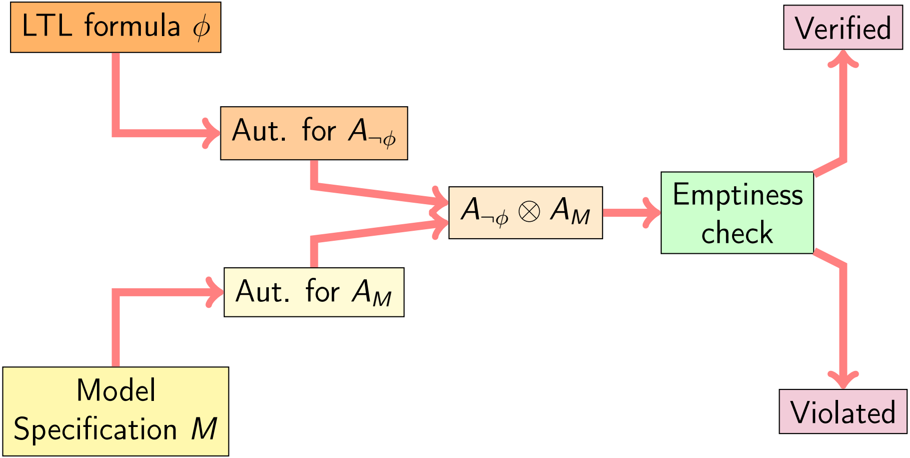
Strength-based decomposition of the property Büchi automaton for faster model checking

E. Renault, A. Duret-Lutz, F. Kordon, D. Poitrenaud

LRDE/LIP6

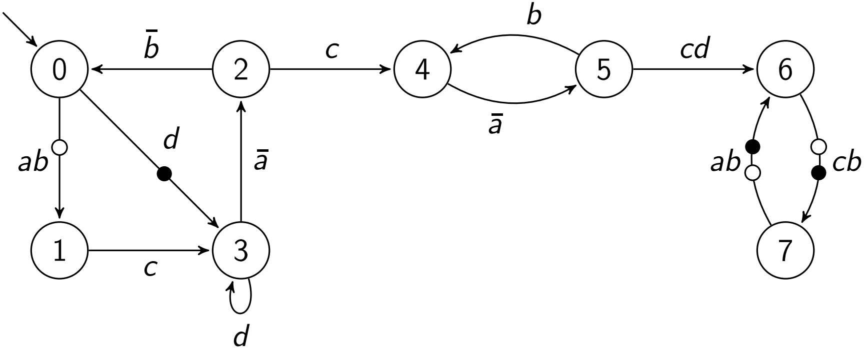
December 9, 2012

Automata-Theoretic Approach to Model Checking



Transition-based Generalized Büchi Automaton

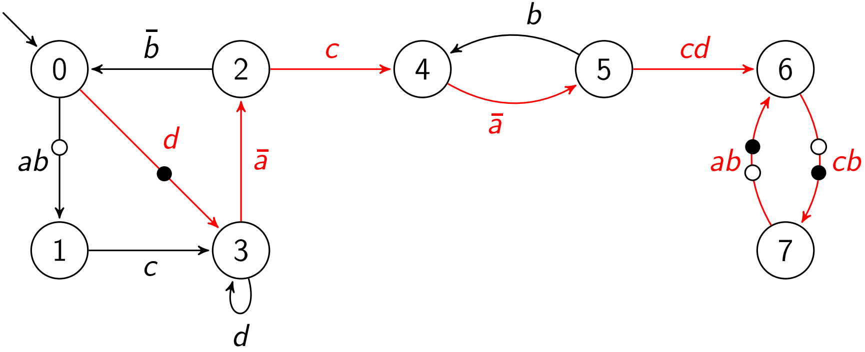
- A TGBA is a tuple, $A = < AP, Q, q_0, \delta, F >$



Accepting runs are infinite sequences visiting infinitely often each acceptance mark.

Transition-based Generalized Büchi Automaton

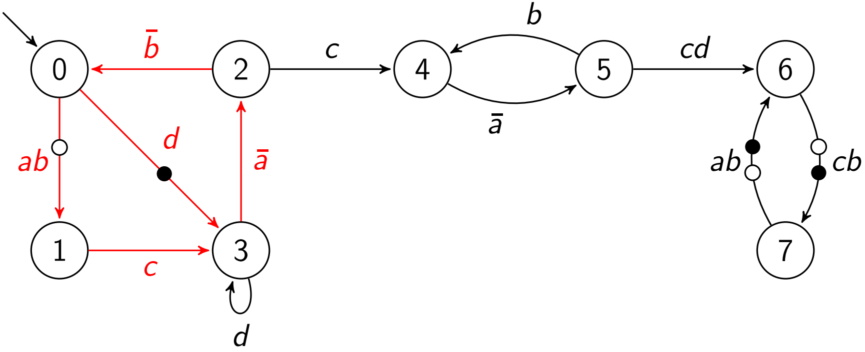
- A TGBA is a tuple, $A =< AP, Q, q_0, \delta, F >$



Accepting runs are infinite sequences visiting infinitely often each acceptance mark.

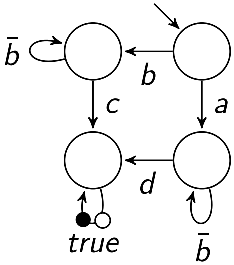
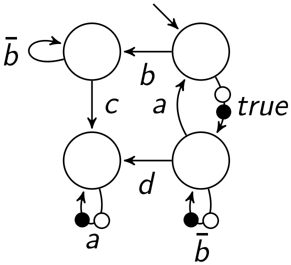
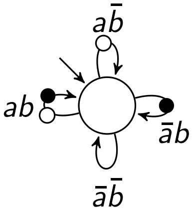
Transition-based Generalized Büchi Automaton

- A TGBA is a tuple, $A =< AP, Q, q_0, \delta, F >$

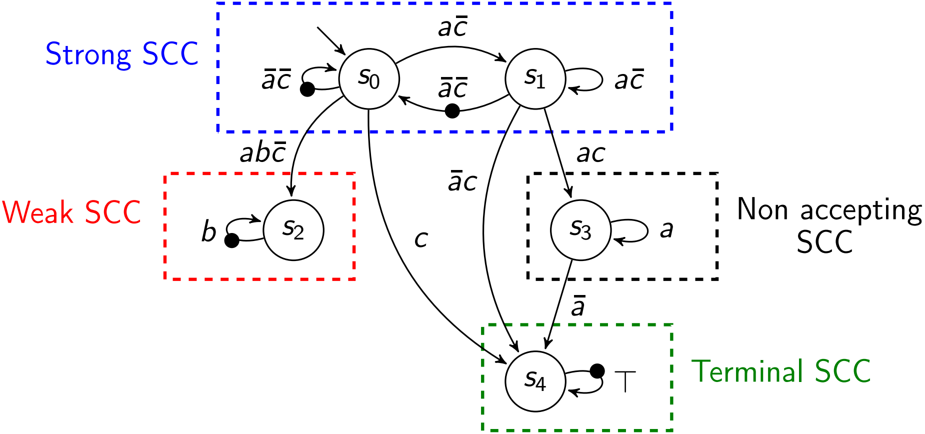


Accepting runs are infinite sequences visiting infinitely often each acceptance mark.

Categories of Automata

Terminal Automaton	Weak Automaton	Strong Automaton
		
Reachability	Simple search	Nested (or iterated) searches

Multiple SCC strengths

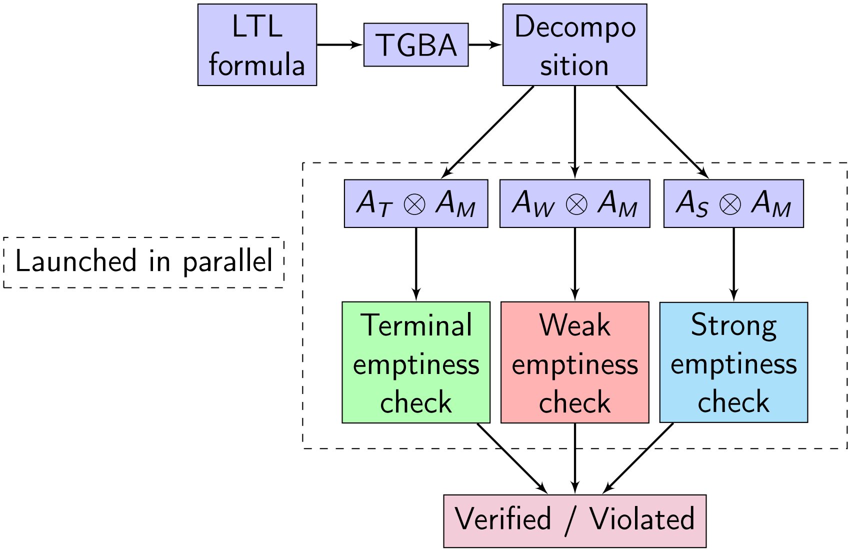


A_ϕ with $\phi = (\mathsf{G}\, a \implies \mathsf{G}\, b)\,\mathsf{W}\, c$

Related Works

- Černá and Pelánek [2003] provide a syntactic characterization of the strength of the property automaton,
- Edelkamp et al. [2004] limit the NDFS to strong SCC and perform specialized emptiness check considering the strength of SCC,
- Barnat [2002] distributes the emptiness check according to the SCC of the property automaton.

Main Idea


LTL formula
TGBA
Decomposition
$A_T \otimes A_M$
$A_W \otimes A_M$
$A_S \otimes A_M$
Launched in parallel
Terminal emptiness check
Weak emptiness check
Strong emptiness check
Verified / Violated

Determining SCC strength: three heuristics

- *syntactic heuristic*: the strength is deduced from the formula labelling states,
- *structural heuristic [Bloem et al., 1999]*: the strength is deduced from the graph structure,
- *inherent heuristic*: the strength is deduced from the elementary cycles of an SCC.

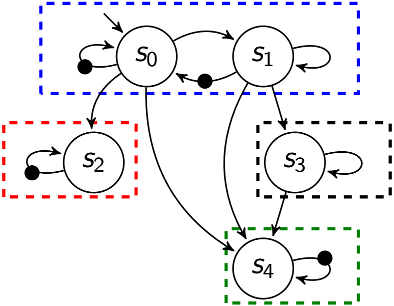
Experiments over 10000 formulae shows (using Spot's translation) that:

- the 3 approaches catch 100% of (inherently) *terminal SCC*
- the syntactic approach catches 87.77% of (inherently) *weak SCC*
- the structural approach catches 99.85% of (inherently) *weak SCC*

Decomposing the Property automaton

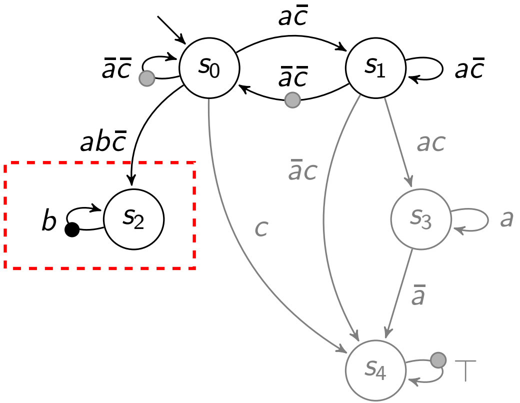
A TGBA A can be decomposed into three derived automata:

- A_T: captures the terminal behaviours of A
- A_W: captures the weak behaviours of A
- A_S: captures the strong behaviours of A



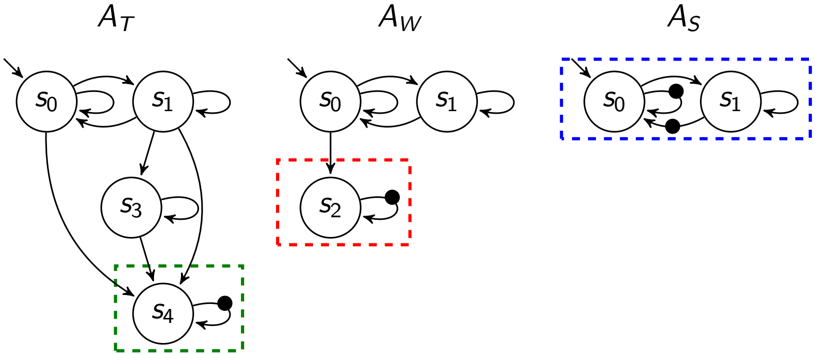
$$\mathscr{L}(A) = \mathscr{L}(A_T) \cup \mathscr{L}(A_W) \cup \mathscr{L}(A_S).$$

Construction for A_W



All acceptance marks are removed and
a single acceptance mark labels all transitions of *weak* SCC.

Exemple of decomposition



For $A_T \otimes A_M$, $A_W \otimes A_M$, specialized emptiness checks are used.
For $A_S \otimes A_M$ we use known algorithms (SE, ELL, Cou, OWCTY)

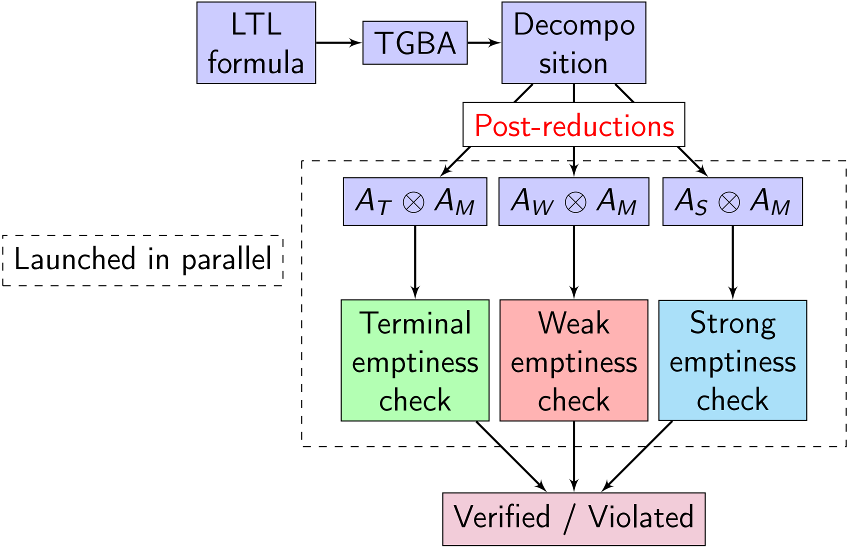
Post-reductions

Decomposed automata have a simpler structure and are therefore easier to reduce.

	no post-reductions		post-reductions	
	states	trans.	states	trans.
A_S	50.66%	37.87%	46.57%	34.85%
A_W	68.71%	51.47%	62.95%	44.77%
A_T	75.27%	63.68%	64.70%	49.28%

Sizes of the automata A_S, A_W, A_T relative to A, with or without the postprocessing applied after decomposition, averaged on 2600 formulas.

Workflow



LTL formula
TGBA
Decomposition
Post-reductions
Launched in parallel
$A_T \otimes A_M$
$A_W \otimes A_M$
$A_S \otimes A_M$
Terminal emptiness check
Weak emptiness check
Strong emptiness check
Verified / Violated

Benchmark

- 13 models from the BEEM benchmark [1]
- 200 random properties for each model
- $A_{\neg\phi} \otimes A_M$ has more than 2000 states

Algorithm		Time	Memory
SE	Schwoon and Esparza [2005]	51.58%	91.38%
ELL	Edelkamp et al. [2004]	49.28%	90.70%
Cou	Couvreur [1999]	61.90%	92.92%
OWCTY	Hojati et al. [1993]	40.43%	83.91%

Gain of the decomposition technique for each algorithm

[1] For more details see `http://anna.fi.muni.cz/models`

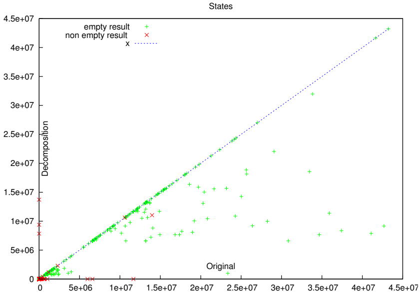
Evaluation of the decomposition technique

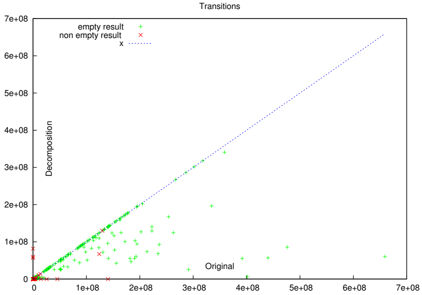
		Empty Product							
		classical				decomposition			
model	algorithm	states	trans.	time	mem	states	trans.	time	mem
at.4	SE	11778840	55765492	112	3034	7620732	30150665	63	2691
84 cases	ELL	11778840	55748407	117	3050	7620732	30150665	63	2688
	Cou	11692421	54326243	95	2913	7542343	28859760	58	2657
	OWCTY			149	3227			75	2841
elevat-	SE	17583328	208607370	273	3622	12709300	106105555	161	3418
-or2.3	ELL	17583328	200251800	287	3639	12709300	106105555	161	3419
64cases	Cou	17144611	171043227	186	3464	12479194	99666774	141	3348
	OWCTY			14	1607			6	1534
		Non Empty Product							
at.4	SE	362202	2384803	4	842	138	181	0	795
93 cases	ELL	362202	2100874	4	861	146	186	0	798
	Cou	362196	2095924	3	837	172	217	0	799
	OWCTY			343	3501			80	2623
elevat-	SE	998871	14729965	15	1023	7967	50455	0	721
or2.3	ELL	998725	13980443	16	1031	7978	50466	0	720
100cases	Cou	984226	9916942	10	986	7975	50464	0	720
	OWCTY			30	2079			6	1172

Time is in seconds, memory is in MB.

Conclusion & results with SPIN

- This approch advantages only with multiple strength automata
- Various ways to implement SCC strength characterization
- Do not depend on the temporal logic (PSL,LTL,...)
- Adaptable to all automata-based model checking approaches
- First results of experiments with SPIN:





Future works

- Mix this approach with other distribution techniques:
  - Holzmann et al. [2011]: multiple independant NDFS,
  - Evangelista et al. [2012]: NDFS sharing memory
- Mix with Partial Order:
  - This approach reduces the number of observable atomic propositions
  - If the original automaton is not X free, can we apply Partial Order over one of the decomposed automaton?

Bibliography I

Barnat, J. (2002). How to Distribute LTL Model-checking Using Decomposition of Negative Claim Automaton. In Bieliková, M., editor, SOFSEM 2002 Student Research Forum, pages 9–14, Milovy, Czech Republic.

Bloem, R., Ravi, K., and Somenzi, F. (1999). Efficient decision procedures for model checking of linear time logic properties. In Proceedings of the Eleventh Conference on Computer Aided Verification (CAV'99), volume 1633 of Lecture Notes in Computer Science, pages 222–235. Springer-Verlag.

Černá, I. and Pelánek, R. (2003). Relating hierarchy of temporal properties to model checking. In Rovan, B. and Vojtáă, P., editors, Proceedings of the 28th International Symposium on Mathematical Foundations of Computer Science (MFCS'03), volume 2747 of Lecture Notes in Computer Science, pages 318–327, Bratislava, Slovak Republic. Springer-Verlag.

Couvreur, J.-M. (1999). On-the-fly verification of temporal logic. In Wing, J. M., Woodcock, J., and Davies, J., editors, Proceedings of the World Congress on Formal Methods in the Development of Computing Systems (FM'99), volume 1708 of Lecture Notes in Computer Science, pages 253–271, Toulouse, France. Springer-Verlag.

Edelkamp, S., Leue, S., and Lluch-Lafuente, A. (2004). Directed explicit-state model checking in the validation of communication protocols. STTT, 5(2–3):247–267.

Evangelista, S., Laarman, A., Petrucci, L., and van de Pol, J. (2012). Improved multi-core nested depth-first search. In ATVA, pages 269–283.

Bibliography II

Hojati, R., Touati, H. J., Kurshan, R. P., and Brayton, R. K. (1993). Efficient omega-regular language containment. In Proceedings of the Fourth International Workshop on Computer Aided Verification, CAV '92, pages 396–409, London, UK, UK. Springer-Verlag.

Holzmann, G. J., Joshi, R., and Groce, A. (2011). Swarm verification techniques. IEEE Trans. Software Eng., 37(6):845–857.

Schwoon, S. and Esparza, J. (2005). A note on on-the-fly verification algorithms. In Halbwachs, N. and Zuck, L., editors, Proceedings of the 11th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'05), volume 3440 of Lecture Notes in Computer Science, Edinburgh, Scotland. Springer.