp. 255–264. IEEE Computer Society, Los Alamitos (2006)
14. Sebastiani, R., Tonetta, S.: More deterministic vs. smaller Büchi automata for efficient LTL model checking. In: Geist, D., Tronci, E. (eds.) CHARME 2003. LNCS, vol. 2860, pp. 126–140. Springer, Heidelberg (2003)
15. Somenzi, F., Bloem, R.: Efficient Büchi automata from LTL formulae. In: Emerson, E.A., Sistla, A.P. (eds.) CAV 2000. LNCS, vol. 1855, pp. 248–263. Springer, Heidelberg (2000)
16. The Spec Patterns repository, `http://patterns.projects.cis.ksu.edu/`
17. Tauriainen, H.: Automata and Linear Temporal Logic: Translations with Transition-based Acceptance. PhD thesis, Helsinki University of Technology (2006)
18. Thomas, W.: Complementation of Büchi automata revisited. In: Jewels are Forever, Contributions on Theoretical Computer Science in Honor of Arto Salomaa (1998)
19. Tsai, M.-H., Chan, W.-C., Tsay, Y.-K., Luo, C.-J.: Full PTL to Büchi automata translation for on-the-fly model checking. Manuscript (2007)
20. Tsay, Y.-K., Chen, Y.-F., Tsai, M.-H., Wu, K.-N., Chan, W.-C.: GOAL: A graphical tool for manipulating Büchi automata and temporal formulae. In: Grumberg, O., Huth, M. (eds.) TACAS 2007. LNCS, vol. 4424, pp. 466–471. Springer, Heidelberg (2007)