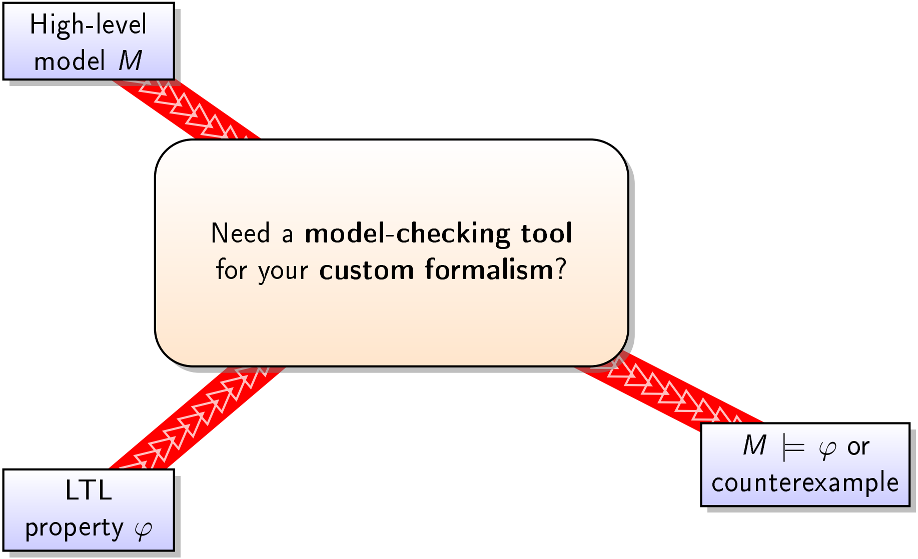
Buiding LTL Model Checkers using Transition-based Generalized Büchi Automata

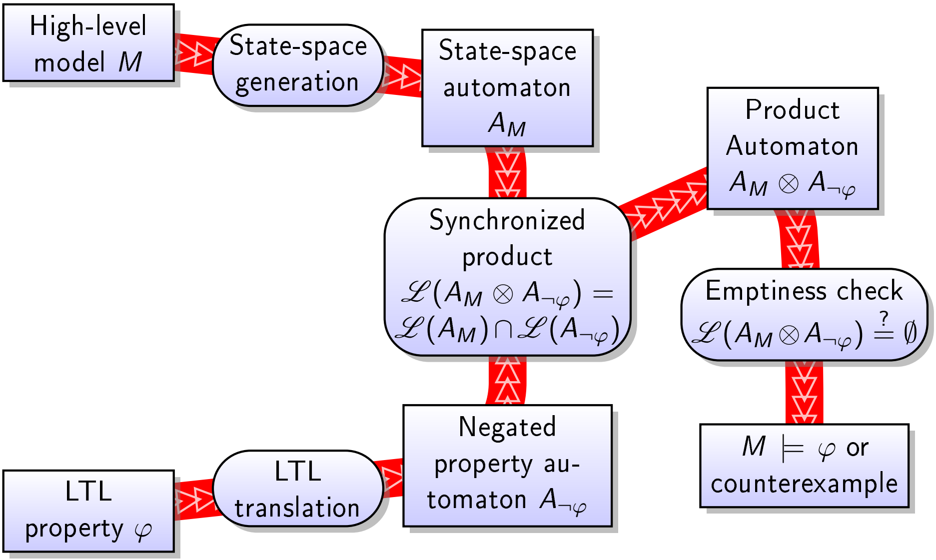
Alexandre Duret-Lutz
<adl@lrde.epita.fr>

SUMo+CompoNet 2011
21 June 2011

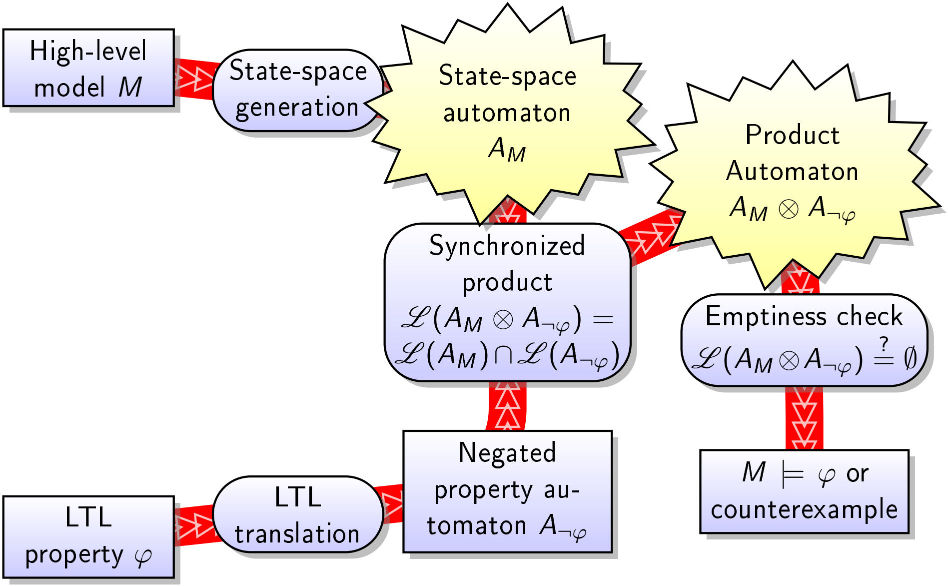
Context


High-level model M
Need a **model-checking tool** for your **custom formalism**?
LTL property φ
$M \models \varphi$ or counterexample

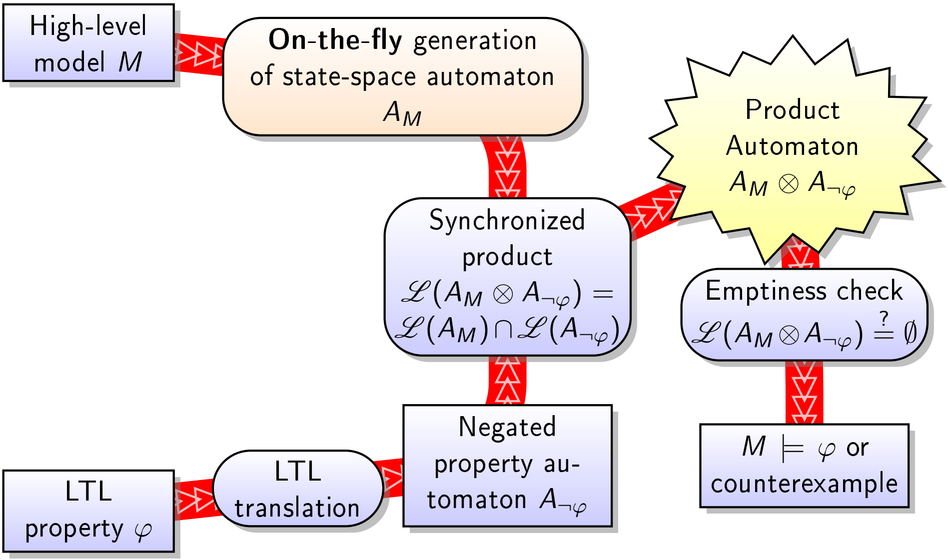
Automata-Theoretic LTL Model Checking



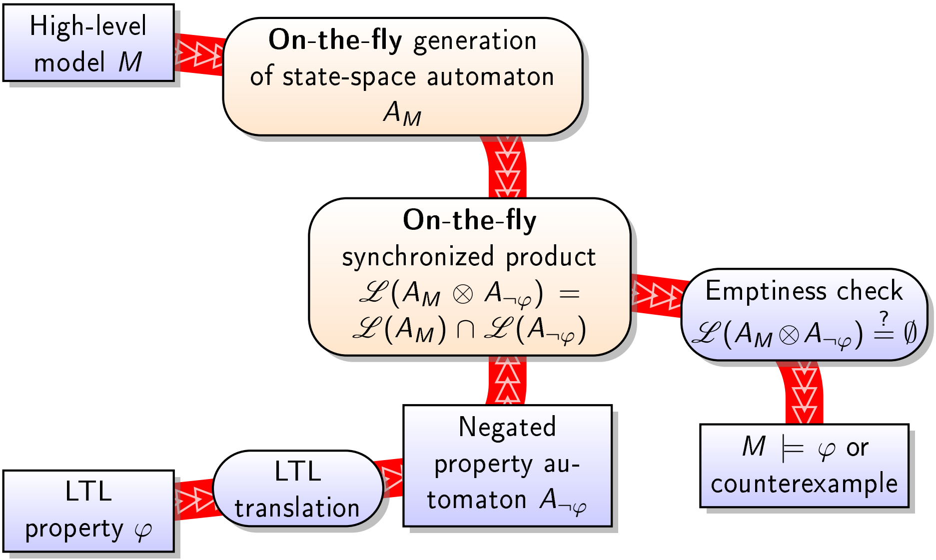
Automata-Theoretic LTL Model Checking



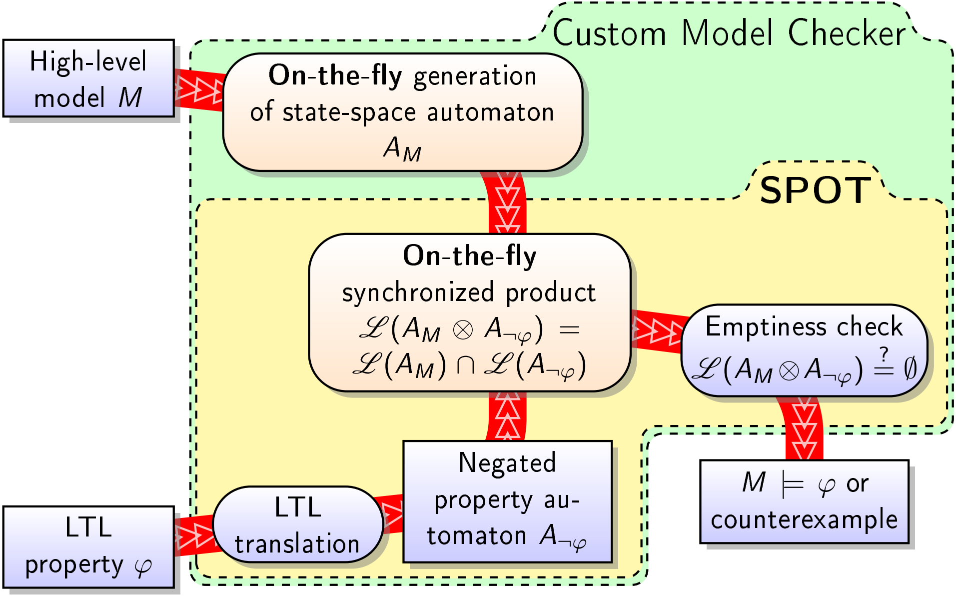
Automata-Theoretic LTL Model Checking



Automata-Theoretic LTL Model Checking



Automata-Theoretic LTL Model Checking



Some (active) Frameworks for LTL Model Checking

JavaPathFinder Java. Model checking of Java bytecode. LTL verification possible using Büchi automata.
`http://babelfish.arc.nasa.gov/trac/jpf/`

DiVinE C++. Büchi automata. Focus on parallel model checking.
`http://divine.fi.muni.cz/`

LTSmin C. Büchi automata. Various input formalisms are supported thanks to a simple state-space interface.
`http://fmt.cs.utwente.nl/tools/ltsmin/`

Spot C++. Transition-based Generalized Automata. An abstract state-space interface, but less input formalisms implemented.
`http://spot.lip6.fr/`

The Spot Library

`http://spot.lip6.fr/`

- A C++ model checking library started in 2003
- Cornerstone: Transition-based Generalized Büchi Automata
- Features several algorithms to combine and build your own model checker
  - 4 algorithms to translates LTL into Büchi automata
  - 5 emptiness-check algorithms (with many variants)
  - 2 Büchi complementation algorithms
  - simplifications for formulas and automata
- We mostly use it to evaluate different algorithms and to develop new model checking techniques
- other people usually use Spot just to translate LTL formulas into Büchi Automata (thanks to Rozier & Vardi)

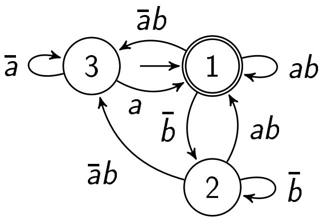
This talk...

1. Why do we prefer to use transition-based and generalized Büchi automata?
2. Why is Spot's translation fast and good?
3. How we reused the Spot architecture to experiment some hybrid (explicit/symbolic) approaches.

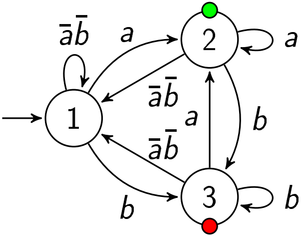
Transition-based Generalized Acceptance Conditions

Different Kinds of Büchi Automata ($\mathbf{G}\,\mathbf{F}\,a \wedge \mathbf{G}\,\mathbf{F}\,b$)

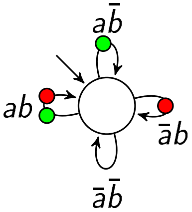
Büchi Automaton



Generalized Büchi Automaton

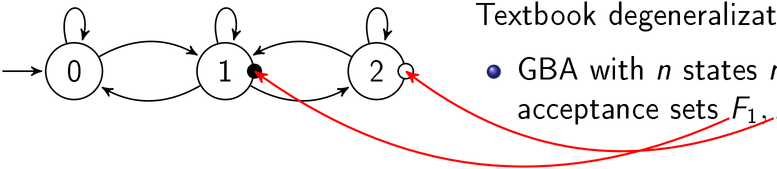


Transition-based Generalized Büchi Automaton



- Same expressive power.
- Converting BA to GBA, or GBA to TGBA, is trivial.
- The opposite direction requires a degeneralization.
- (T)GBA occur naturally when translating LTL.

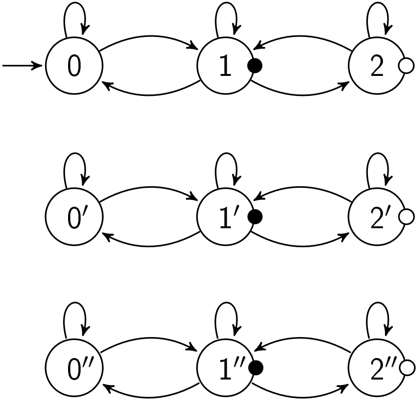
Degeneralization (from GBA to BA)



Textbook degeneralization:

- GBA with n states m acceptance sets $F_1, F_2 \ldots, F_m$

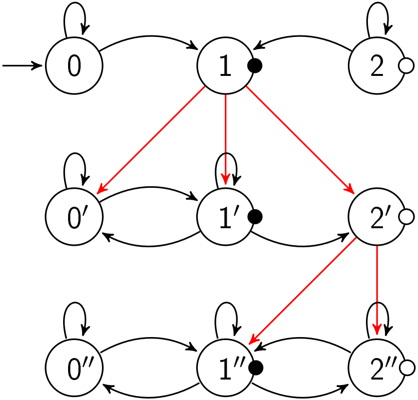
Degeneralization (from GBA to BA)



Textbook degeneralization:

- GBA with n states m acceptance sets $F_1, F_2 \ldots, F_m$
- Duplicate $m + 1$ times

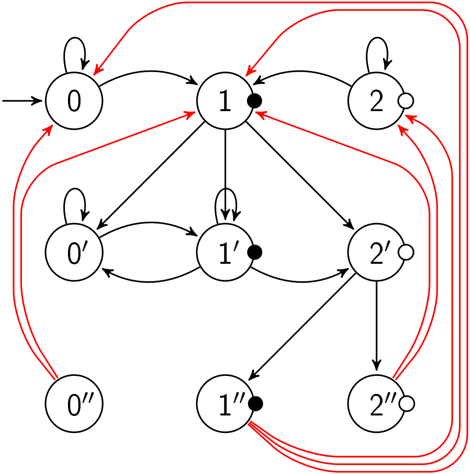
Degeneralization (from GBA to BA)



Textbook degeneralization:

- GBA with n states m acceptance sets $F_1, F_2 \ldots, F_m$
- Duplicate $m+1$ times
- Level i redirects outputs from F_i states to level $i+1$

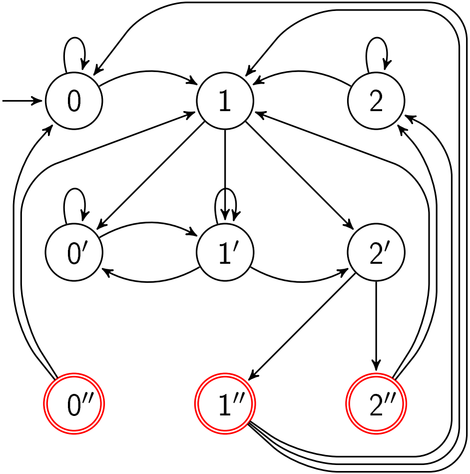
Degeneralization (from GBA to BA)



Textbook degeneralization:

- GBA with n states m acceptance sets $F_1, F_2 \ldots, F_m$
- Duplicate $m+1$ times
- Level i redirects outputs from F_i states to level $i+1$
- Level $m+1$ redirects all outputs to level 1

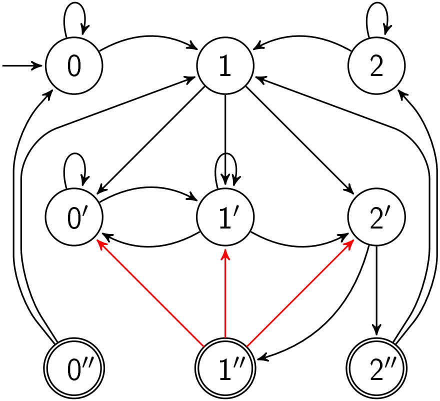
Degeneralization (from GBA to BA)



Textbook degeneralization:

- GBA with n states m acceptance sets $F_1, F_2 \ldots, F_m$
- Duplicate $m+1$ times
- Level i redirects outputs from F_i states to level $i+1$
- Level $m+1$ redirects all outputs to level 1
- Level $m+1$ is accepting

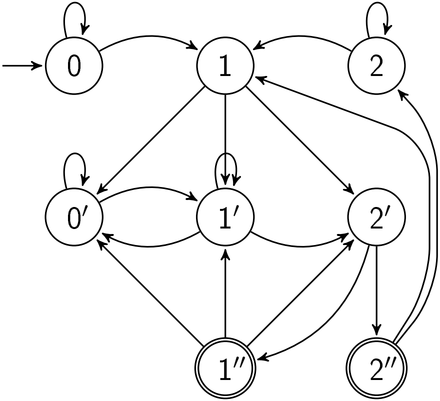
Degeneralization (from GBA to BA)



Usual optimization:

- GBA with n states m acceptance sets $F_1, F_2 \ldots, F_m$
- Duplicate $m+1$ times
- Level i redirects outputs from $F_i \cap F_{i+1} \cap \ldots F_j$ to $j+1$
- Level $m+1$ redirects outputs from $F_1 \cap F_2 \cap \ldots F_j$ to $j+1$
- Level $m+1$ is accepting

Degeneralization (from GBA to BA)

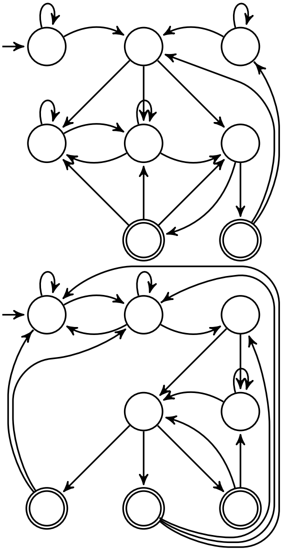


Usual optimization:

- GBA with n states m acceptance sets $F_1, F_2 \ldots, F_m$
- Duplicate $m+1$ times
- Level i redirects outputs from $F_i \cap F_{i+1} \cap \ldots F_j$ to $j+1$
- Level $m+1$ redirects outputs from $F_1 \cap F_2 \cap \ldots F_j$ to $j+1$
- Level $m+1$ is accepting
- Prune unreachable states

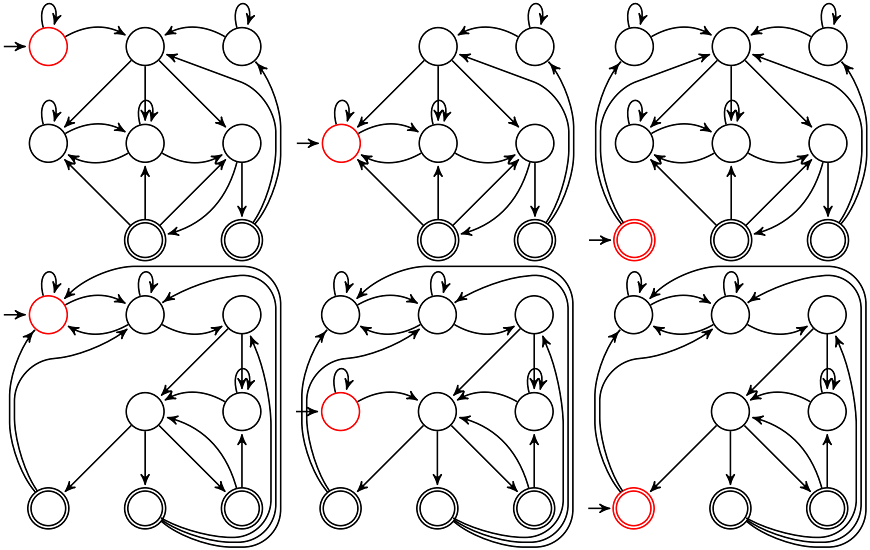
Degeneralization Options

$m!$ orders of acceptance sets



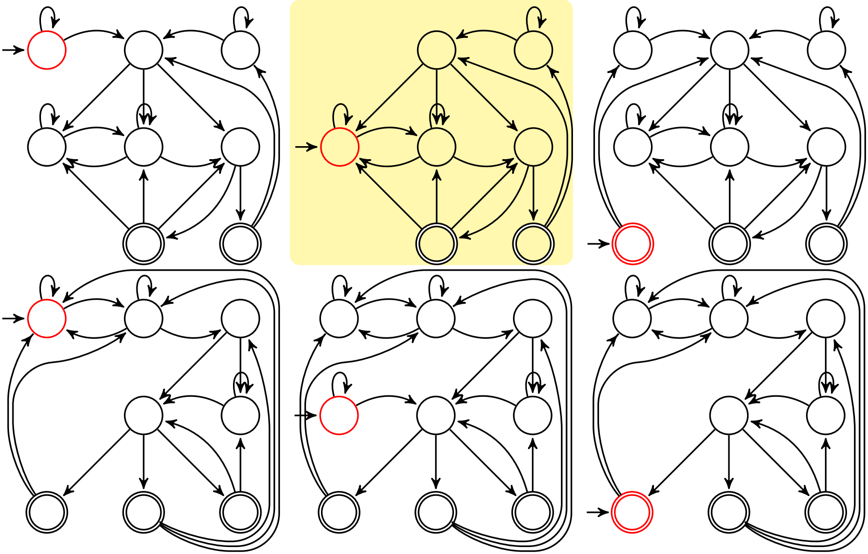
Degeneralization Options

$m!$ orders of acceptance sets and $m+1$ possible initial states



Degeneralization Options

$m!$ orders of acceptance sets and $m+1$ possible initial states



Degeneralization in Spot

- Not needed.
- We have two emptiness-check algorithms that will work with TGBA directly. (Couvreur et al.; 2005)

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

Degeneralization in Spot

- Not needed.
- We have two emptiness-check algorithms that will work with TGBA directly. (Couvreur et al.; 2005)

In case a degeneralization is nonetheless required:

- It is performed on-the-fly, so only one order and one initial state are tried.
- Heuristic to select the order based on the subformulas they are associated to.
- Heuristic to select the initial state based on the acceptance sets of its self loops.

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

Translating LTL into (TG)BA efficiently

Translation of Litterature Formulas

Cumulated sizes of automata for 188 formulas from the litterature

Products with a random state-space of 200 states

		$\Sigma\lvert A_{\neg\varphi}\rvert$		$\Sigma\lvert A_M \otimes A_{\neg\varphi}\rvert$	
		st.	tr.	st.	tr.
BA	Spin 6.1.0 (☠×9)	1 572	7 214	311 032	20 924 268
	LTL2BA 1.1	1 080	3 646	215 717	12 766 425
	Modella 1.5.9 (☢×1)	1 394	4 576	274 881	10 960 064
	Spot 0.7.1	834	2 419	166 579	8 749 162
	Spot 0.7.1 det.	834	2 419	165 677	6 258 605
	Spot 0.7.1 WDBA	773	2 166	153 535	5 657 125
TGBA	Spot 0.7.1	757	2 089	151 185	7 573 811
	Spot 0.7.1 det.	757	2 089	150 445	5 696 034
	Spot 0.7.1 WDBA	705	1 886	140 100	5 156 767

☠ = 15min timeout
☢ = bogus translation

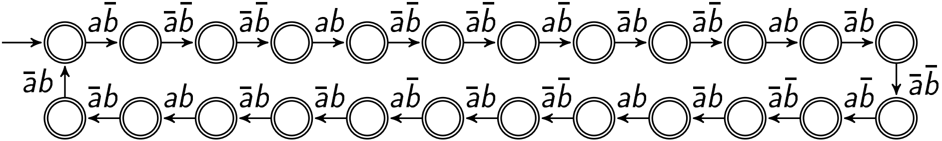
Produce more **det**erministic aut.

WDBA minimization when applicable

Rozier & Vardi's Scalability Experiment (1/2)

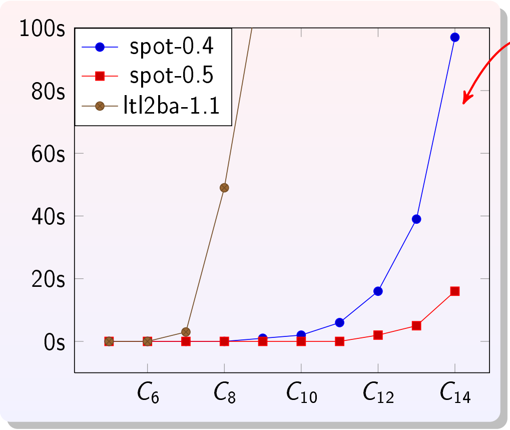
- An LTL formula C_n encoding the values of a n-bit counter.
- E.g. $C_3 = ((a \land (\mathbf{G}(a \rightarrow (\mathbf{X}(\neg a \land \mathbf{X}(\neg a \land \mathbf{X}\, a)))))) \land ((\neg b) \land \mathbf{X}(\neg b \land \mathbf{X}\, \neg b)) \land (\mathbf{G}((a \land \neg b) \rightarrow (\mathbf{X}((\mathbf{X}\,\mathbf{X}\, b) \land (((\neg a) \land (b \rightarrow \mathbf{X}\,\mathbf{X}\,\mathbf{X}\, b) \land ((\neg b) \rightarrow (\mathbf{X}\,\mathbf{X}\,\mathbf{X}\, \neg b)))\, \mathbf{U}\, a))))) \land (\mathbf{G}((a \land b) \rightarrow (\mathbf{X}((\mathbf{X}\,\mathbf{X}\, \neg b) \land ((b \land (\neg a) \land \mathbf{X}\,\mathbf{X}\,\mathbf{X}\, \neg b)\, \mathbf{U} (a \lor ((\neg a) \land (\neg b) \land (\mathbf{X}((\mathbf{X}\,\mathbf{X}\, b) \land (((\neg a) \land (b \rightarrow \mathbf{X}\,\mathbf{X}\,\mathbf{X}\, b) \land ((\neg b) \rightarrow \mathbf{X}\,\mathbf{X}\,\mathbf{X}\, \neg b))\, \mathbf{U}\, a)))))))))))$
- C_3 can by encoded by a $n2^n$-state automaton that accepts

  a: 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 1 0 0 …
  b: 0 0 0 1 0 0 0 1 0 1 1 0 0 0 1 1 0 1 0 1 1 1 1 1 … repeated infinitely.



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

Rozier & Vardi's Scalability Experiment (2/2)



Time to translate C_n into BA

Other explicit translators are off the chart:

- Modella 1.5.9 took 5:47 minutes to compute C_4 and ran out of memory on C_5.

- Spin 6.1.0 took more than 11 hours to translate C_1 into a 33-state automaton with 447 transitions (instead of 2 states and 2 transitions). Many transitions having unsatisfiable guards such as "`((!b) && (a) && (b))`".

All experiments done on an Intel Core2 Q9550 @2.83GHz with 8GB of RAM.

Approaches for Translating LTL to Büchi

Combinatoric approaches Consider the set S of all subformulas of φ. Use 2^S as set of states for the automaton, connect as appropriate. The exponential blowup is in the definition. Easy to implement symbolically using BDD.

Approaches for Translating LTL to Büchi

Combinatoric approaches Consider the set S of all subformulas of φ. Use 2^S as set of states for the automaton, connect as appropriate. The exponential blowup is in the definition. Easy to implement symbolically using BDD.

Compositional approaches An operator like $\varphi \mathbf{U} \psi$ is seen a transducer with two input bands and one output band; with rules to compose transducers. Combinatoric approach in disguise.

Approaches for Translating LTL to Büchi

Combinatoric approaches Consider the set S of all subformulas of φ. Use 2^S as set of states for the automaton, connect as appropriate. The exponential blowup is in the definition. Easy to implement symbolically using BDD.

Compositional approaches An operator like $\varphi \mathbf{U} \psi$ is seen a transducer with two input bands and one output band; with rules to compose transducers. Combinatoric approach in disguise.

Alternating automata There is a linear mapping of LTL formulas to alternating automata. Generalized Büchi automata are obtained by removing alternation. Powerful framework to extend the logic (e.g., to PSL). Exponential blowup during alternation removal.

Approaches for Translating LTL to Büchi

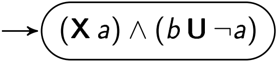
Combinatoric approaches Consider the set S of all subformulas of φ. Use 2^S as set of states for the automaton, connect as appropriate. The exponential blowup is in the definition. Easy to implement symbolically using BDD.

Compositional approaches An operator like $\varphi \mathbf{U} \psi$ is seen a transducer with two input bands and one output band; with rules to compose transducers. Combinatoric approach in disguise.

Alternating automata There is a linear mapping of LTL formulas to alternating automata. Generalized Büchi automata are obtained by removing alternation. Powerful framework to extend the logic (e.g., to PSL). Exponential blowup during alternation removal.

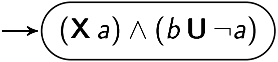
Tableau constructions Rewrite φ as a tree whose branches represent satisfiable conjunctions. Use these branches to deduce possible successors. Simple. Exponential blowup during the constructions of branches.

Tableau Construction for LTL



1. Label the initial state by the formula to translate

J.-M. Couvreur. On-the-fly verification of linear temporal logic. In Proc. of FM'99, vol. 1708 of LNCS, pp. 253–271. Springer



1. Label the initial state by the formula to translate
2. Rewrite each state label φ as

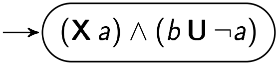
$$\bigvee_i \beta_i \wedge \mathbf{X}\, \psi_i$$

Boolean formula (β_i) — LTL formula (ψ_i)

Since $f\, \mathbf{U}\, g = g \vee (f \wedge \mathbf{X}(f\, \mathbf{U}\, g))$ we have:
$(\mathbf{X}\, a) \wedge (b\, \mathbf{U} \neg a) = (\neg a \wedge \mathbf{X}\, a) \vee (b \wedge (\mathbf{X}\, a) \wedge \mathbf{X}(b\, \mathbf{U} \neg a))$

J.-M. Couvreur. On-the-fly verification of linear temporal logic. In Proc. of FM'99, vol. 1708 of LNCS, pp. 253–271. Springer

Tableau Construction for LTL



1. Label the initial state by the formula to translate
2. Rewrite each state label φ as

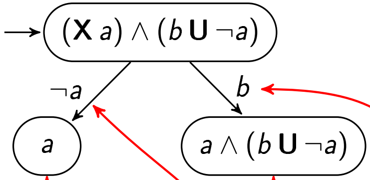
$$\bigvee_i \beta_i \wedge \mathbf{X}\,\psi_i$$

Since $f \,\mathbf{U}\, g = g \vee (f \wedge \mathbf{X}(f \,\mathbf{U}\, g))$ we have:

$$(\mathbf{X}\, a) \wedge (b \,\mathbf{U} \neg a) = (\neg a \wedge \mathbf{X}\, \mathrm{a}) \vee (b \wedge \mathbf{X}(a \wedge (b \,\mathbf{U} \neg a)))$$

J.-M. Couvreur. On-the-fly verification of linear temporal logic. In Proc. of FM'99, vol. 1708 of LNCS, pp. 253–271. Springer

Tableau Construction for LTL



1. Label the initial state by the formula to translate
2. Rewrite each state label φ as

$$\bigvee_i \beta_i \land \mathbf{X}\, \psi_i$$

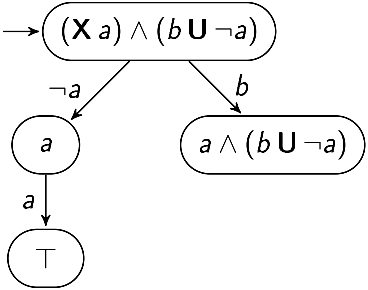
Then connect φ to each ψ_i using β_i as label

Since $f\, \mathbf{U}\, g = g \lor (f \land \mathbf{X}(f\, \mathbf{U}\, g))$ we have:

$(\mathbf{X}\, a) \land (b\, \mathbf{U} \neg a) = (\neg a \land \mathbf{X}\, a) \lor (b \land \mathbf{X}(a \land (b\, \mathbf{U} \neg a)))$

J.-M. Couvreur. On-the-fly verification of linear temporal logic. In Proc. of FM'99, vol. 1708 of LNCS, pp. 253–271. Springer

Tableau Construction for LTL



1. Label the initial state by the formula to translate
2. Rewrite each state label φ as

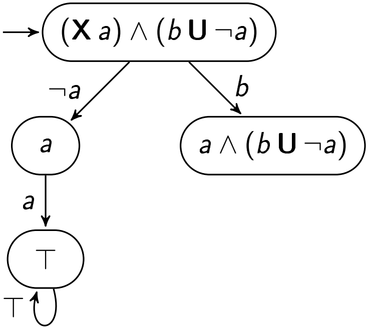
$$\bigvee_i \beta_i \wedge \mathbf{X}\, \psi_i$$

Then connect φ to each ψ_i using β_i as label

Since $f\, \mathbf{U}\, g = g \vee (f \wedge \mathbf{X}(f\, \mathbf{U}\, g))$ we have:
$(\mathbf{X}\, a) \wedge (b\, \mathbf{U} \neg a) = (\neg a \wedge \mathbf{X}\, a) \vee (b \wedge \mathbf{X}(a \wedge (b\, \mathbf{U} \neg a)))$
$a = a \wedge \mathbf{X} \top$

J.-M. Couvreur. On-the-fly verification of linear temporal logic. In Proc. of FM'99, vol. 1708 of LNCS, pp. 253–271. Springer

Tableau Construction for LTL



1. Label the initial state by the formula to translate
2. Rewrite each state label φ as

$$\bigvee_i \; \beta_i \wedge \mathbf{X}\, \psi_i$$

Then connect φ to each ψ_i using β_i as label

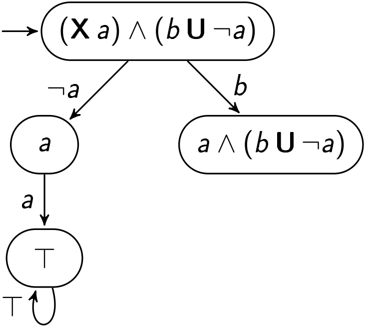
Since $f\, \mathbf{U}\, g = g \vee (f \wedge \mathbf{X}(f\, \mathbf{U}\, g))$ we have:

$(\mathbf{X}\, a) \wedge (b\, \mathbf{U}\, \neg a) = (\neg a \wedge \mathbf{X}\, \mathrm{a}) \vee (b \wedge \mathbf{X}(a \wedge (b\, \mathbf{U}\, \neg a)))$

$a = a \wedge \mathbf{X}\, \top$; $\top = \top \wedge \mathbf{X}\, \top$

J.-M. Couvreur. On-the-fly verification of linear temporal logic. In Proc. of FM'99, vol. 1708 of LNCS, pp. 253–271. Springer

Tableau Construction for LTL



1. Label the initial state by the formula to translate
2. Rewrite each state label φ as

   $$\bigvee_i \; \beta_i \wedge \mathbf{X}\,\psi_i$$

   Then connect φ to each ψ_i using β_i as label

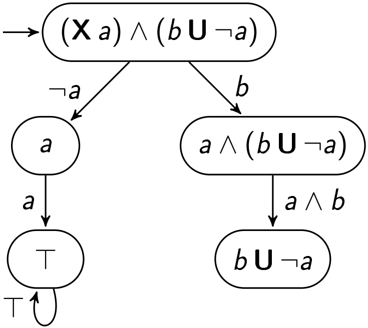
Since $f\,\mathbf{U}\,g = g \vee (f \wedge \mathbf{X}(f\,\mathbf{U}\,g))$ we have:

$(\mathbf{X}\,a) \wedge (b\,\mathbf{U}\,\neg a) = (\neg a \wedge \mathbf{X}\,a) \vee (b \wedge \mathbf{X}(a \wedge (b\,\mathbf{U}\,\neg a)))$

$a = a \wedge \mathbf{X}\,\top$; $\top = \top \wedge \mathbf{X}\,\top$; $a \wedge (b\,\mathbf{U}\,\neg a) = a \wedge (\neg a \vee (b \wedge \mathbf{X}(b\,\mathbf{U}\,\neg a)))$

J.-M. Couvreur. On-the-fly verification of linear temporal logic. In Proc. of FM'99, vol. 1708 of LNCS, pp. 253–271. Springer

Tableau Construction for LTL



1. Label the initial state by the formula to translate
2. Rewrite each state label φ as

$$\bigvee_i \quad \beta_i \wedge \mathbf{X}\, \psi_i$$

Then connect φ to each ψ_i using β_i as label

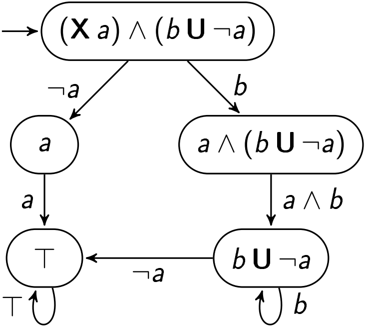
Since $f \,\mathbf{U}\, g = g \vee (f \wedge \mathbf{X}(f \,\mathbf{U}\, g))$ we have:

$(\mathbf{X}\, a) \wedge (b \,\mathbf{U}\, \neg a) = (\neg a \wedge \mathbf{X}\, \mathrm{a}) \vee (b \wedge \mathbf{X}(a \wedge (b \,\mathbf{U}\, \neg a)))$

$a = a \wedge \mathbf{X}\, \top$; $\top = \top \wedge \mathbf{X}\, \top$; $a \wedge (b \,\mathbf{U}\, \neg a) = a \wedge b \wedge \mathbf{X}\, (b \,\mathbf{U}\, \neg a)$

J.-M. Couvreur. On-the-fly verification of linear temporal logic. In Proc. of FM'99, vol. 1708 of LNCS, pp. 253–271. Springer

Tableau Construction for LTL



1. Label the initial state by the formula to translate
2. Rewrite each state label φ as

$$\bigvee_i \beta_i \wedge \mathbf{X}\, \psi_i$$

Then connect φ to each ψ_i using β_i as label

Since $f\, \mathbf{U}\, g = g \vee (f \wedge \mathbf{X}(f\, \mathbf{U}\, g))$ we have:

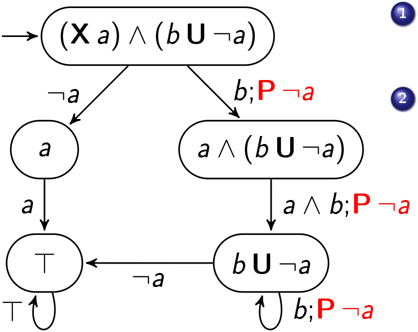
$(\mathbf{X}\, a) \wedge (b\, \mathbf{U}\, \neg a) = (\neg a \wedge \mathbf{X}\, a) \vee (b \wedge \mathbf{X}(a \wedge (b\, \mathbf{U}\, \neg a)))$

$a = a \wedge \mathbf{X}\, \top$; $\top = \top \wedge \mathbf{X}\, \top$; $a \wedge (b\, \mathbf{U}\, \neg a) = a \wedge b \wedge \mathbf{X}\, (b\, \mathbf{U}\, \neg a)$

$b\, \mathbf{U}\, \neg a = (a \wedge \mathbf{X}\, \top) \vee (b \wedge \mathbf{X}\, (b\, \mathbf{U}\, \neg a))$

J.-M. Couvreur. On-the-fly verification of linear temporal logic. In Proc. of FM'99, vol. 1708 of LNCS, pp. 253–271. Springer

Tableau Construction for LTL



1. Label the initial state by the formula to translate
2. Rewrite each state label φ as
$$\bigvee_i \left(\beta_i \land \mathbf{X}\, \psi_i \land \bigwedge_j \mathbf{P}\, \gamma_{ij} \right)$$
Then connect φ to each ψ_i using β_i as label and $\{\mathbf{P}\, \gamma_{ij}\}_j$ as promises

Since $f\, \mathbf{U}\, g = g \lor (f \land \mathbf{X}(f\, \mathbf{U}\, g) \land \mathbf{P}\, g)$ we have:

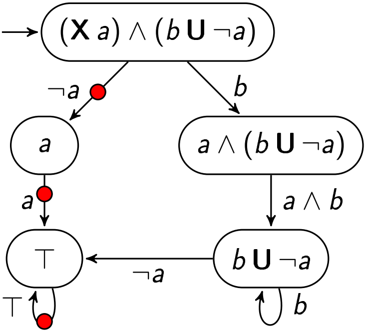
$(\mathbf{X}\, a) \land (b\, \mathbf{U} \neg a) = (\neg a \land \mathbf{X}\, a) \lor (b \land \mathbf{X}(a \land (b\, \mathbf{U} \neg a)) \land \mathbf{P}\, \neg a)$

$a = a \land \mathbf{X}\, \top$; $\top = \top \land \mathbf{X}\, \top$; $a \land (b\, \mathbf{U} \neg a) = a \land b \land \mathbf{X}\, (b\, \mathbf{U} \neg a) \land \mathbf{P}\, \neg a$

$b\, \mathbf{U} \neg a = (a \land \mathbf{X}\, \top) \lor (b \land \mathbf{X}\, (b\, \mathbf{U} \neg a) \land \mathbf{P}\, \neg a)$

J.-M. Couvreur. On-the-fly verification of linear temporal logic. In Proc. of FM'99, vol. 1708 of LNCS, pp. 253–271. Springer

Tableau Construction for LTL



1. Label the initial state by the formula to translate
2. Rewrite each state label φ as

$$\bigvee_i \left(\beta_i \wedge \mathbf{X}\, \psi_i \wedge \bigwedge_j \mathbf{P}\, \gamma_{ij} \right)$$

   Then connect φ to each ψ_i using β_i as label and $\{\mathbf{P}\,\gamma_{ij}\}_j$ as promises
3. Create Büchi acceptance sets complementing each promise

Since $f\,\mathbf{U}\,g = g \vee (f \wedge \mathbf{X}(f\,\mathbf{U}\,g) \wedge \mathbf{P}\,g)$ we have:

$(\mathbf{X}\,a) \wedge (b\,\mathbf{U}\,\neg a) = (\neg a \wedge \mathbf{X}\,a) \vee (b \wedge \mathbf{X}(a \wedge (b\,\mathbf{U}\,\neg a)) \wedge \mathbf{P}\,\neg a)$

$a = a \wedge \mathbf{X}\,\top$; $\top = \top \wedge \mathbf{X}\,\top$; $a \wedge (b\,\mathbf{U}\,\neg a) = a \wedge b \wedge \mathbf{X}\,(b\,\mathbf{U}\,\neg a) \wedge \mathbf{P}\,\neg a$

$b\,\mathbf{U}\,\neg a = (a \wedge \mathbf{X}\,\top) \vee (b \wedge \mathbf{X}\,(b\,\mathbf{U}\,\neg a) \wedge \mathbf{P}\,\neg a)$

J.-M. Couvreur. On-the-fly verification of linear temporal logic. In Proc. of FM'99, vol. 1708 of LNCS, pp. 253–271. Springer

Implementing the Translation with BDDs (1/2)

$$
\begin{aligned}
r(\top) &= \top \\
r(\bot) &= \bot \\
r(p) &= \mathrm{Var}[p] \\
r(\neg p) &= \neg\mathrm{Var}[p] \\
r(f \vee g) &= r(f) \vee r(g) \\
r(f \wedge g) &= r(f) \wedge r(g) \\
r(\neg(f \vee g)) &= r(\neg f) \wedge r(\neg g) \\
r(\neg(f \wedge g)) &= r(\neg f) \vee r(\neg g) \\
r(\mathbf{X}\, f) &= \mathrm{Next}[f] \\
r(\neg\, \mathbf{X}\, f) &= \mathrm{Next}[\neg f] \\
r(f\, \mathbf{U}\, g) &= r(g) \vee (r(f) \wedge \mathrm{Next}[f\, \mathbf{U}\, g] \;\wedge\; \mathrm{P}[g]) \\
r(\neg(f\, \mathbf{U}\, g)) &= r(\neg g) \wedge (r(\neg f) \vee \mathrm{Next}[\neg(f\, \mathbf{U}\, g)])
\end{aligned}
$$

Implementing the Translation using BDDs (2/2)

$$
\begin{aligned}
& r((\mathbf{X}\, a) \wedge (b\, \mathbf{U}\, \neg a)) \\
= {} & r(\mathbf{X}\, a) \wedge r(b\, \mathbf{U}\, \neg a) \\
= {} & \mathrm{Next}[a] \wedge (r(\neg a) \vee (\mathrm{P}[\neg a] \wedge r(b) \wedge \mathrm{Next}[b\, \mathbf{U}\, \neg a])) \\
= {} & \mathrm{Next}[a] \wedge (\neg \mathrm{Var}[a] \vee (\mathrm{P}[\neg a] \wedge \mathrm{Var}[b] \wedge \mathrm{Next}[b\, \mathbf{U}\, \neg a]))
\end{aligned}
$$

Implementing the Translation using BDDs (2/2)

$$
\begin{aligned}
& r((\mathbf{X}\, a) \land (b\, \mathbf{U} \neg a)) \\
&= r(\mathbf{X}\, a) \land r(b\, \mathbf{U} \neg a) \\
&= \mathrm{Next}[a] \land (r(\neg a) \lor (\mathrm{P}[\neg a] \land r(b) \land \mathrm{Next}[b\, \mathbf{U} \neg a])) \\
&= \mathrm{Next}[a] \land (\neg \mathrm{Var}[a] \lor (\mathrm{P}[\neg a] \land \mathrm{Var}[b] \land \mathrm{Next}[b\, \mathbf{U} \neg a]))
\end{aligned}
$$

This BDD is then massaged into an irredundant sum of products:

$$
= (\neg \mathrm{Var}[a] \land \mathrm{Next}[a]) \lor (\mathrm{Var}[b] \land \mathrm{Next}[a] \land \mathrm{Next}[b\, \mathbf{U} \neg a] \land \mathrm{P}[\neg a])
$$

S. Minato. Fast generation of irredundant sum-of-products forms from binary decision diagrams. In Proc. of SASIMI'92, pp. 64–73

Implementing the Translation using BDDs (2/2)

$$
\begin{aligned}
& r((\mathbf{X}\, a) \wedge (b\, \mathbf{U} \neg a)) \\
&= r(\mathbf{X}\, a) \wedge r(b\, \mathbf{U} \neg a) \\
&= \mathrm{Next}[a] \wedge (r(\neg a) \vee (\mathrm{P}[\neg a] \wedge r(b) \wedge \mathrm{Next}[b\, \mathbf{U} \neg a])) \\
&= \mathrm{Next}[a] \wedge (\neg \mathrm{Var}[a] \vee (\mathrm{P}[\neg a] \wedge \mathrm{Var}[b] \wedge \mathrm{Next}[b\, \mathbf{U} \neg a]))
\end{aligned}
$$

This BDD is then massaged into an irredundant sum of products:

$$
= (\neg \mathrm{Var}[a] \wedge \mathrm{Next}[a]) \vee (\mathrm{Var}[b] \wedge \mathrm{Next}[a] \wedge \mathrm{Next}[b\, \mathbf{U} \neg a] \wedge \mathrm{P}[\neg a])
$$

S. Minato. Fast generation of irredundant sum-of-products forms from binary decision diagrams. In Proc. of SASIMI'92, pp. 64–73

BDD Gains (1/2): Automatic Simplifications

- Trivial simplifications of dead branches:
- Less trivial simplifications: e.g. $\neg((a \mathbf{U} b) \lor (b \mathbf{U} c))$.

BDD Gains (1/2): Automatic Simplifications

- Trivial simplifications of dead branches:
  A transition labeled by $a \wedge \neg a$ cannot exist
- Less trivial simplifications: e.g. $\neg((a \mathbf{U} b) \vee (b \mathbf{U} c))$.

BDD Gains (1/2): Automatic Simplifications

- Trivial simplifications of dead branches:
  A transition labeled by $a \land \neg a$ cannot exist
- Less trivial simplifications: e.g. $\neg((a \mathbf{U} b) \lor (b \mathbf{U} c))$.
  Tableau rules give four immediate successors:

$$
\begin{aligned}
& (\neg a) \land (\neg b) \land (\neg c) \\
\lor\ & (\neg a) \land (\neg b) \land (\neg c) \land \mathbf{X} \neg(b \mathbf{U} c) \\
\lor\ & (\neg b) \land (\neg c) \land (\mathbf{X} \neg(a \mathbf{U} b)) \\
\lor\ & (\neg b) \land (\neg c) \land (\mathbf{X} \neg(a \mathbf{U} b)) \land \mathbf{X} \neg(b \mathbf{U} c)
\end{aligned}
$$

  BDD rewritings give two successors:

$$
\begin{aligned}
& \neg \mathrm{Var}[b] \land (\neg \mathrm{Var}[a] \lor \mathrm{Next}[\neg(a \mathbf{U} b)]) \\
\land\ & \neg \mathrm{Var}[c] \land (\neg \mathrm{Var}[b] \lor \mathrm{Next}[\neg(b \mathbf{U} c)])
\end{aligned}
$$

BDD Gains (1/2): Automatic Simplifications

- Trivial simplifications of dead branches:
  A transition labeled by $a \wedge \neg a$ cannot exist
- Less trivial simplifications: e.g. $\neg((a \mathbf{U} b) \vee (b \mathbf{U} c))$.
  Tableau rules give four immediate successors:

$$\begin{aligned} & (\neg a) \wedge (\neg b) \wedge (\neg c) \\ \vee\ & (\neg a) \wedge (\neg b) \wedge (\neg c) \wedge \mathbf{X} \neg(b \mathbf{U} c) \\ \vee\ & (\neg b) \wedge (\neg c) \wedge (\mathbf{X} \neg(a \mathbf{U} b)) \\ \vee\ & (\neg b) \wedge (\neg c) \wedge (\mathbf{X} \neg(a \mathbf{U} b)) \wedge \mathbf{X} \neg(b \mathbf{U} c) \end{aligned}$$

  BDD rewritings give two successors:

$$\begin{aligned} & \neg\mathrm{Var}[b] \wedge (\neg\mathrm{Var}[a] \vee \mathrm{Next}[\neg(a \mathbf{U} b)]) \\ \wedge\ & \neg\mathrm{Var}[c] \wedge (\neg\mathrm{Var}[b] \vee \mathrm{Next}[\neg(b \mathbf{U} c)]) \\ =\ & \neg\mathrm{Var}[b] \wedge \neg\mathrm{Var}[c] \wedge (\neg\mathrm{Var}[a] \vee \mathrm{Next}[\neg(a \mathbf{U} b)]) \end{aligned}$$

BDD Gains (2/2): Automatic Quotienting

BDD rewritings give us an equivalence relation between LTL formulas. For instance we have

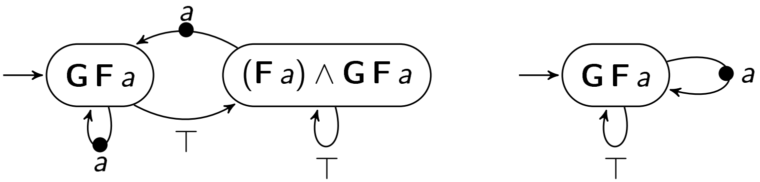
$$
\begin{aligned}
& r(\neg((a \mathbf{U} b) \vee (b \mathbf{U} c))) \\
= & \neg\mathrm{Var}[b] \wedge \neg\mathrm{Var}[c] \wedge (\neg\mathrm{Var}[a] \vee \mathrm{Next}[\neg(a \mathbf{U} b)]) \\
& r(\neg((a \mathbf{U} b) \vee c)) \\
= & \neg\mathrm{Var}[b] \wedge \neg\mathrm{Var}[c] \wedge (\neg\mathrm{Var}[a] \vee \mathrm{Next}[\neg(a \mathbf{U} b)])
\end{aligned}
$$

therefore $\neg((a \mathbf{U} b) \vee (b \mathbf{U} c)) = \neg((a \mathbf{U} b) \vee c)$

Instead of identifying automaton states with LTL formulas, let's identify them with their BDD rewritings. This gives us a quotient automaton for free.

Example of "Automatic Quotienting"

$$r(\mathsf{G}\,\mathsf{F}\,a) = ((\mathrm{Next}[\mathsf{F}\,a] \wedge \mathrm{P}[a]) \vee \mathrm{Var}[a]) \wedge \mathrm{Next}[\mathsf{G}\,\mathsf{F}\,a]$$
$$r(\mathsf{F}\,a \wedge \mathsf{G}\,\mathsf{F}\,a) = ((\mathrm{Next}[\mathsf{F}\,a] \wedge \mathrm{P}[a]) \vee \mathrm{Var}[a]) \wedge \mathrm{Next}[\mathsf{G}\,\mathsf{F}\,a]$$



Could we produce something *more deterministic*?

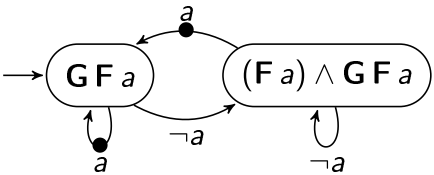
Improving Determinism (1/2)

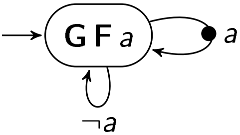
Let's improve the determinism of the previous construction by checking how different variable assignments influence destinations:

$$r(\mathbf{G}\,\mathbf{F}\,a) = ((\mathrm{Next}[\mathbf{F}\,a] \wedge \mathrm{P}[a]) \vee \mathrm{Var}[a]) \wedge \mathrm{Next}[\mathbf{G}\,\mathbf{F}\,a]$$

$$r(\mathbf{G}\,\mathbf{F}\,a) \wedge \mathrm{Var}[a] = \mathrm{Var}[a] \wedge \mathrm{Next}[\mathbf{G}\,\mathbf{F}\,a]$$

$$r(\mathbf{G}\,\mathbf{F}\,a) \wedge \neg\mathrm{Var}[a] = \neg\mathrm{Var}[a] \wedge \mathrm{Next}[\mathbf{F}\,a] \wedge \mathrm{P}[a] \wedge \mathrm{Next}[\mathbf{G}\,\mathbf{F}\,a]$$





Improving Determinism (2/2)

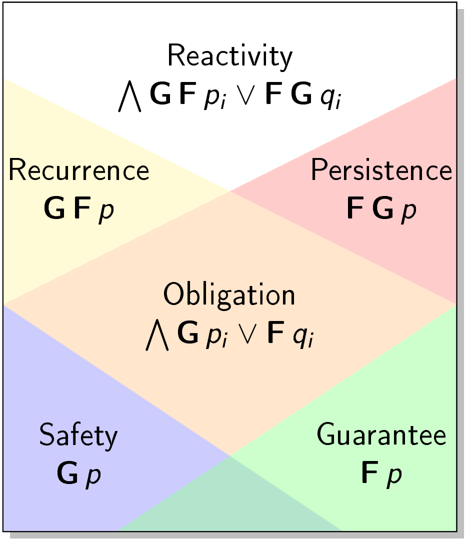
If we have n atomic properties ($\mathrm{Var}[a]$, $\mathrm{Var}[b]$, ...), there are 2^n assignments to consider.

However:

- The structure of the BDD helps to ignore useless assignments.
- Assignments share many branches in the BDD, so the cache will help.
- Small number of atomic properties in practice
- Runtime roughly equivalent on random formulas

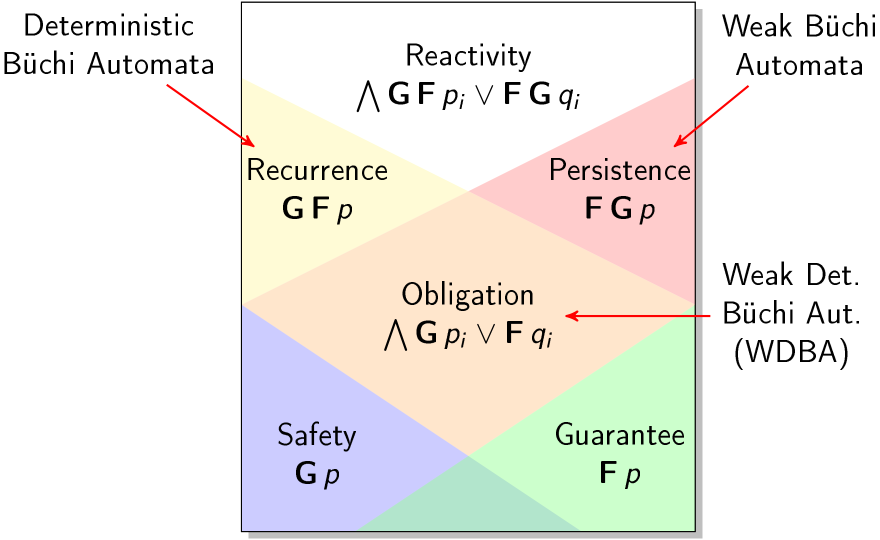
Experiment on 188 LTL formulas from the literature applied to random graphs: 0.4% less states, **25% less transitions** in the product.

Temporal Hierarchy



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

Temporal Hierarchy



I. Černá and R. Pelánek. Relating hierarchy of temporal properties to model checking. In Proc. of MFCS'03, vol. 2747 of LNCS, pp. 318–327. Springer

Dealing with Obligation Properties

- 118 out of the 188 formulas (62%) are obligations properties
- WDBA can be minimized in a same way as DFAs (Löding; 2001)
- Dax et al. (2007) show how to minimize any automaton that can be expressed as a WDBA. In Spot this algorithm takes a TGBA and output a WDBA when the minimization is applicable
- Although converting to a minimal WDBA incurs a determinization, the number of states seldom increases

C. Löding. Efficient minimization of deterministic weak ω-automata. Information Processing Letters, 79(3):105–109, 2001

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

Translation of Litterature Formulas

Cumulated sizes of automata for 188 formulas from the litterature

Products with a random state-space of 200 states

		$\Sigma\lvert A_{\neg\varphi}\rvert$ st.	$\Sigma\lvert A_{\neg\varphi}\rvert$ tr.	$\Sigma\lvert A_M \otimes A_{\neg\varphi}\rvert$ st.	$\Sigma\lvert A_M \otimes A_{\neg\varphi}\rvert$ tr.
BA	Spin 6.1.0 (☠×9)	1 572	7 214	311 032	20 924 268
	LTL2BA 1.1	1 080	3 646	215 717	12 766 425
	Modella 1.5.9 (☢×1)	1 394	4 576	274 881	10 960 064
	Spot 0.7.1	834	2 419	166 579	8 749 162
	Spot 0.7.1 det.	834	2 419	165 677	6 258 605
	Spot 0.7.1 WDBA	773	2 166	153 535	5 657 125
TGBA	Spot 0.7.1	757	2 089	151 185	7 573 811
	Spot 0.7.1 det.	757	2 089	150 445	5 696 034
	Spot 0.7.1 WDBA	705	1 886	140 100	5 156 767

☠ = 15min timeout

☢ = bogus translation

Produce more **det**erministic aut.

WDBA minimization when applicable

Cross-Comparison of Translations

One-to-one comparisons of who yielded less transitions in the products $A_M \otimes A_{\neg\varphi}$ for 188 LTL formulas φ from the litterature.

			(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
BA	Spin 6.1.0	(1)	0	14	34	10	0	1	0	0	1
	LTL2BA 1.1	(2)	113	0	83	9	0	6	2	0	5
	Modella 1.5.9	(3)	150	96	0	47	4	5	35	1	2
	Spot 0.7.1	(4)	160	137	105	0	0	14	0	0	14
	Spot 0.7.1 det.	(5)	176	170	125	107	0	20	94	0	20
	Spot 0.7.1 WDBA	(6)	175	165	135	122	44	0	110	44	0
TGBA	Spot 0.7.1	(7)	170	153	117	36	15	28	0	0	14
	Spot 0.7.1 det.	(8)	176	170	128	109	36	51	107	0	20
	Spot 0.7.1 WDBA	(9)	175	166	138	124	75	33	122	44	0

E.g.: translation (9) strictly dominates translation (3) in 138 cases out of 188.

The converse occurs twice.

Cross-Comparison of Translations

One-to-one comparisons of who yielded less transitions in the products $A_M \otimes A_{\neg\varphi}$ for 188 LTL formulas φ from the litterature.

			(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
BA	Spin 6.1.0	(1)	0	14	34	10	**0**	1	0	**0**	1
	LTL2BA 1.1	(2)	113	0	83	9	**0**	6	2	**0**	5
	Modella 1.5.9	(3)	150	96	0	47	**4**	5	35	**1**	2
	Spot 0.7.1	(4)	160	137	105	0	**0**	14	0	**0**	14
	Spot 0.7.1 det.	(5)	176	170	125	107	0	20	94	0	20
	Spot 0.7.1 WDBA	(6)	175	165	135	122	44	0	110	44	0
TGBA	Spot 0.7.1	(7)	170	153	117	36	15	28	0	0	14
	Spot 0.7.1 det.	(8)	176	170	128	109	36	51	107	0	20
	Spot 0.7.1 WDBA	(9)	175	166	138	124	75	33	122	44	0

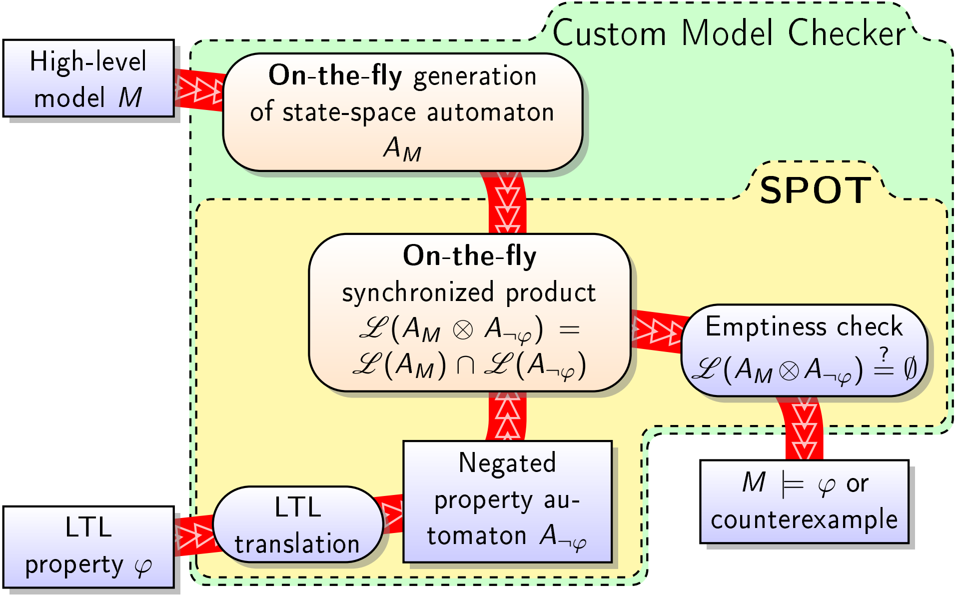
Cross-Comparison of Translations

One-to-one comparisons of who yielded less transitions in the products $A_M \otimes A_{\neg\varphi}$ for 188 LTL formulas φ from the litterature.

			(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)
BA	Spin 6.1.0	(1)	0	14	34	10	0	1	0	0	1
	LTL2BA 1.1	(2)	113	0	83	9	0	6	2	0	5
	Modella 1.5.9	(3)	150	96	0	47	4	5	35	1	2
	Spot 0.7.1	(4)	160	137	105	0	0	14	0	0	14
	Spot 0.7.1 det.	(5)	176	170	125	107	0	**20**	94	0	**20**
	Spot 0.7.1 WDBA	(6)	175	165	135	122	**44**	0	110	**44**	0
TGBA	Spot 0.7.1	(7)	170	153	117	36	15	28	0	0	14
	Spot 0.7.1 det.	(8)	176	170	128	109	36	51	107	0	**20**
	Spot 0.7.1 WDBA	(9)	175	166	138	124	75	33	122	**44**	0

Hybrid State-Space Generation

Automata-Theoretic LTL Model Checking



Third-Party Interfaces

- CheckPN (LIP6, Paris; FR)
- GreatSPN (Univ. Turino; IT)
- DiVinE (Masaryk Univ., Brno; CZ) modified by the LTSmin team (Univ. Twente; NL)
- ITS Tools (LIP6, Paris; FR)

Hybrid Approaches with ITS Tools

Common to the three approaches:

- Using SDDs to represent sets of states.
- Build a graph of sets of states, and model check this graph.

Symbolic Observation Graph (SOG) Stuttering-insensitive prop.

- Gather all states whose valuation do not change the propositions observed by the property.

Symbolic Observation Product (SOP) Stuttering-insensitive prop.

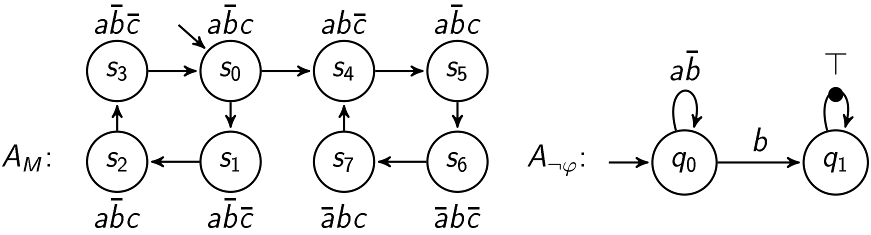
- Gather all states whose valuation do not change the propositions observed by the property automaton **at the current point**.

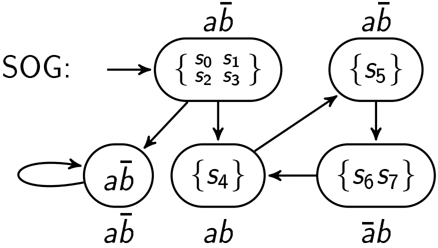
Self-Loop Aggregating Product (SLAP) Full LTL

- Gather all states compatible with the self-loops of the property automaton **at the current point**.

Joint work with D. Poitrenaud, Y. Thierry-Mieg, and K. Klai.
http://move.lip6.fr/software/DDD/ltl_bench.html

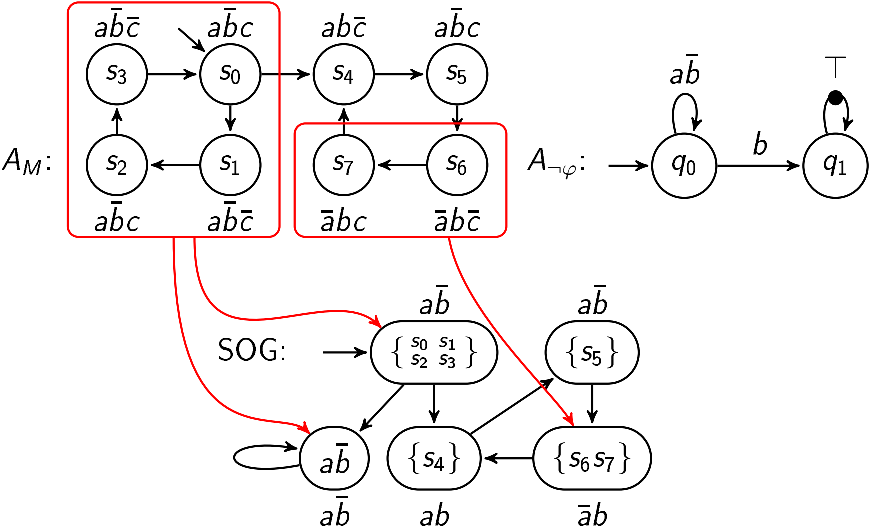
Symbolic Observation Graph





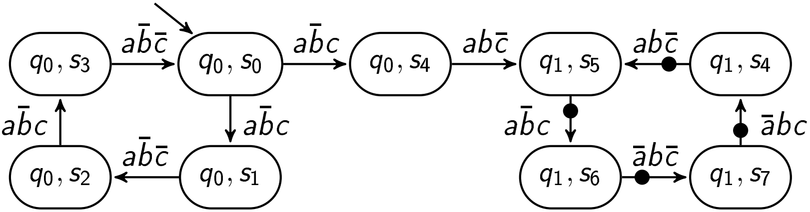
K. Klai and D. Poitrenaud. MC-SOG: An LTL model checker based on symbolic observation graphs. In Proc.Proc. of Petri Nets'08, vol. 5062 of LNCS, pp. 288–306. Springer

Symbolic Observation Graph

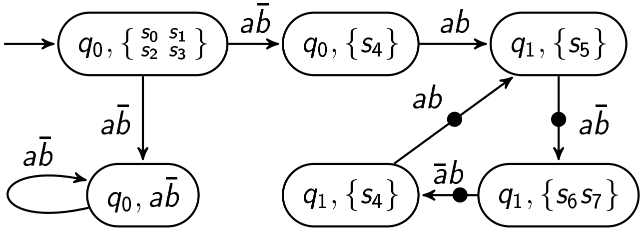


K. Klai and D. Poitrenaud. MC-SOG: An LTL model checker based on symbolic observation graphs. In Proc.Proc. of Petri Nets'08, vol. 5062 of LNCS, pp. 288–306. Springer

$A_M \otimes A_{\neg\varphi}$:

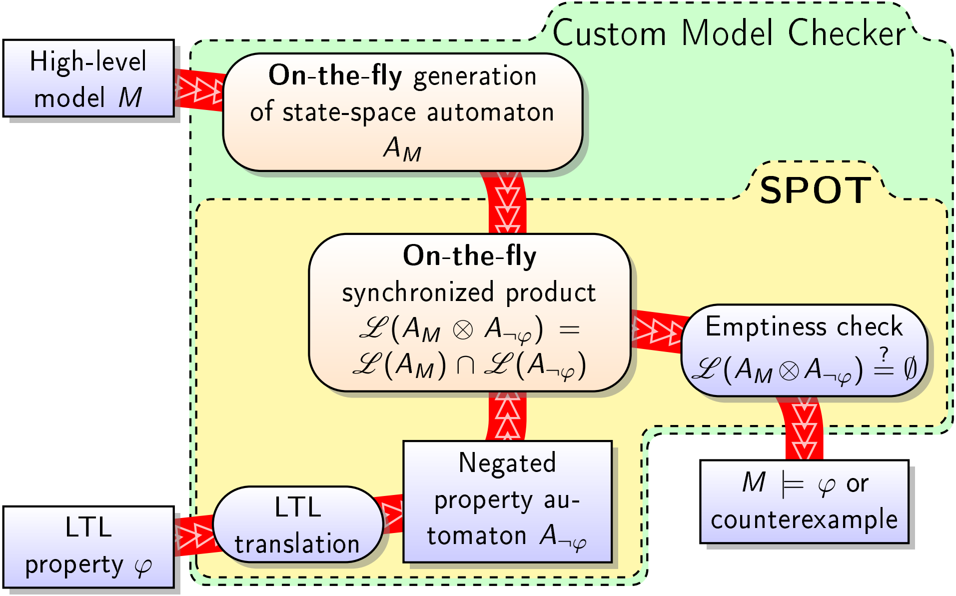


$SOG \otimes A_{\neg\varphi}$:

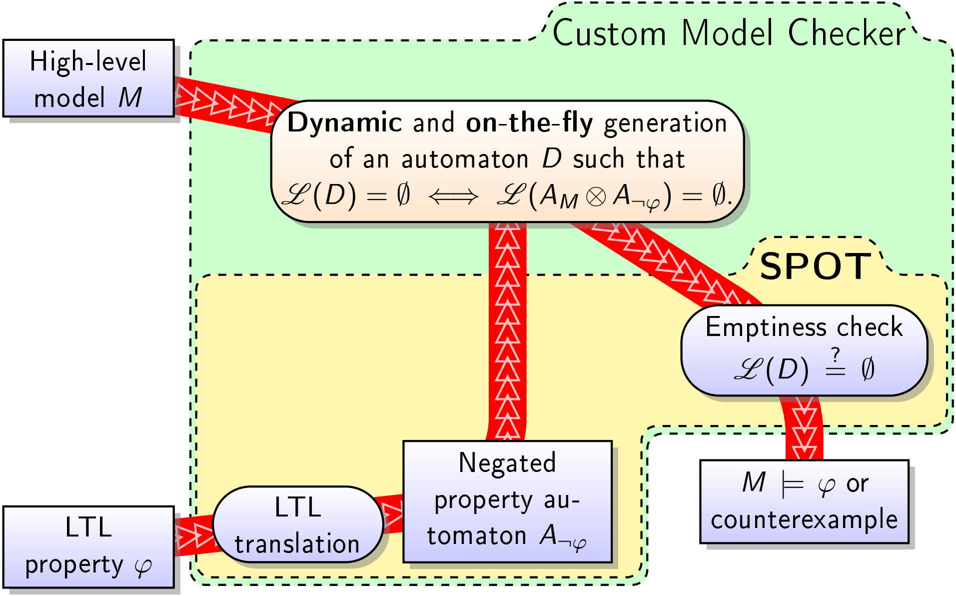


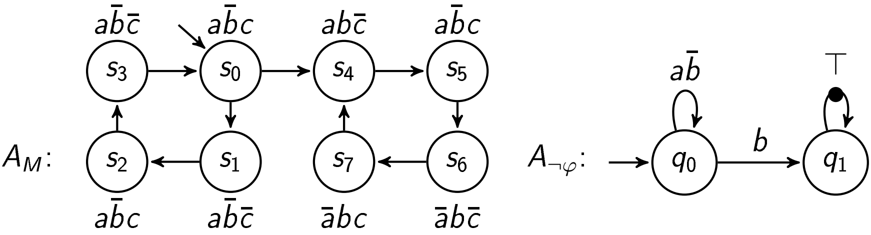
K. Klai and D. Poitrenaud. MC-SOG: An LTL model checker based on symbolic observation graphs. In Proc.Proc. of Petri Nets'08, vol. 5062 of LNCS, pp. 288–306. Springer

Automata-Theoretic LTL Model Checking

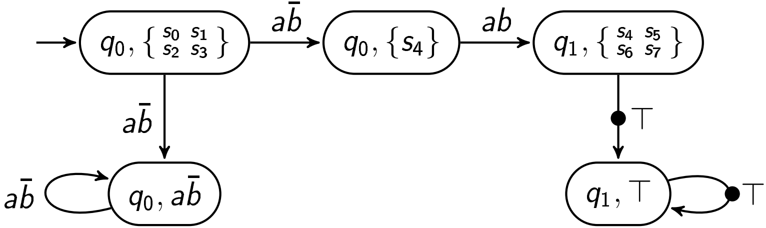


Automata-Theoretic LTL Model Checking



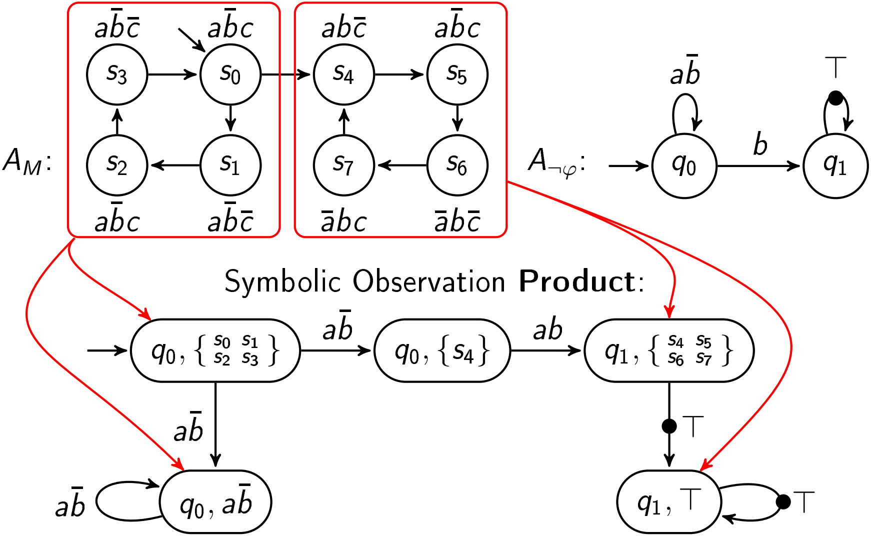


Symbolic Observation **Product**:



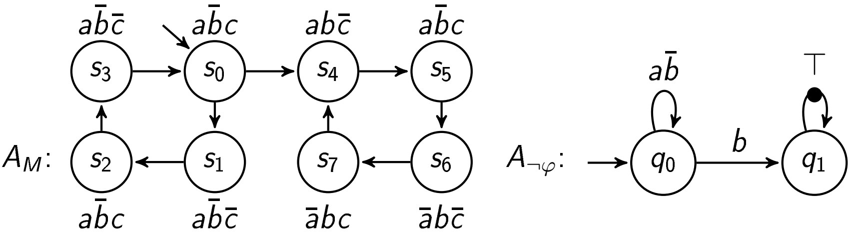
Joint work with D. Poitrenaud, Y. Thierry-Mieg, and K. Klai.
http://move.lip6.fr/software/DDD/ltl_bench.html

Symbolic Observation Product

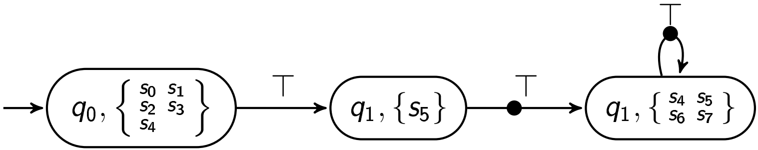


Joint work with D. Poitrenaud, Y. Thierry-Mieg, and K. Klai.
http://move.lip6.fr/software/DDD/ltl_bench.html

Self-Loop Aggregation Product



Self-Loop Aggregation **Product**:



Joint work with D. Poitrenaud, Y. Thierry-Mieg, and K. Klai.
http://move.lip6.fr/software/DDD/ltl_bench.html

Cross-Comparison of Aggregation Methods

One-to-one comparisons of who yielded less transitions in the "products", on 3093 experiments (toy models with random verified formulas)

	BCZ	SOG	SOP	SLAP
BCZ	0	0	0	8
SOG	2825	0	8	142
SOP	2908	692	0	165
SLAP	3032	2931	2909	0

E.g.: SLAP strictly dominates SOG in 2931 cases out of 3093

... the converse occurs 142 times.

A. Biere, E. M. Clarke, and Y. Zhu. Multiple state and single state tableaux for combining local and global model checking. In Correct System Design, vol. 1710 of LNCS, pp. 163–179. Springer, 1999

Conclusion

TGBA

- They are small
- They represent weak fairness properties ($\bigwedge \mathbf{G}\,\mathbf{F}\, p_i$) naturally

LTL translation

- It is efficient and simple to implement
- Using BDD is important (for speed and determinism)
- Buiding WDBA when possible is often a good idea (not always)
- Try it on-line: `http://spot.lip6.fr/ltl2tgba.html`

New products

- Two new hybrid approaches: SOP and SLAP
- The Spot architecture allows easy experimentation of such ideas

What's Cooking

- Support for PSL (Property Specification Language)
  Essentially LTL + rational operators.
  `{ 1[*]; init; busy[=2]; done } []-> (!init U bell)`
- Simulation-based reductions (for TGBA)
- Testing Automata