The Spot Library, and its Online Translator from LTL (and PSL) to Büchi Automata

http://spot.lip6.fr/
Alexandre.Duret-Lutz@lrde.epita.fr

1 Introduction

Spot [12] is a C++ library of algorithms to implement the automata-theoretic approach to model-checking of linear-time temporal properties. To verify that a property φ holds on a model M of some system, this approach [19] splits the verification process into four operations:

1. Computation of the state-space for the model M. This state-space can be seen as an ω-automaton A_M whose language, $\mathscr{L}(A_M)$, represents all possible infinite executions of M.
2. Translation of the temporal property φ into an ω-automaton $A_{\neg\varphi}$ whose language, $\mathscr{L}(A_{\neg\varphi})$, is the set of all infinite executions that would invalidate φ.
3. Synchronization of these automata. This constructs a product automaton $A_M \otimes A_{\neg\varphi}$ whose language, $\mathscr{L}(A_M) \cap \mathscr{L}(A_{\neg\varphi})$, is the set of executions of M invalidating φ.
4. Emptiness check of this product. This decides whether $A_M \otimes A_{\neg\varphi}$ accepts an infinite word, and can return such a word (a counterexample) if it does. The model M verifies φ iff $\mathscr{L}(A_M \otimes A_{\neg\varphi}) = \emptyset$.

While Büchi Automata (BA) are commonly used in such a framework, Spot implements a generalization thereof called TGBA (*Transition-based Generalized Büchi Automata*) where infinite words are accepted iff they visit infinitely often one transition from each acceptance set. TGBA are as expressive as BA, but can be more compact than BA, especially when expressing *weak fairness* properties.

2 LTL Translation

Cumulated sizes of automata for 188 formulas from the litterature

Count of nondeterministic states and automata

Products with a random state-space of 200 states

	$\Sigma\lvert A_{\neg\varphi}\rvert$		nondet.		$\Sigma\lvert A_M \otimes A_{\neg\varphi}\rvert$	
	st.	tr.	st.	aut.	st.	tr.
Spin 6.1.0 (7 timeouts)	1 635	7 825	1 402	176	314 218	21 549 478
ltl2ba 1.1	1 080	3 646	871	177	215 717	12 766 425
LTL→NBA	989	3 214	784	178	197 568	12 063 463
Modella 1.5.9	1 391	4 562	679	125	274 281	10 907 038
ltl3ba 1.0.1	884	2 538	349	126	175 626	6 562 700
Spot 0.9.1	742	2 018	139	51	147 709	5 396 354

Spot is renowned for its translation of Linear-time Temporal Logic (LTL) formulas into small BA (or TGBA). The above table compares the BA output by Spot to those output by some other tools and shows that the automata output by Spot are, on the average, smaller, more deterministic, and yield substantially smaller products (in all columns, smaller numbers are better). More detailed benchmarks are available at `http://spot.lip6.fr/dl/bench-0.9.1.pdf`.

Spot obtains small automata thanks to a combination of: many syntactic simplifications on LTL formulas, a tableau construction from LTL to TGBA that uses binary decision diagrams for many simplifications [10], several post-processing simplifications including a minimization algorithm applied to *weak deterministic Büchi automata* [9] and a simulation algorithm applied to TGBA that are not *weak deterministic*.

Spot's LTL translator has been used in a number of benchmarks [16, 15, 7, 3], and as the translation step of several approaches [17, 18, 13, 6].

Spot's translator can be tested and easily demonstrated on-line at `http://spot.lip6.fr/ltl2tgba.html`. This web page offers access to three of the four translation algorithms implemented in Spot, with options to enable/disable all pre- and post-processings, and a choice of different automaton output (BA, TGBA, monitor). This interface can also be use to classify LTL formulas into the Manna-Pnueli hierachy [14], either syntactically, or (for the obligation, safety, and guarantee classes), structurally.

Recent versions of Spot have added support for the linear fragment of PSL (*Property Specification Language*) an industrial standard [1] that mixes LTL operators with extended regular expressions. This can also be demonstrated online.

Besides being a showcase for a subset of the features available in the Spot library, we have found this online translator to be a great help in our day-to-day research on LTL/PSL model-checking.

3 Model-checking and More

Amusingly, Spot's main purpose, as a library, is not really to translate temporal logic into automata: it is to help people build model checkers for their custom formalisms. All they need is to create an object that follows the TGBA interface and that performs the on-the-fly computation of the state-space of their models. TGBA are abstract objects in Spot, and they have an interface that allows on-the-fly computations: this way we build only the part of the state-space, and the part of its product with the property automaton, that the emptiness check needs to explore. We have built at least six custom model-checkers using Spot (four of which are for different flavors Petri Net models).

Spot actually offers more than just algorithms to implement the above automata-theoretic approach. It has four algorithms to translate Linear-time Temporal Logic (LTL) to TGBA or BA, five emptiness checks algorithms with some variants, two Büchi complementation algorithms, several LTL simplification rewritings, and a couple of reduction algorithms for TGBA, etc.

Spot has been used to experiment with various variants of the standard approach [8, 2, 11, 4], but also as an ω-automaton library to implement non-model-checking procedures [5].

4 Availability

Spot is developed collaboratively between LRDE (`www.lrde.epita.fr`) and LIP6 (`move.lip6.fr`). Its source code is distributed under the GNU GPL at `spot.lip6.fr`, and comes with a large test-suite. Comments, questions, and bug-reports can be sent to our mailing list at `spot@lrde.epita.fr`.

References

[1] *Property Specification Language Reference Manual v1.1*. Accellera, June 2004. `http://www.eda.org/vfv/`.

[2] S. Baarir and A. Duret-Lutz. Emptiness check of powerset Büchi automata. In *Proc. of ACSD'07*, pp. 41–50. IEEE Computer Society.

[3] T. Babiak, M. Křetínský, V. Řeehák, and J. Strejček. LTL to Büchi automata translation: Fast and more deterministic. In *Proc. of TACAS'12*, vol. 7214 of *LNCS*, pp. 95–109. Springer.

[4] A. E. Ben Salem, A. Duret-Lutz, and F. Kordon. Generalized Büchi automata versus testing automata for model checking. In *Proc. of SUMO'11*, vol. 626 of *Workshop Proceedings*. CEUR.

[5] J. Brotherston, N. Gorogiannis, and R. L. Petersen. A generic cyclic theorem prover. Submitted, 2012.

[6] P. Cabalar and M. Diéguez. STeLP — a tool for temporal answer set programming. In *Proc. of LPNMR'11*, vol. 6645 of *LNCS*, pp. 370–375. Springer, 2011.

[7] J. Cichoń, A. Czubak, and A. Jasiński. Minimal Büchi automata for certain classes of LTL formulas. In *Proc. of DEPCOS'09*, pp. 17–24. IEEE Computer Society.

[8] J.-M. Couvreur, A. Duret-Lutz, and D. Poitrenaud. On-the-fly emptiness checks for generalized Büchi automata. In *Proc. of SPIN'05*, vol. 3639 of *LNCS*, pp. 143–158. Springer.

[9] C. Dax, J. Eisinger, and F. Klaedtke. Mechanizing the powerset construction for restricted classes of ω-automata. In *Proc. of ATVA'07*, vol. 4762 of *LNCS*. Springer.

[10] A. Duret-Lutz. LTL translation improvements in Spot. In *Proc. of VECoS'11*, Electronic Workshops in Computing. British Computer Society.

[11] A. Duret-Lutz, K. Klai, D. Poitrenaud, and Y. Thierry-Mieg. Self-loop aggregation product — a new hybrid approach to on-the-fly LTL model checking. In *Proc. of ATVA'11*, vol. 6996 of *LNCS*, pp. 336–350. Springer.

[12] A. Duret-Lutz and D. Poitrenaud. SPOT: an extensible model checking library using transition-based generalized Büchi automata. In *Proc. of MASCOTS'04*, pp. 76–83. IEEE Computer Society Press.

[13] R. Ehlers and B. Finkbeiner. On the virtue of patience: minimizing Büchi automata. In *Proc. of SPIN'10*, vol. 6349 of *LNCS*, pp. 129–145. Springer.

[14] Z. Manna and A. Pnueli. A hierarchy of temporal properties. In *Proc. of PODC'90*, pp. 377–410. ACM.

[15] K. Y. Rozier and M. Y. Vardi. LTL satisfiability checking. In *Proc. of SPIN'07*, vol. 4595 of *LNCS*, pp. 149–167. Springer.

[16] R. Sebastiani, S. Tonetta, and M. Y. Vardi. Symbolic systems, explicit properties: on hybrid approches for LTL symbolic model checking. In *Proc. of CAV'05*, vol. 3576 of *LNCS*, pp. 350–363. Springer.

[17] M. Staats and M. P. E. Heimdahl. Partial translation verification for untrusted code-generators. In *Proc. of ICFEM'08*, vol. 5256 of *LNCS*, pp. 226–237. Springer.

[18] D. Tabakov and M. Y. Vardi. Optimized temporal monitors for SystemC. In *Proc. of RV'10*, vol. 6418 of *LNCS*, pp. 436–451. Springer.

[19] M. Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Proc. of Banff'94*, vol. 1043 of *LNCS*, pp. 238–266. Springer.