Contributions to LTL and ω-Automata for Model Checking

Alexandre Duret-Lutz
LRDE/EPITA

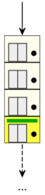
10 February 2017

Javier Esparza	Technische Universität München	reviewer
Radu Mateescu	INRIA Grenoble	reviewer
Moshe Y. Vardi	Rice University, Houston, Texas	reviewer
Rüdiger Ehlers	Universität Bremen	examiner
Stephan Merz	INRIA Nancy & LORIA	examiner
Jaco van de Pol	University of Twente	examiner
Fabrice Kordon	Univ. Pierre & Marie Curie, Paris	examiner

Live demo

```
In [62]: @interact(n=IntSlider(1, 1, 150, 5))
         def uncover_state_space(n):
             return lift_display(k, n)
```

n 1



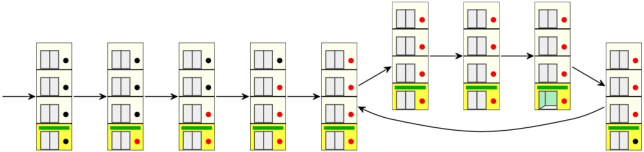

```
In [63]: spot.stats_reachable(k).states
```

```
Out[63]: 352
```

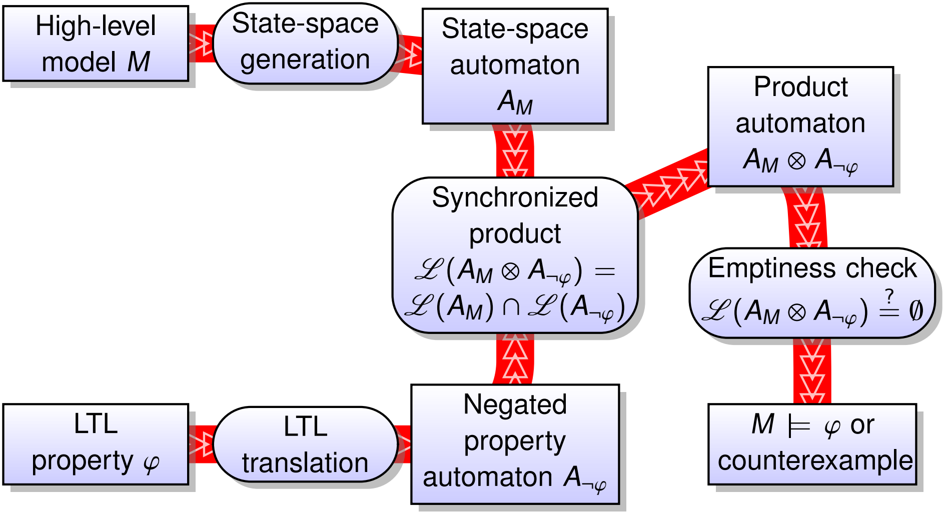
```
In [64]: def model_check(model, f):
             f = spot.formula(f)
             ss = model.kripke(spot.atomic_prop_collect(f))
             nf = spot.formula_Not(f).translate()
             return ss.intersecting_run(nf)
```

```
In [65]: lift_display(model_check(m, 'G("req[1]" -> F("p==1" && "cabin.open"))'))
```

Out[65]:

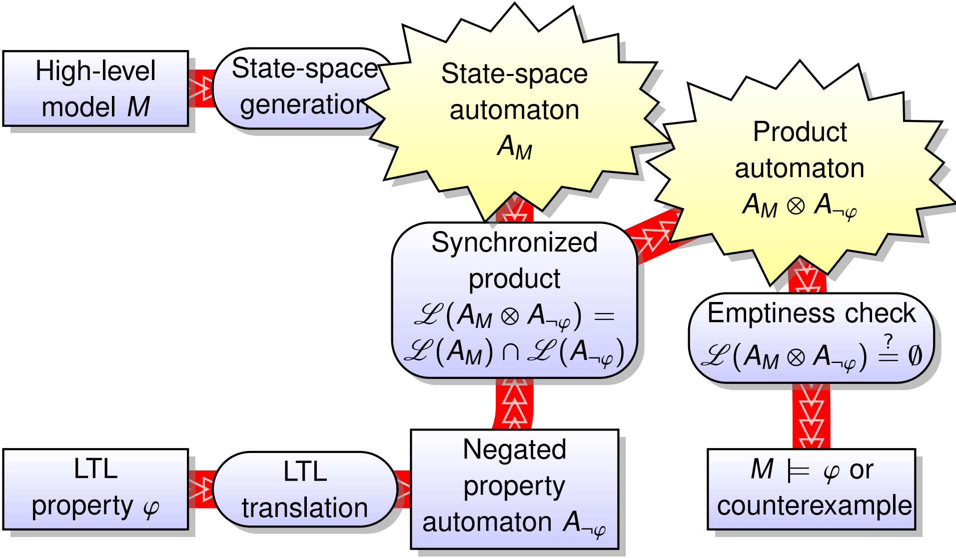


Automata-Theoretic LTL Model Checking



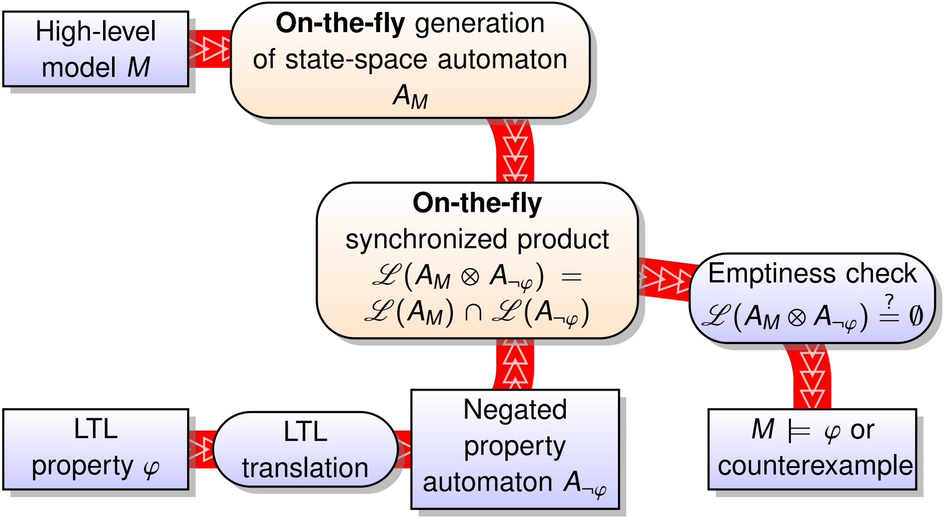
M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. *LICS'86*

Automata-Theoretic LTL Model Checking



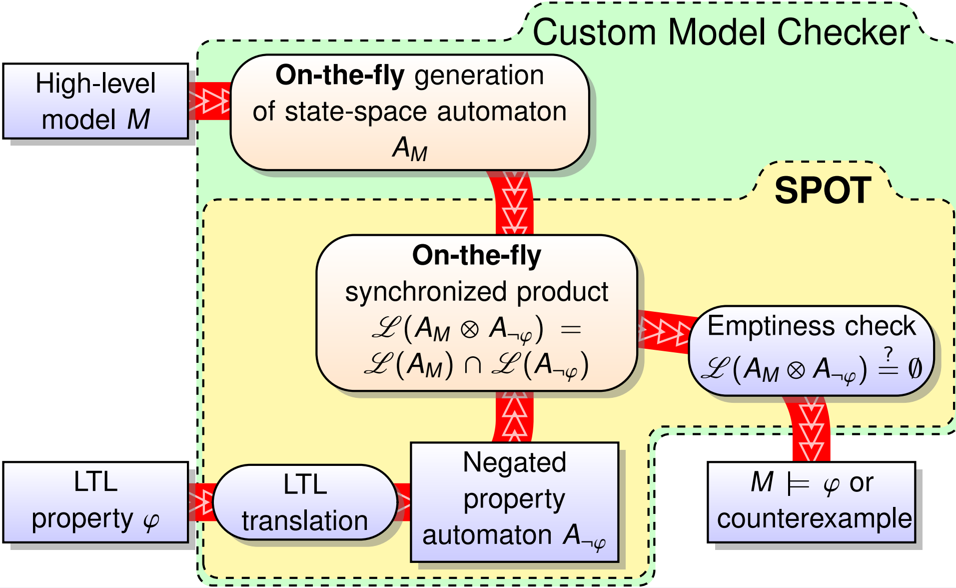
M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. *LICS'86*

Automata-Theoretic LTL Model Checking



M. Y. Vardi and P. Wolper. An automata-theoretic approach to automatic program verification. *LICS'86*

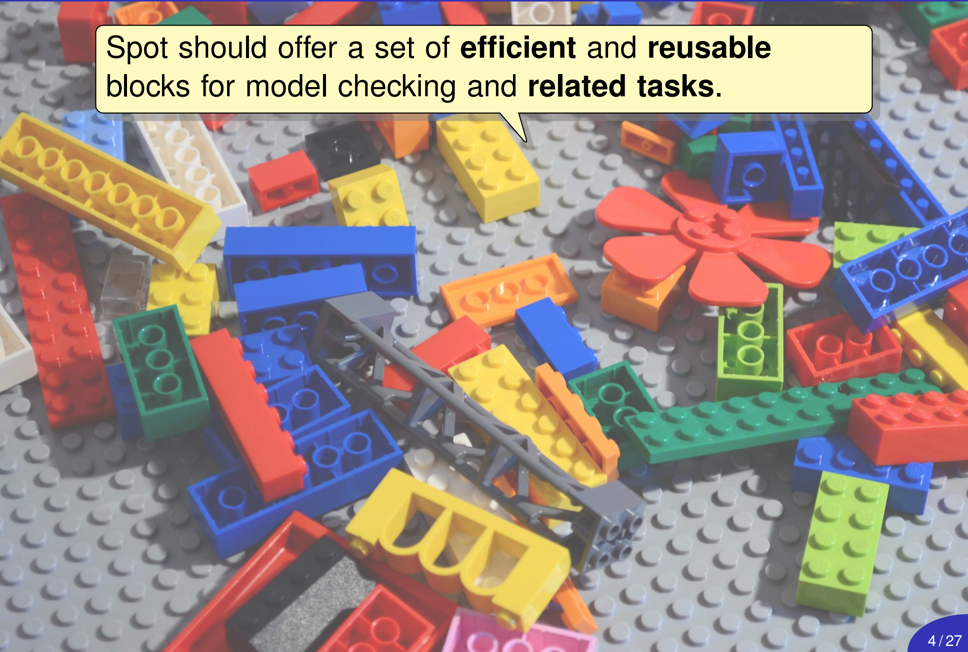
Automata-Theoretic LTL Model Checking



A. Duret-Lutz and D. Poitrenaud. SPOT: an Extensible Model Checking Library using Transition-based Generalized Büchi Automata. *MASCOTS'04*

Motivation: Supporting Research

Spot should offer a set of **efficient** and **reusable** blocks for model checking and **related tasks**.



Motivation: Supporting Research

Spot should offer a set of **efficient** and **reusable** blocks for model checking and **related tasks**.

Efficient:
- Implement state-of-the-art algorithms
- Improve them
- Propose new algorithms

Reusable:
- Multiple interfaces (C++/Python/Shell)
- Documented
- Tested

Related tasks:
- LTL and ω-automata toolbox
- Glue between third-party tools

Motivation: Supporting Research

Spot should offer a set of **efficient** and **reusable** blocks for model checking and **related tasks**.

Efficient:

- Implement state-of-the-art algorithms
- Improve them
- Propose new algorithms

Research

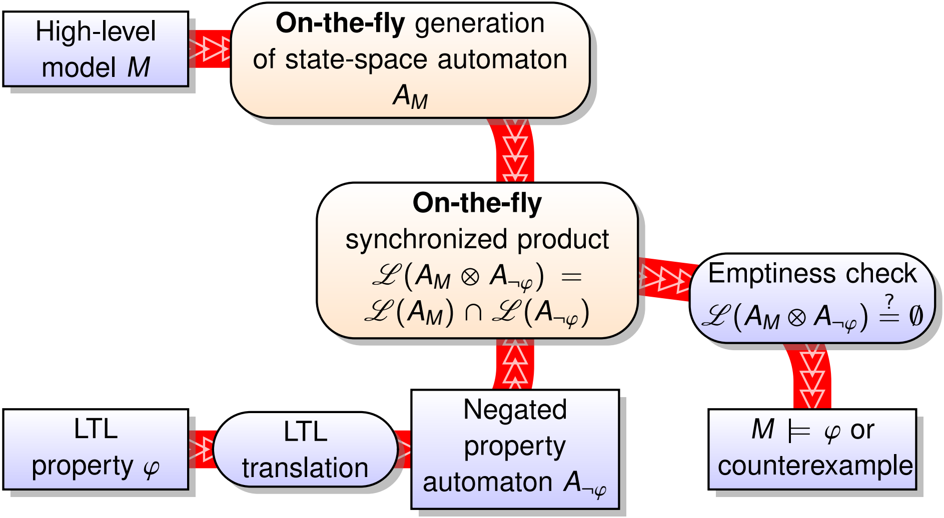
Reusable:

- Multiple interfaces (C++/Python/Shell)
- Documented
- Tested

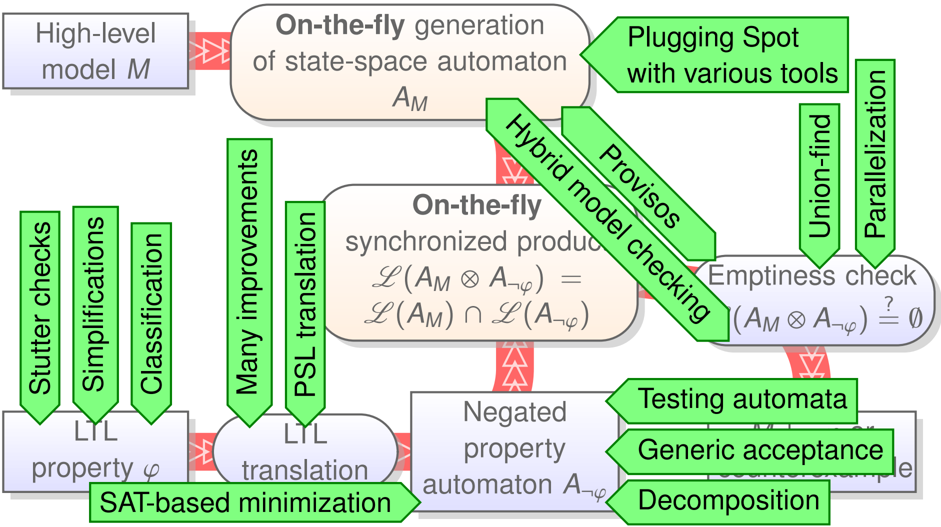
Related tasks:

- LTL and ω-automata toolbox
- Glue between third-party tools

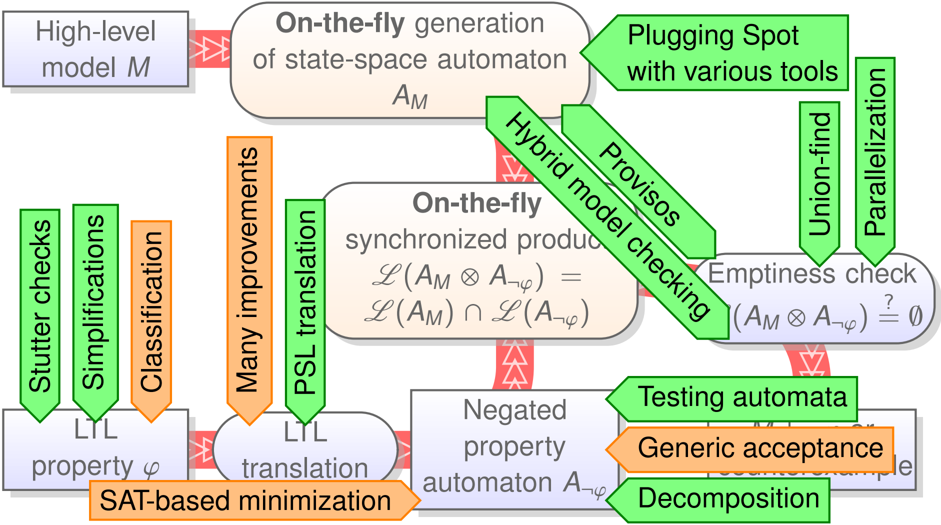
Contributions



Contributions



Contributions



0 Context & Motivation

1 LTL to Büchi

Spot has a very good translator, combining several improved procedures.

2 Generalized Acceptance

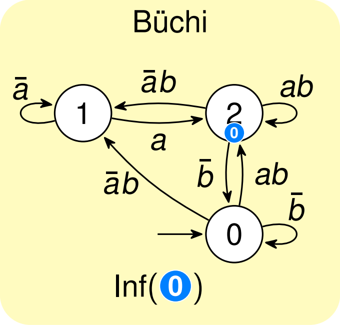
Named acceptances are a hindrance. Generic algorithms are more elegant.

3 Tooling for improvement

Spot: groundwork for research + tools for experimenting, testing, finding interesting cases.

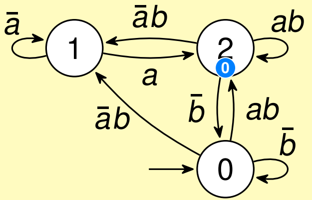
4 Closing remarks

Büchi Variations on $\mathbf{G}\,\mathbf{F}\,a \wedge \mathbf{G}\,\mathbf{F}\,b$



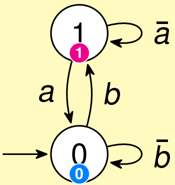
Büchi Variations on $\mathbf{G}\,\mathbf{F}\,a \wedge \mathbf{G}\,\mathbf{F}\,b$

Büchi



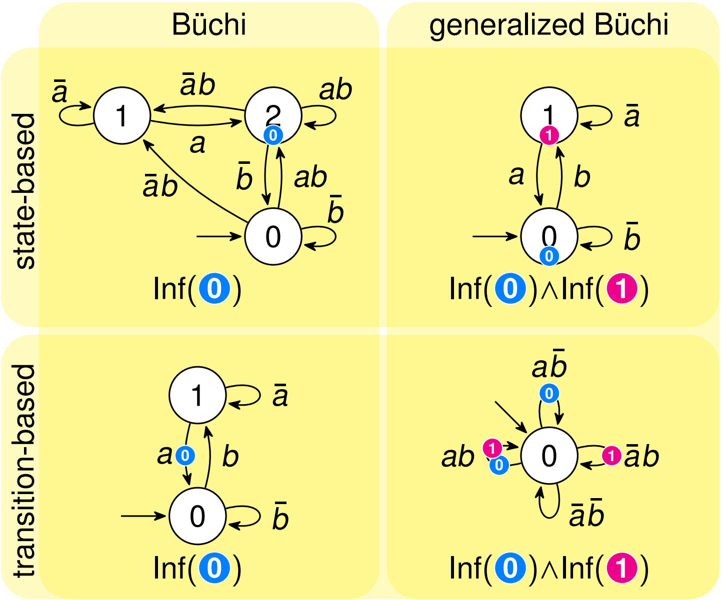
Inf(0)

generalized Büchi

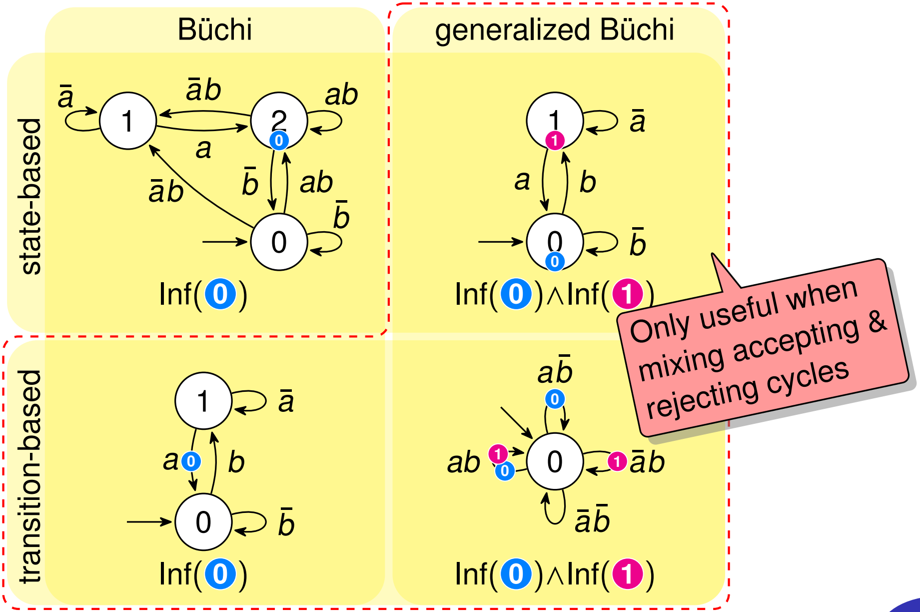


Inf(0)∧Inf(1)

Büchi Variations on $\mathbf{G}\,\mathbf{F}\,a \land \mathbf{G}\,\mathbf{F}\,b$



Büchi Variations on $\mathbf{G}\,\mathbf{F}\,a \land \mathbf{G}\,\mathbf{F}\,b$

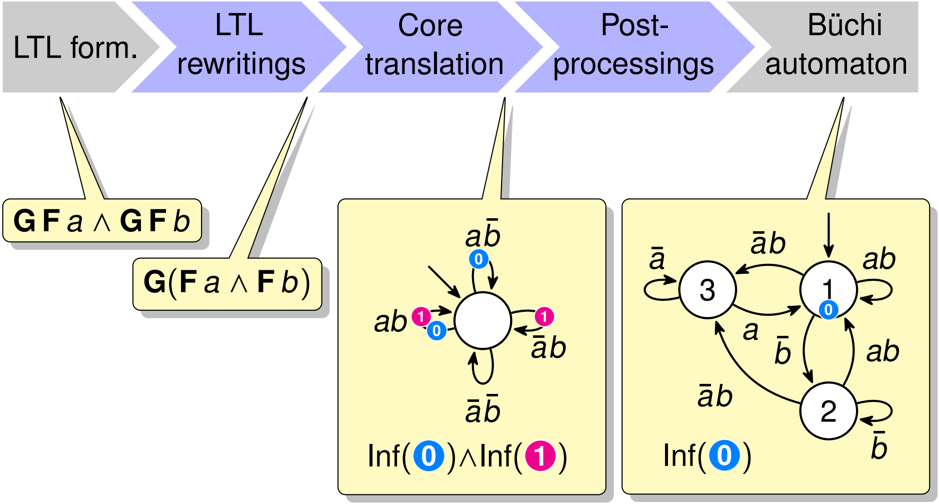


Comparison of Some “LTL to Büchi” Translators

Results summed over 178 formulas from the literature.

				automaton size			product size	
		nd.	time	st.	nd.st.	tr.	st.	tr.
	`spin` (11×☠)	162	220.7s	1440	1236	46033	259313	9433430
	`ltl2ba`	169	0.3s	1000	801	29974	190898	5616566
	`modella`	109	18.5s	1244	577	23474	210494	4033414
	`trans`	119	0.5s	957	398	16798	172246	3276714
	`ltl3ba`	115	0.7s	829	307	14322	155220	2913043
Spot	`ltl2tgba -s`	49	1.9s	666	102	10346	129419	2399328
Spot	`ltl2tgba -Ds`	44	1.9s	671	96	10456	129804	2401471

From LTL to Büchi Automata



From LTL to Büchi Automata

LTL form. → LTL rewritings → Core translation → Post-processings → Büchi automaton

- lots of rewritings
  (e.g. $f\,\mathbf{U}\,\mathbf{G}\,f \equiv \mathbf{G}\,f$)
- implication-based rewritings
  (e.g., if $f \rightarrow g$ then $f\,\mathbf{U}\,g \equiv g$)
  syntactic or automata-based

From LTL to Büchi Automata

LTL form. → LTL rewritings → Core translation → Post-processings → Büchi automaton

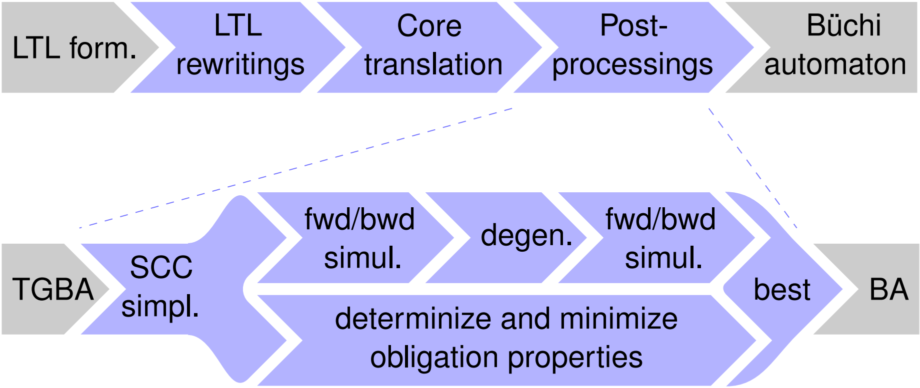
Couvreur's translation, plus:

- Improved determinism
- Improved translation of persistent formulas
- Improved translation of **G**-subformulas

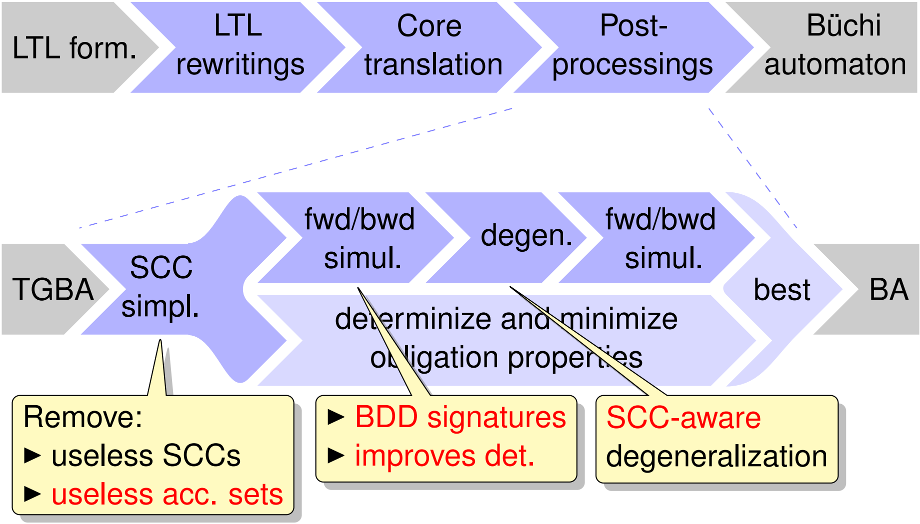
J.-M. Couvreur. On-the-fly verification of temporal logic. *FM'99*

A. Duret-Lutz. LTL translation improvements in Spot 1.0. *Int. J. on Crit. Comp.-Based Sys.*, 5(1/2):31–54, Mar. 2014

From LTL to Büchi Automata

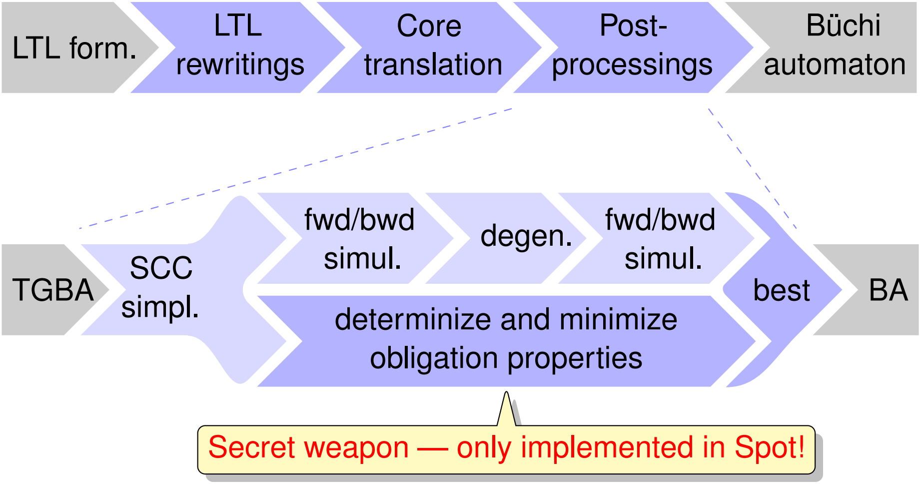


From LTL to Büchi Automata



T. Babiak, T. Badie, A. Duret-Lutz, M. Křetínský, and J. Strejček.
Compositional approach to suspension and other improvements to LTL translation. *SPIN'13*

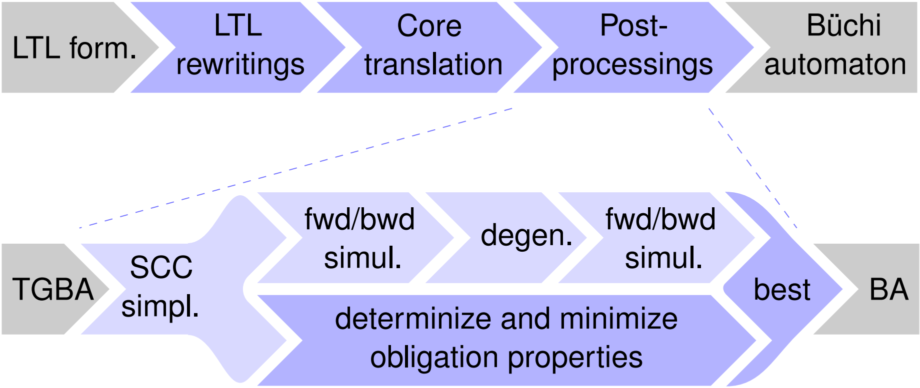
From LTL to Büchi Automata



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

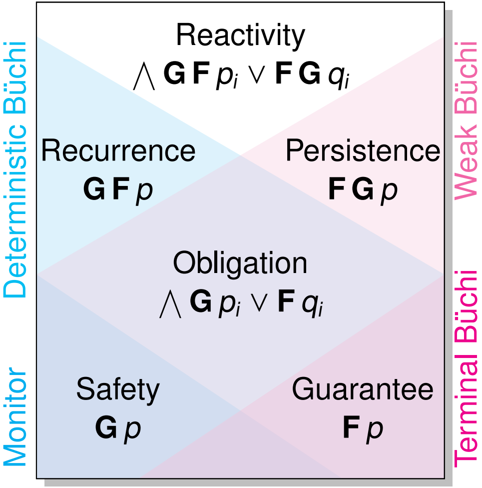
C. Dax, J. Eisinger, and F. Klaedtke. Mechanizing the powerset construction for restricted classes of ω-automata. *ATVA'07*

From LTL to Büchi Automata



Secret weapon — only implemented in Spot!
Requires: product, emptiness check, DBA complementation, NFA determinization, DFA minimization, SCC enumeration...

The Temporal Hierarchy



Z. Manna and A. Pnueli. A hierarchy of temporal properties. *PODC'90*

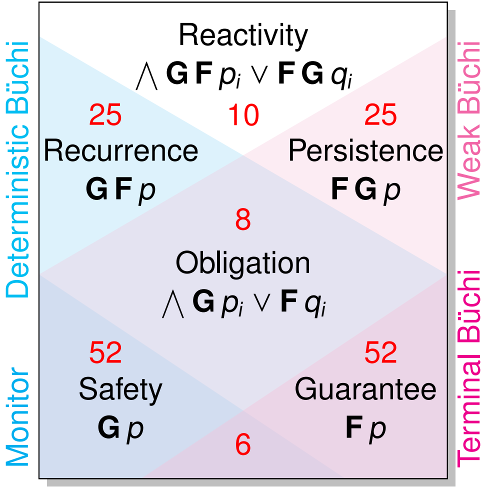
I. Černá and R. Pelánek. Relating hierarchy of temporal properties to model checking. *MFCS'03*

Comparison of Some “LTL to Büchi” Translators

Results summed over 178 formulas from the literature.

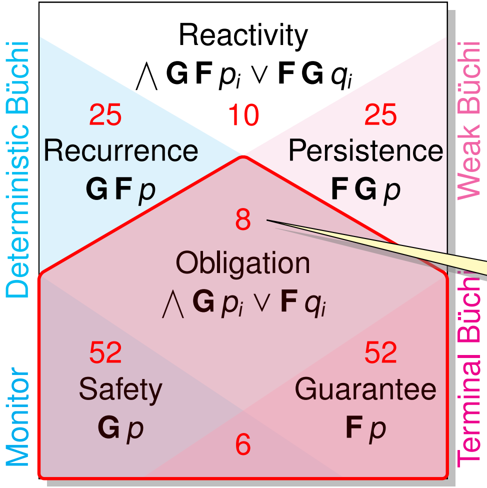
				automaton size			product size	
		nd.	time	st.	nd.st.	tr.	st.	tr.
	`spin` (11×☠)	162	220.7s	1440	1236	46033	259313	9433430
	`ltl2ba`	169	0.3s	1000	801	29974	190898	5616566
	`modella`	109	18.5s	1244	577	23474	210494	4033414
	`trans`	119	0.5s	957	398	16798	172246	3276714
	`ltl3ba`	115	0.7s	829	307	14322	155220	2913043
Spot	`ltl2tgba -s`	49	1.9s	666	102	10346	129419	2399328
	`ltl2tgba -Ds`	44	1.9s	671	96	10456	129804	2401471

The Temporal Hierarchy



- Z. Manna and A. Pnueli. A hierarchy of temporal properties. *PODC'90*
- I. Černá and R. Pelánek. Relating hierarchy of temporal properties to model checking. *MFCS'03*

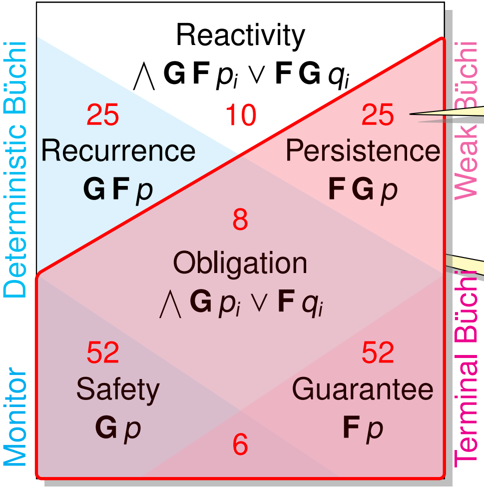
The Temporal Hierarchy



66% of the 178 formulas are obligations

Good for the secret weapon!

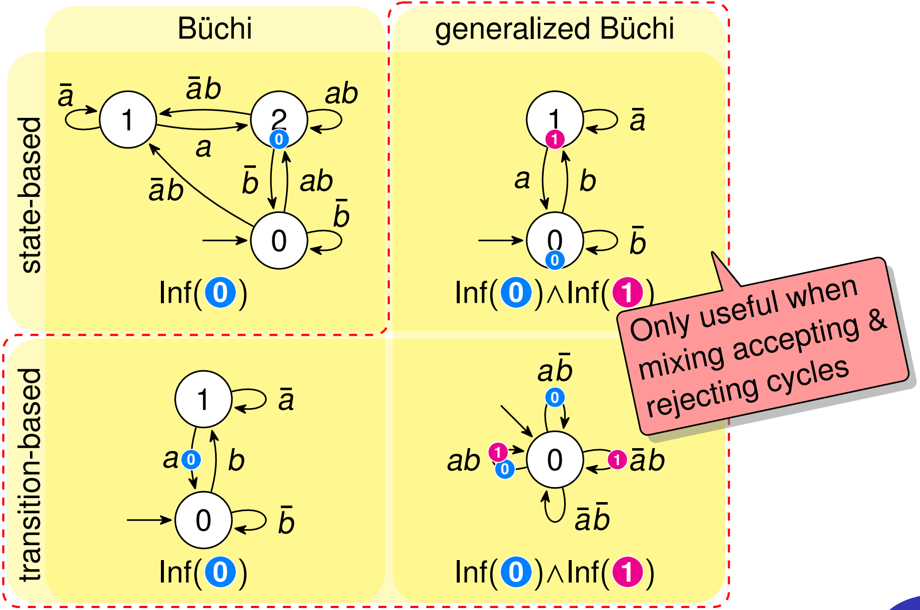
The Temporal Hierarchy



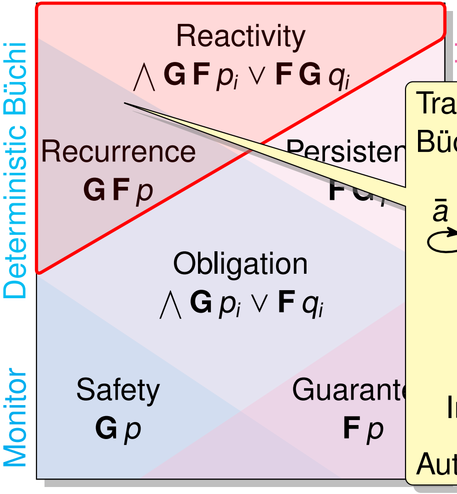
80% are persistences
Good for the improved translation!

66% of the 178 formulas are obligations
Good for the secret weapon!

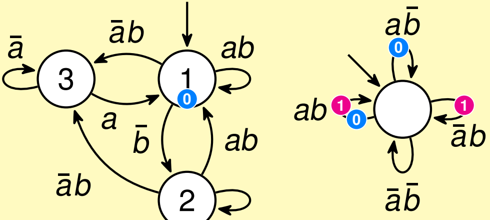
Büchi Variations on $\mathbf{G}\,\mathbf{F}\,a \wedge \mathbf{G}\,\mathbf{F}\,b$



The Temporal Hierarchy



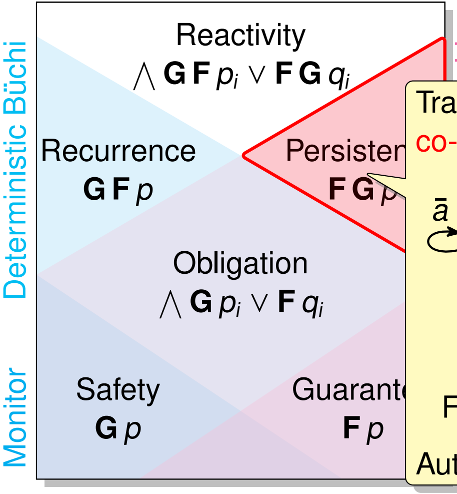
Transition-based and generalized Büchi acceptance useful. Compare:



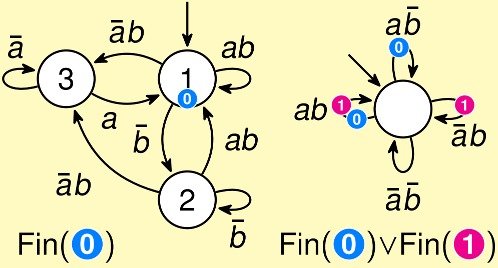
Inf(0)   Inf(0)∧Inf(1)

Automata for $\mathbf{G}\,\mathbf{F}\,a \wedge \mathbf{G}\,\mathbf{F}\,b$.

The Temporal Hierarchy

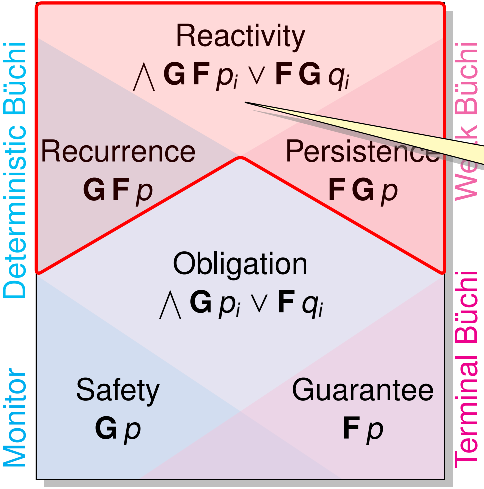


Transition-based and generalized co-Büchi acceptance useful.



Automata for $\mathbf{F}\,\mathbf{G}\,\bar{a} \vee \mathbf{F}\,\mathbf{G}\,\bar{b}$.

The Temporal Hierarchy



More complex acceptance conditions (Rabin, Streett, Parity, generalized-Rabin, etc.) compete here.

0. Context & Motivation
1. LTL to Büchi

   Spot has a very good translator, combining several improved procedures.

2. **Generalized Acceptance**

   Named acceptances are a hindrance. Generic algorithms are more elegant.

3. Tooling for improvement

   Spot: groundwork for research + tools for experimenting, testing, finding interesting cases.

4. Closing remarks

The Hanoi Omega-Automata Format

T. Babiak, F. Blahoudek, A. Duret-Lutz, J. Klein, J. Křetínský, D. Müller, D. Parker, and J. Strejček. The Hanoi Omega-Automata format. *CAV'15*

The Hanoi Omega-Automata Format

Tool support at publication

`ltl2dstar 0.5.3`	`Rabinizer 3.1`	`jhoafparser`
`ltl3ba 1.1.2`	`PRISM 4.3`	`cpphoafparser`
`ltl3dra 0.2.2`	`Spot 1.99.2`	

T. Babiak, F. Blahoudek, A. Duret-Lutz, J. Klein, J. Křetínský, D. Müller, D. Parker, and J. Strejček. The Hanoi Omega-Automata format. *CAV'15*

The Hanoi Omega-Automata Format

Original motivations

- Unify output formats for different tools/acceptance conditions
- Allow **new** acceptance conditions

positive Boolean formulas of $\mathsf{Inf}(x)$ and $\mathsf{Fin}(y)$ terms

Tool support at publication

`ltl2dstar 0.5.3`	`Rabinizer 3.1`	`jhoafparser`
`ltl3ba 1.1.2`	`PRISM 4.3`	`cpphoafparser`
`ltl3dra 0.2.2`	`Spot 1.99.2`	

T. Babiak, F. Blahoudek, A. Duret-Lutz, J. Klein, J. Křetínský, D. Müller, D. Parker, and J. Strejček. The Hanoi Omega-Automata format. *CAV'15*

The Hanoi Omega-Automata Format

Original motivations

- Unify output formats for different tools/acceptance conditions
- Allow **new** acceptance conditions

positive Boolean formulas of $\mathsf{Inf}(x)$ and $\mathsf{Fin}(y)$ terms

Tool support at publication

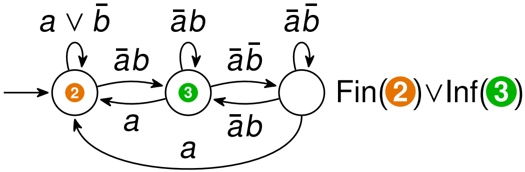
`ltl2dstar 0.5.3`	`Rabinizer 3.1`	`jhoafparser`
`ltl3ba 1.1.2`	`PRISM 4.3`	`cpphoafparser`
`ltl3dra 0.2.2`	`Spot 1.99.2`	

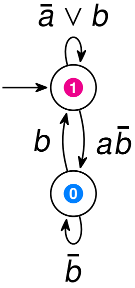
Resulting challenge

Can we build tools that process automata with arbitrary acceptance conditions?

T. Babiak, F. Blahoudek, A. Duret-Lutz, J. Klein, J. Křetínský, D. Müller, D. Parker, and J. Strejček. The Hanoi Omega-Automata format. *CAV'15*

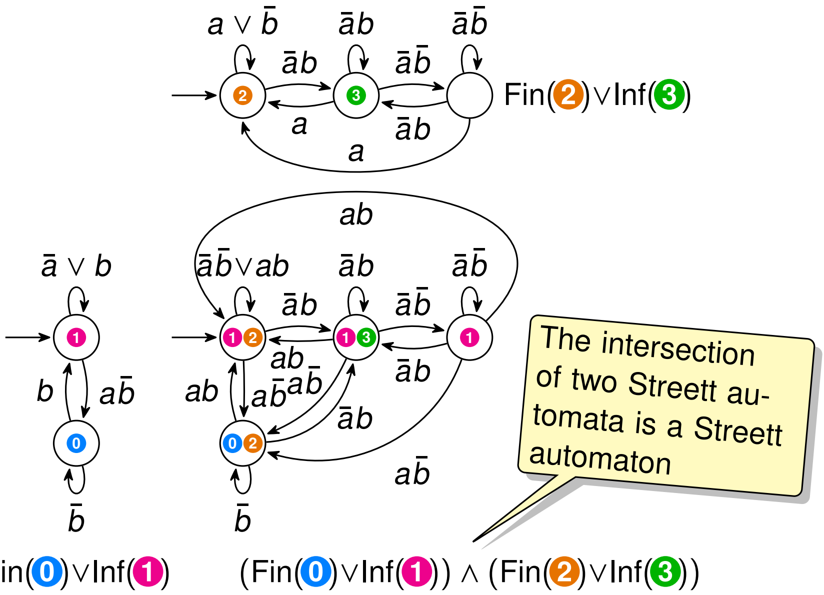
Generic Intersections and Unions



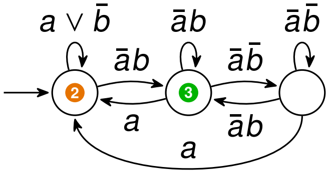


Fin(0)∨Inf(1)

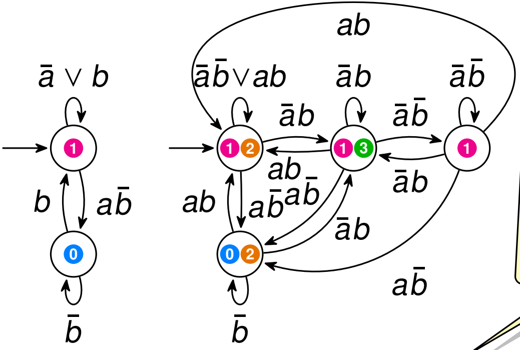
Generic Intersections and Unions



Generic Intersections and Unions



Fin(2)∨Inf(3)

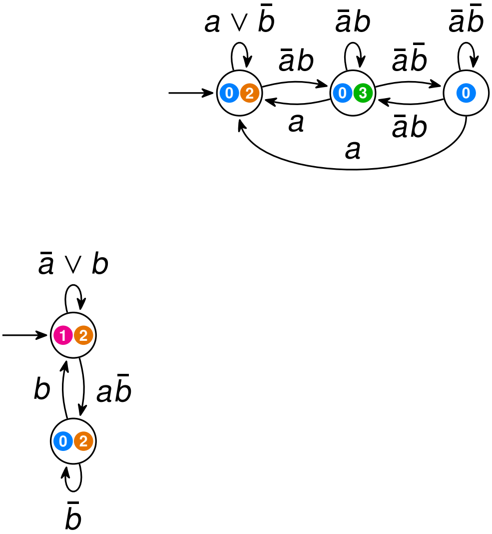


The union of two Streett automata is... nothing special

Fin(0)∨Inf(1)

(Fin(0)∨Inf(1)) ∨ (Fin(2)∨Inf(3))

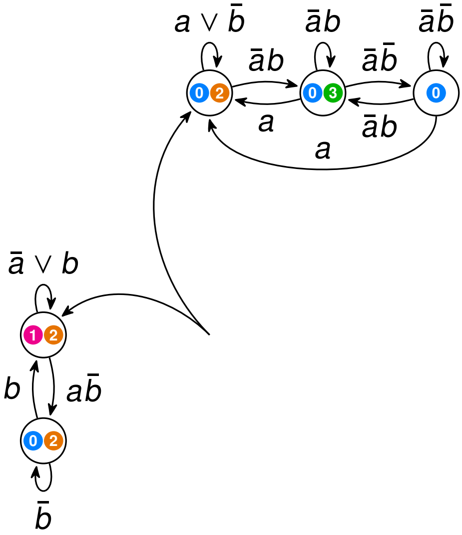
Generic Intersections and Unions



Fin(0)$\vee$Inf(1)$\vee$Fin(2)$\vee$Inf(3)

Union using non-deterministic initial states

Generic Intersections and Unions



Fin(0)∨Inf(1)∨Fin(2)∨Inf(3)

Intersection using universal initial states

Other Operations

automata simplifications
: Most of Spot's simplifications have been generalized already.

complementation
: Trivial on any deterministic ω-automata.
: What about non-deterministic ω-automata?

Other Operations

automata simplifications

Most of Spot's simplifications have been generalized already.

complementation

Trivial on any deterministic ω-automata.

What about non-deterministic ω-automata?

emptiness check

Easy for "Fin-less acceptance".

More generic algorithms in the works.

(NP-complete in the general case.)

Other Operations

automata simplifications

Most of Spot's simplifications have been generalized already.

complementation

Trivial on any deterministic ω-automata.

What about non-deterministic ω-automata?

emptiness check

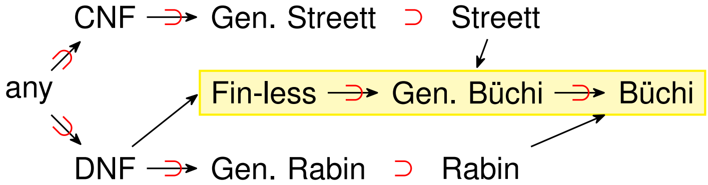
Easy for "Fin-less acceptance".

More generic algorithms in the works.

(NP-complete in the general case.)

acceptance conversions

Used before "non-generic" algorithms:



0 Context & Motivation

1 LTL to Büchi

Spot has a very good translator, combining several improved procedures.

2 Generalized Acceptance

Named acceptances are a hindrance. Generic algorithms are more elegant.

3 Tooling for improvement

Spot: groundwork for research + tools for experimenting, testing, finding interesting cases.

4 Closing remarks

`ltlcross` — testing LTL translators

How to test an LTL translator?

By comparing results of multiple translators.

- Spot has been using LBTT (*LTL-to-Büchi Translator Testbench*) since 2003 in its test-suite.

H. Tauriainen and K. Heljanko. Testing LTL formula translation into Büchi automata. *STTT*, 4(1):57–70, 2002

`ltlcross` — testing LTL translators

How to test an LTL translator?

By comparing results of multiple translators.

- Spot has been using LBTT (*LTL-to-Büchi Translator Testbench*) since 2003 in its test-suite.
  - LBTT is no longer maintained (last release in 2005),
  - LBTT is restricted to LTL,
  - LBTT is restricted generalized Büchi acceptance.

H. Tauriainen and K. Heljanko. Testing LTL formula translation into Büchi automata. *STTT*, 4(1):57–70, 2002

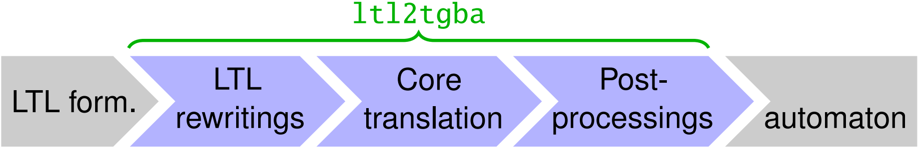
`ltlcross` — testing LTL translators

How to test an LTL translator?

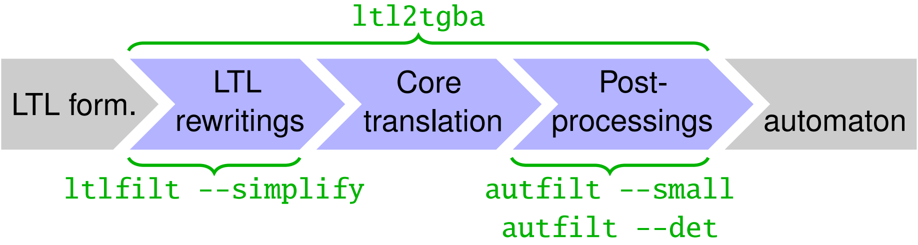
By comparing results of multiple translators.

- Spot has been using LBTT (*LTL-to-Büchi Translator Testbench*) since 2003 in its test-suite.
  - LBTT is no longer maintained (last release in 2005),
  - LBTT is restricted to LTL,
  - LBTT is restricted generalized Büchi acceptance.
- `ltlcross` is Spot's replacement of LBTT. It supports:
  - linear fragment of PSL (since Spot 1.0),
  - arbitrary acceptance conditions (since Spot 1.99.1),
  - weak alternating automata (since Spot 2.3),
  - optional determinization-based checks (since Spot 1.99.8).

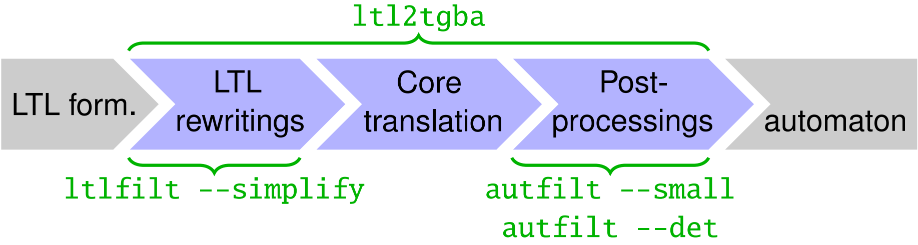
Reusing Algorithms to Improve Other Tools



Reusing Algorithms to Improve Other Tools



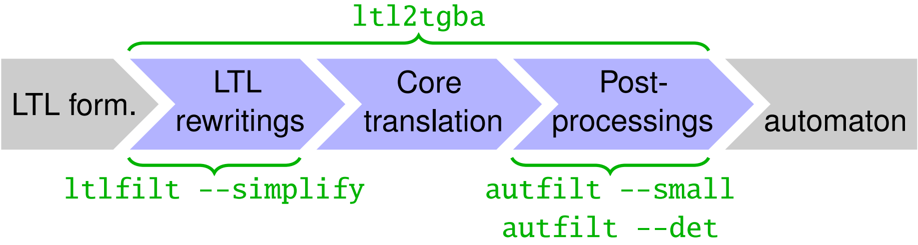
Reusing Algorithms to Improve Other Tools




```
$ ltl3dra -f 'F!p0 && Xp0 && (Gp1 || XF!p0)' | grep States:
States: 7
```

```
$ ltl3dra -f 'F!p0 && Xp0 && (Gp1 || XF!p0)' | autfilt --det |
grep States:
States: 2
```

Reusing Algorithms to Improve Other Tools




```
$ ltl3dra -f 'F!p0 && Xp0 && (Gp1 || XF!p0)' | grep States:
States: 7
```

```
$ ltl3dra -f 'F!p0 && Xp0 && (Gp1 || XF!p0)' | autfilt --det |
grep States:
States: 2
```

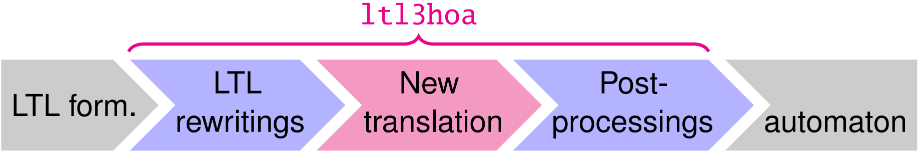
An obligation. The secret weapon did all the work.

Building New Tools Efficiently

Let us not reinvent the wheel every time a new tool is made.

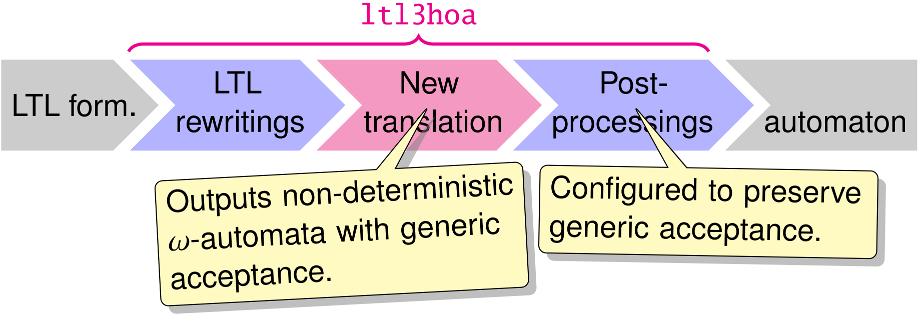
Building New Tools Efficiently

Let us not reinvent the wheel every time a new tool is made.



Building New Tools Efficiently

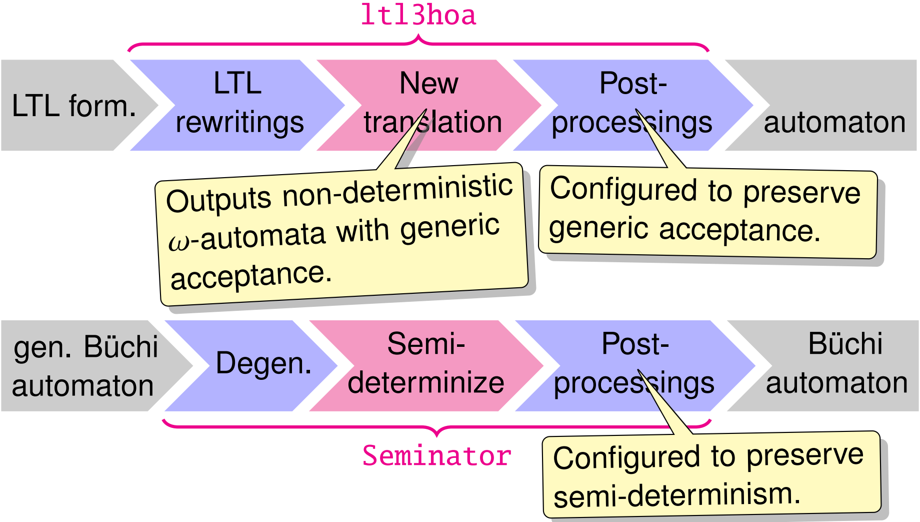
Let us not reinvent the wheel every time a new tool is made.



Tool by Juraj Major; `https://github.com/jurajmajor/ltl3hoa`.

Building New Tools Efficiently

Let us not reinvent the wheel every time a new tool is made.



Tool by Mikulas Klokocka; `https://github.com/mklokocka/seminator`.

SAT-based minimization of det. ω-automata

Using a SAT solver, we can mine for cases where our routines (or other tools) could be improved.

R. Ehlers. Minimising deterministic Büchi automata precisely using SAT solving. *SAT'10*

S. Baarir and A. Duret-Lutz. SAT-based minimization of deterministic ω-automata. *LPAR'15*

SAT-based minimization of det. ω-automata

Using a SAT solver, we can mine for cases where our routines (or other tools) could be improved.

```
$ ltldo ltl2dstar -f 'GFa->GFb' | grep 'States:\|Accept'
States: 4
Acceptance: 4 (Fin(0) & Inf(1)) | (Fin(2) & Inf(3))
```

```
$ ltldo ltl2dstar -f 'GFa->GFb' | autfilt --det |
grep 'States:\|Accept'
States: 4
Acceptance: 4 (Fin(0) & Inf(1)) | (Fin(2) & Inf(3))
```

```
$ ltldo ltl2dstar -f 'GFa->GFb' | autfilt -S --sat-minimize |
grep 'States:\|Accept'
States: 3
Acceptance: 4 (Fin(0) & Inf(1)) | (Fin(2) & Inf(3))
```

S. Baarir and A. Duret-Lutz. SAT-based minimization of deterministic ω-automata. *LPAR'15*

0. Context & Motivation
1. LTL to Büchi
2. Generalized Acceptance
3. Tooling for improvement
4. **Closing remarks**

Spot has a very good translator, combining several improved procedures.

Named acceptances are a hindrance. Generic algorithms are more elegant.

Spot: groundwork for research + tools for experimenting, testing, finding interesting cases.

Future Directions

Generalizing Algorithms

Problem: algorithms dedicated to "named acceptance conditions"
Goal: algorithms working for larger classes of acceptance
Examples:

- Can we unify emptiness checks for various acceptance conditions? (Also consider model checking under fairness hypothesis.)
- Is there a determinization procedure that unifies existing ones?

Expand into related territories

Goal: build up on the features we support
Examples:

- Alternating automata could be used for games, satisfiability, synthesis.
- With little effort, Spot would be a very nice tool for teaching.

Research Statistics (since 2007)

Publications

3 journal papers
22 conference papers (6×ATVA, 4×SPIN, 3×TACAS, 2×LPAR, 2×CIAA, CAV, FORTE, VeCOS, SUMO, FSMNLP)

Citations

Over 150 references to Spot

Reviewing

17 conference papers
5 journal papers
examiner of 2 PhD thesis

Supervision

2 PhD students
17+ Epita students

Dissemination

7 invited talks

Development

44 releases of Spot
15 releases of Vaucanson

Collaboration programs

1 French-Taiwanese ANR
1 French-Czech PHC

Co-authors (since 2007)

Co-authors of papers...

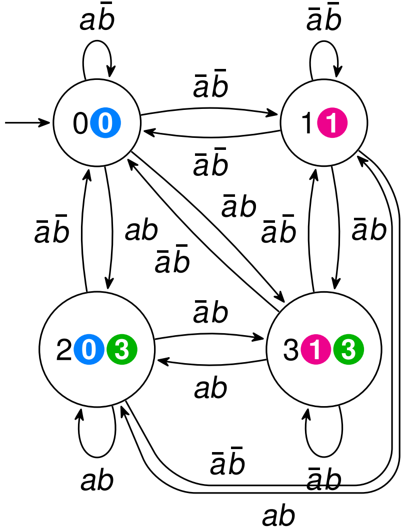
S. Baarir, T. Babiak, T. Badie, A.E. Ben Salem, F. Blahoudek, J.-M. Couvreur, A. Demaille, A. Fauchille, Ł. Fronc, K. Klai, J. Klein, F. Kordon, J. Křetínský, M. Křetínský, F. Lesaint, A. Lewkowicz, S. Lombardy, T. Michaud, D. Müller, D. Parker, D. Poitrenaud, É. Renault, V. Rujbr, L. Saiu, J. Sakarovitch, J. Strejček, F. Terrones, Y. Thierry-Mieg, L. Xu, and H.-C. Yen.

Co-authors of software (not already/yet listed above)...

F. Abecassis, E. Abi Saad, M. Colange, F. D'Halluin, J. Galtier, A. Gbaguidi Aisse, G. Gillard, A. Hamelin, G. Lazzara, D. Lefortier, G. Leroi, J. Ma, D. Moreira, P. Parutto, A. Remaud, G. Sadegh, and V. Tourneur.

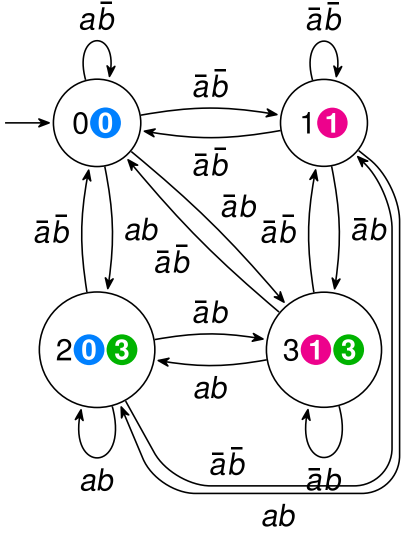
A Rabin Automaton for $\mathbf{G}\,\mathbf{F}\,a \rightarrow \mathbf{G}\,\mathbf{F}\,b$

$\big(\mathrm{Fin}(0) \wedge \mathrm{Inf}(1)\big) \vee \big(\mathrm{Fin}(2) \wedge \mathrm{Inf}(3)\big)$



A Rabin Automaton for $\mathbf{G}\,\mathbf{F}\,a \rightarrow \mathbf{G}\,\mathbf{F}\,b$

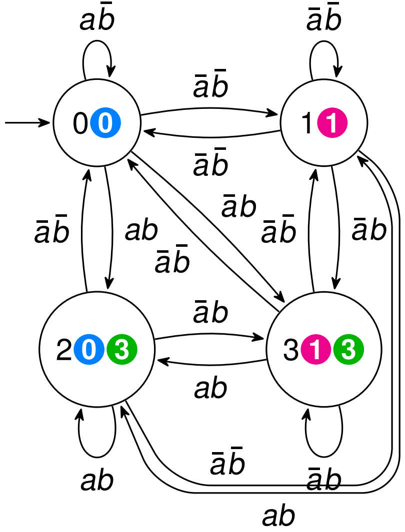
$$\big(\mathrm{Fin}(0) \wedge \mathrm{Inf}(1)\big) \vee \big(\mathrm{Fin}(2) \wedge \mathrm{Inf}(3)\big)$$




```
HOA: v1
States: 4
Start: 0
AP: 2 "a" "b"
acc-name: Rabin 2
Acceptance: 4
  Fin( 0 )&Inf( 1 )|Fin( 2 )&Inf( 3 )
--BODY--
State: 0 { 0 }
[!0&!1] 1 [0&!1] 0 [!0&1] 3 [0&1] 2
State: 1 { 1 }
[!0&!1] 1 [0&!1] 0 [!0&1] 3 [0&1] 2
State: 2 { 0   3 }
[!0&!1] 1 [0&!1] 0 [!0&1] 3 [0&1] 2
State: 3 { 1   3 }
[!0&!1] 1 [0&!1] 0 [!0&1] 3 [0&1] 2
--END--
```

A Rabin Automaton for $\mathbf{G}\,\mathbf{F}\,a \rightarrow \mathbf{G}\,\mathbf{F}\,b$

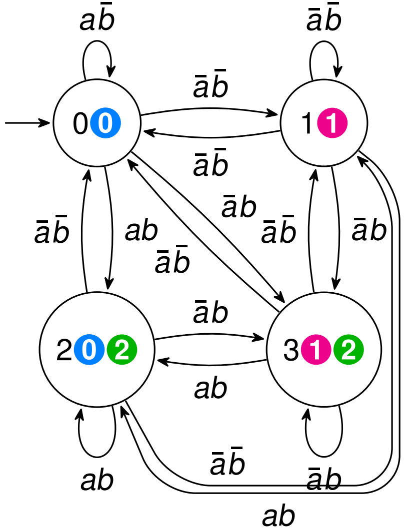
$(\mathrm{Fin}(0) \wedge \mathrm{Inf}(1)) \vee (\mathrm{Fin}(2) \wedge \mathrm{Inf}(3))$




```
HOA: v1
States: 4
Start: 0
AP: 2 "a" "b"
acc-name: Rabin 2
Acceptance: 4
  Fin(0)&Inf(1)|Fin(2)&Inf(3)
--BODY--
State: 0 {0}
[!0&!1] 1 [0&!1] 0 [!0&1] 3 [0&1] 2
State: 1 {1}
[!0&!1] 1 [0&!1] 0 [!0&1] 3 [0&1] 2
State: 2 {0 3}
[!0&!1] 1 [0&!1] 0 [!0&1] 3 [0&1] 2
State: 3 {1 3}
[!0&!1] 1 [0&!1] 0 [!0&1] 3 [0&1] 2
--END--
```

An ω-Automaton for $\mathbf{G}\,\mathbf{F}\,a \rightarrow \mathbf{G}\,\mathbf{F}\,b$

$\big(\mathrm{Fin}(0) \wedge \mathrm{Inf}(1)\big) \vee \mathrm{Inf}(2)$

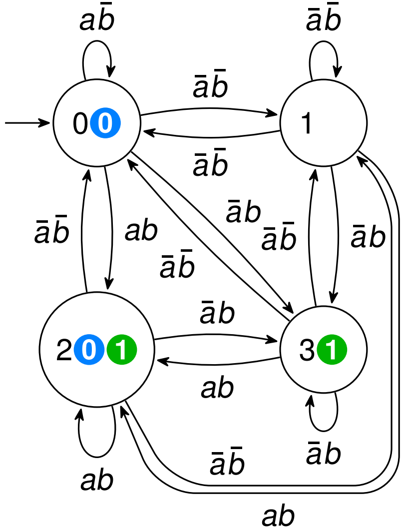



```
HOA: v1
States: 4
Start: 0
AP: 2 "a" "b"

Acceptance: 3
  Fin(0)&Inf(1)|Inf(2)
--BODY--
State: 0 {0}
[!0&!1] 1 [0&!1] 0 [!0&1] 3 [0&1] 2
State: 1 {1}
[!0&!1] 1 [0&!1] 0 [!0&1] 3 [0&1] 2
State: 2 {0 2}
[!0&!1] 1 [0&!1] 0 [!0&1] 3 [0&1] 2
State: 3 {1 2}
[!0&!1] 1 [0&!1] 0 [!0&1] 3 [0&1] 2
--END--
```

A Streett Automaton for $\mathbf{G}\,\mathbf{F}\,a \rightarrow \mathbf{G}\,\mathbf{F}\,b$

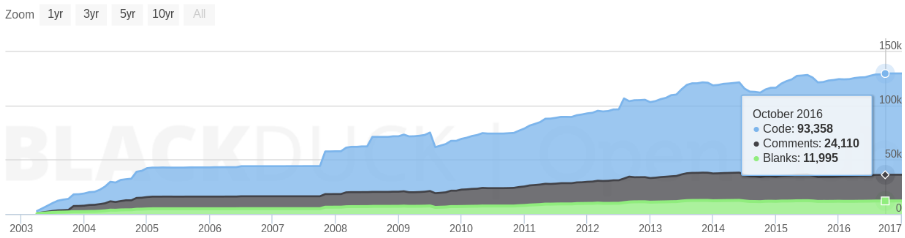
$\mathrm{Fin}(0) \vee \mathrm{Inf}(1)$



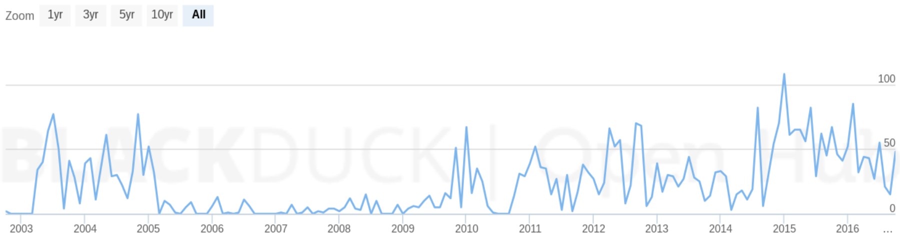

```
HOA: v1
States: 4
Start: 0
AP: 2 "a" "b"
acc-name: Streett 1
Acceptance: 2
  Fin(0)|Inf(1)
--BODY--
State: 0 {0}
[!0&!1] 1 [0&!1] 0 [!0&1] 3 [0&1] 2
State: 1
[!0&!1] 1 [0&!1] 0 [!0&1] 3 [0&1] 2
State: 2 {0 1}
[!0&!1] 1 [0&!1] 0 [!0&1] 3 [0&1] 2
State: 3 {1}
[!0&!1] 1 [0&!1] 0 [!0&1] 3 [0&1] 2
--END--
```

Code metrics

Code, Comments and Blank Lines

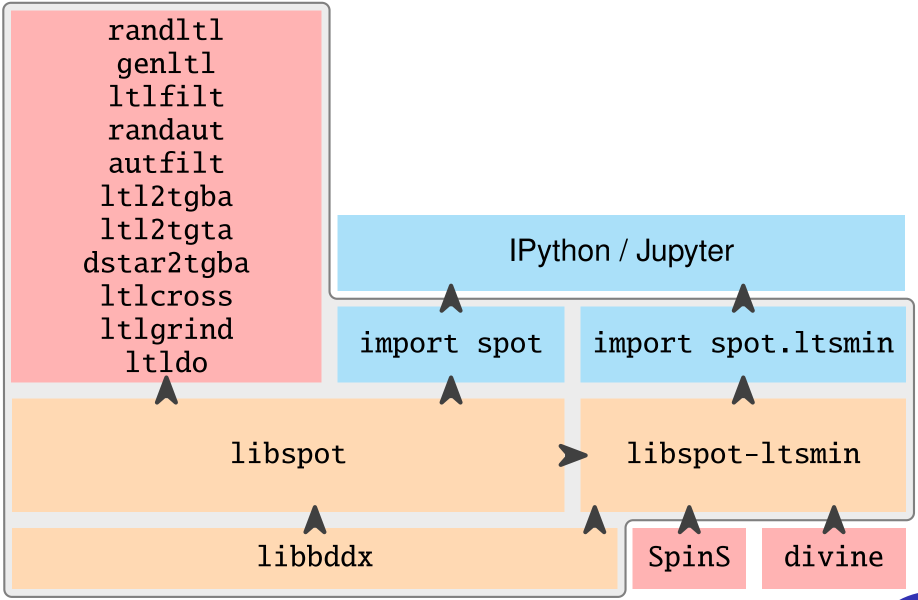


Commits per Month



Graphs provided by `https://www.openhub.net/p/spot`

Spot's Architecture

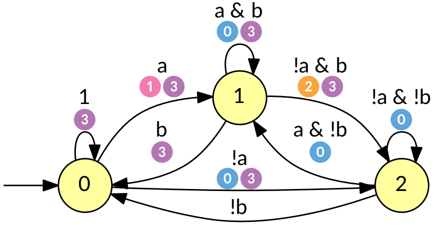


Usual Acceptance Conditions

none	f
all	t
Buchi	Inf(0)
gen. Buchi 2	$\mathrm{Inf}(0) \wedge \mathrm{Inf}(1)$
gen. Buchi 3	$\mathrm{Inf}(0) \wedge \mathrm{Inf}(1) \wedge \mathrm{Inf}(2)$
co-Buchi	Fin(0)
gen. co-Buchi 2	$\mathrm{Fin}(0) \vee \mathrm{Fin}(1)$
Rabin 1	$\mathrm{Fin}(0) \wedge \mathrm{Inf}(1)$
Rabin 2	$(\mathrm{Fin}(0) \wedge \mathrm{Inf}(1)) \vee (\mathrm{Fin}(2) \wedge \mathrm{Inf}(3))$
Streett 1	$\mathrm{Fin}(0) \vee \mathrm{Inf}(1)$
Streett 2	$(\mathrm{Fin}(0) \vee \mathrm{Inf}(1)) \wedge (\mathrm{Fin}(2) \vee \mathrm{Inf}(3))$
gen. Rabin 3 1 0 2	$(\mathrm{Fin}(0) \wedge \mathrm{Inf}(1)) \vee \mathrm{Fin}(2) \vee (\mathrm{Fin}(3) \wedge \mathrm{Inf}(4) \wedge \mathrm{Inf}(5))$
parity min odd 5	$\mathrm{Fin}(0) \wedge (\mathrm{Inf}(1) \vee (\mathrm{Fin}(2) \wedge (\mathrm{Inf}(3) \vee \mathrm{Fin}(4))))$
parity max even 5	$\mathrm{Inf}(4) \vee (\mathrm{Fin}(3) \wedge (\mathrm{Inf}(2) \vee (\mathrm{Fin}(1) \wedge \mathrm{Inf}(0))))$

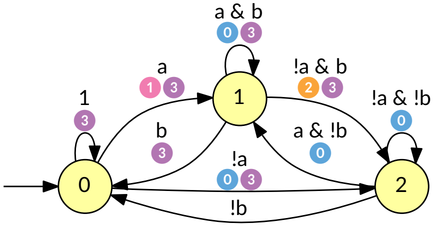
Acceptance Transformations (example)

(Fin(1) & Fin(3) & Inf(0)) | (Inf(2)&Inf(3)) | Inf(1)



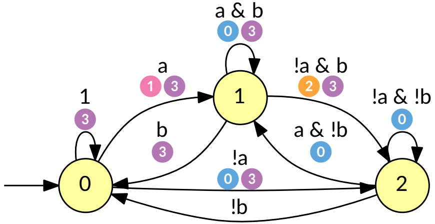
Acceptance Transformations (example)

(Fin(1) & Fin(3) & Inf(0)) | (Inf(2)&Inf(3)) | Inf(1)



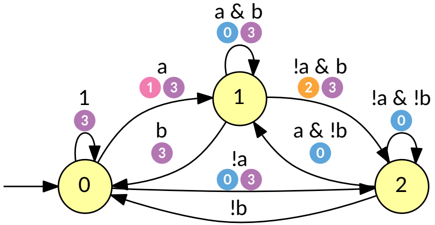

```
$ autfilt --cnf-acceptance example.hoa > output.hoa
```

(Inf(0) | Inf(1) | Inf(3)) & (Fin(3) | Inf(1) | Inf(2))



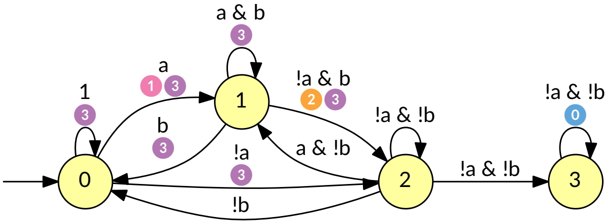
Acceptance Transformations (example)

(Fin(1) & Fin(3) & Inf(0)) | (Inf(2)&Inf(3)) | Inf(1)



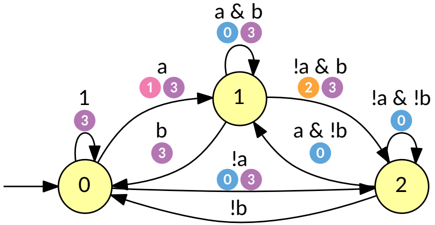

```
$ autfilt --remove-fin example.hoa > output.hoa
```

Inf(0) | Inf(1) | (Inf(2)&Inf(3))



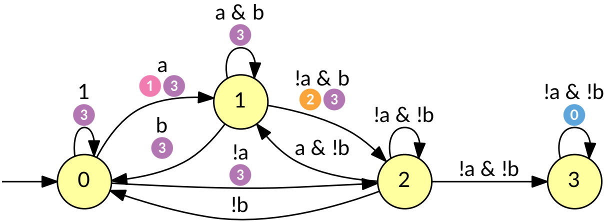
Acceptance Transformations (example)

(Fin(1) & Fin(3) & Inf(0)) | (Inf(2)&Inf(3)) | Inf(1)



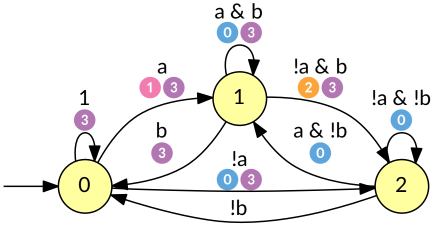

```
$ autfilt --remove-fin --cnf-acc example.hoa > output.hoa
```

(Inf(0) | Inf(1) | Inf(2)) & (Inf(0) | Inf(1) | Inf(3))



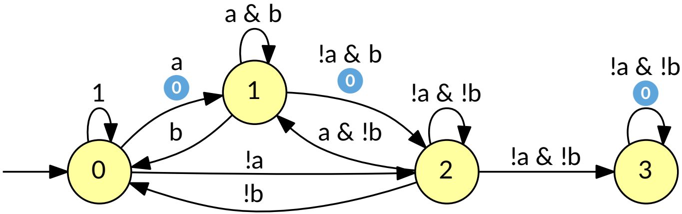
Acceptance Transformations (example)

(Fin(1) & Fin(3) & Inf(0)) | (Inf(2)&Inf(3)) | Inf(1)



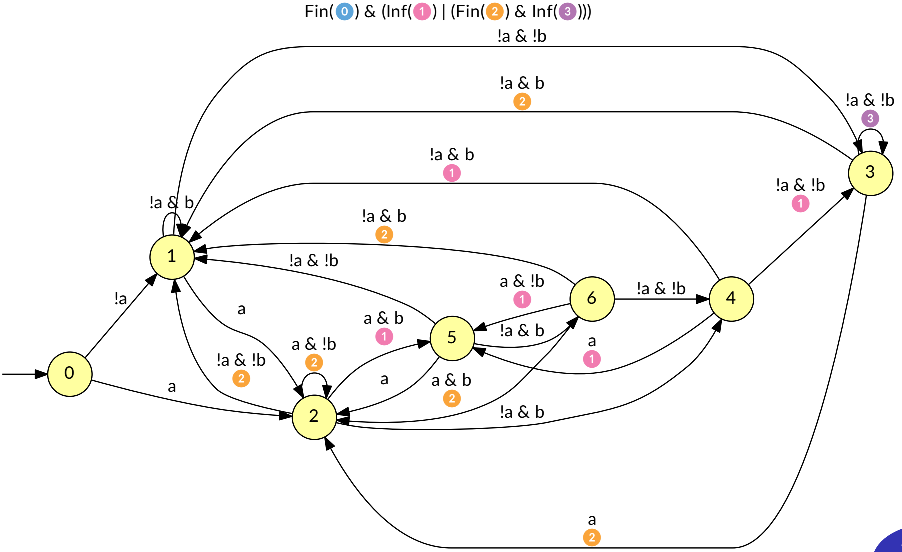

```
$ autfilt --tgba example.hoa > output.hoa
```

Inf(0)



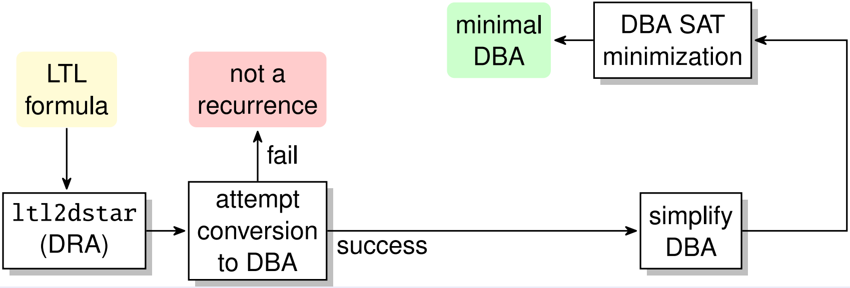
Determinization (example)

```
$ autfilt --deterministic example.hoa > output.hoa
```



From LTL to Minimal D[T][G]BA

Output: DBA. (Ehlers' setup.)

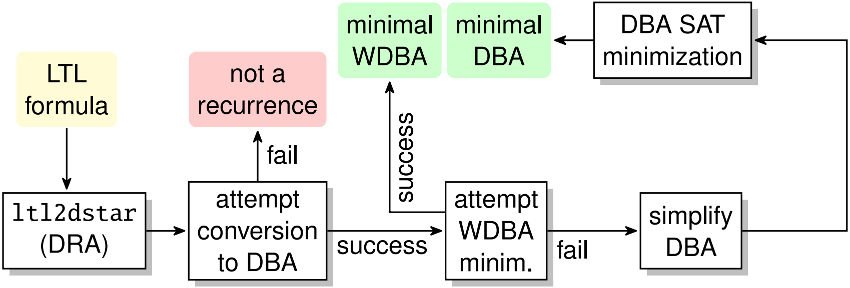


R. Ehlers. Minimising deterministic Büchi automata precisely using SAT solving. *SAT'10*

S. C. Krishnan, A. Puri, and R. K. Brayton. Deterministic ω-automata vis-a-vis deterministic Büchi automata. *ISAAC'94*

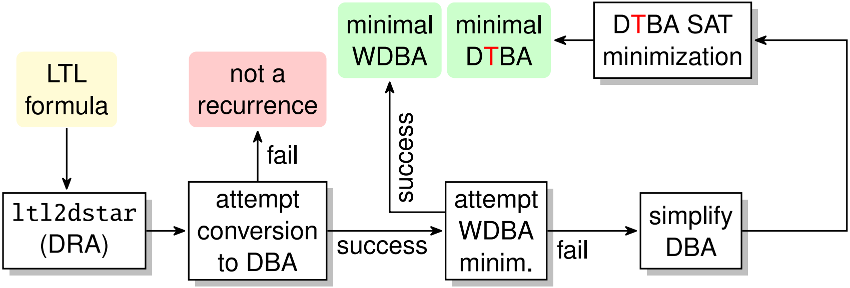
From LTL to Minimal D[T][G]BA

Output: DBA.



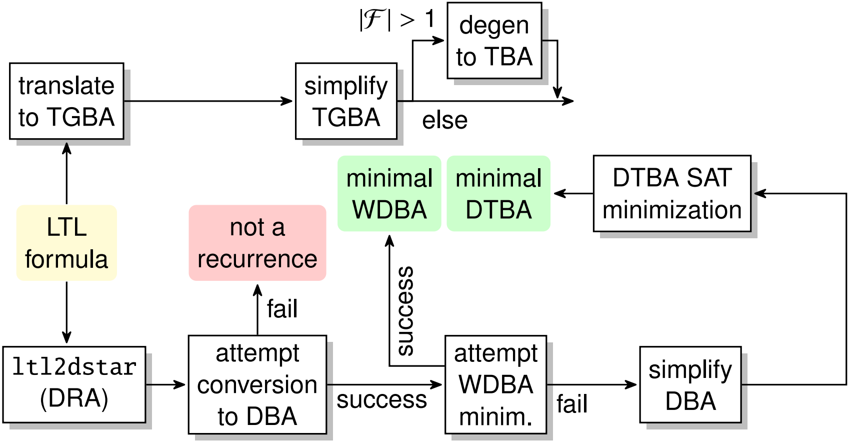
From LTL to Minimal D[T][G]BA

Output: DTBA.

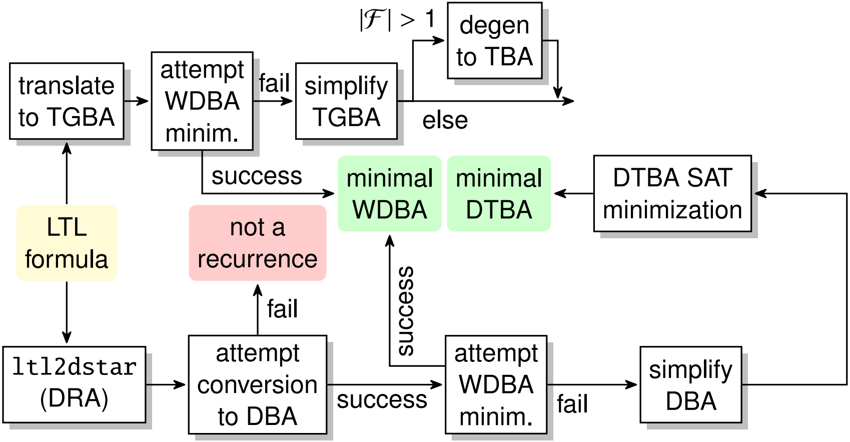


From LTL to Minimal D[T][G]BA

Output: DTBA.

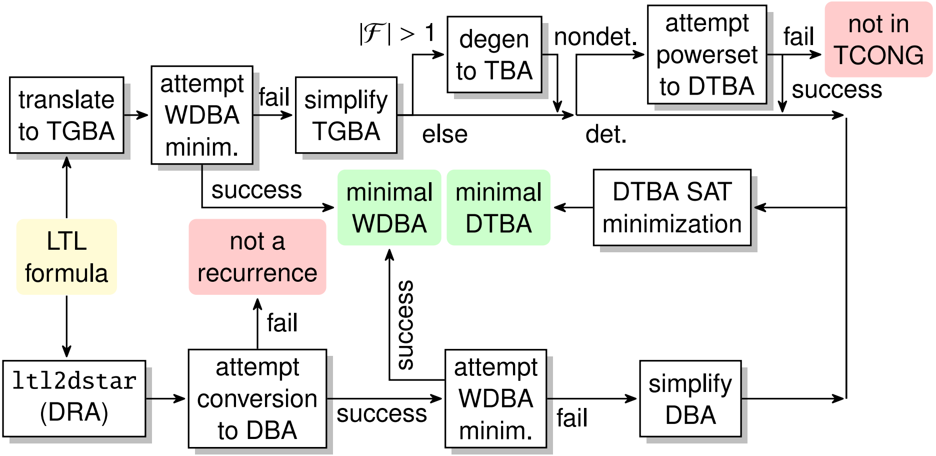


Output: DTBA.



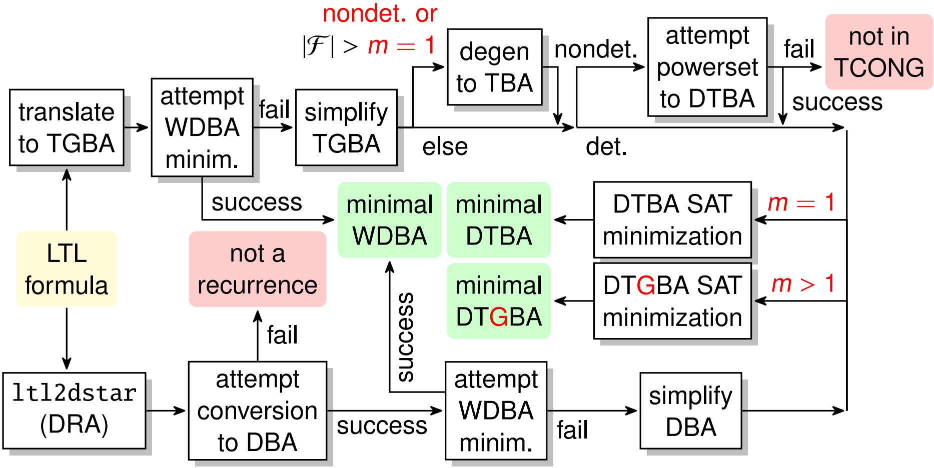
$|\mathcal{F}| > 1$
degen to TBA
translate to TGBA
attempt WDBA minim.
fail
simplify TGBA
else
success
minimal WDBA
minimal DTBA
DTBA SAT minimization
LTL formula
not a recurrence
fail
success
ltl2dstar (DRA)
attempt conversion to DBA
success
attempt WDBA minim.
fail
simplify DBA

Output: DTBA.


$|\mathcal{F}| > 1$
degen
to TBA
nondet.
attempt
powerset
to DTBA
fail
not in
TCONG
translate
to TGBA
attempt
WDBA
minim.
fail
simplify
TGBA
success
else
det.
success
minimal
WDBA
minimal
DTBA
DTBA SAT
minimization
LTL
formula
not a
recurrence
fail
success
ltl2dstar
(DRA)
attempt
conversion
to DBA
success
attempt
WDBA
minim.
fail
simplify
DBA

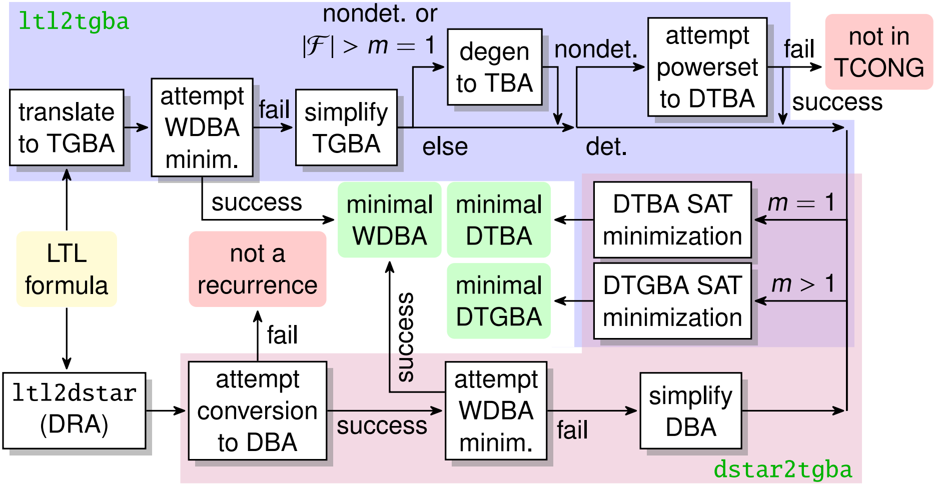
From LTL to Minimal D[T][G]BA

Output: DTGBA ($m > 1$) or DTBA ($m = 1$).



From LTL to Minimal D[T][G]BA

Output: DTGBA ($m > 1$) or DTBA ($m = 1$). Our setup.



`ltlcross` — basic operations

- Take a list of formulas (LTL/PSL) from file, stdin, or arguments.
- Take a list of translators T_1, T_2, ... given as arguments.
- For any formula φ and its negation, run all translators:
  $$P_i = T_i(\varphi) \qquad\qquad N_i = T_i(\neg\varphi)$$
- Perform the three checks of LBTT:
  - intersection tests: $\mathscr{L}(N_i \otimes P_j) = \emptyset \qquad \mathscr{L}(P_i \otimes N_j) = \emptyset$
  - cross-comparison tests (S is a random state-space)
    $$\mathscr{L}(P_i \otimes S) = \emptyset \iff \mathscr{L}(P_j \otimes S) = \emptyset$$
    $$\mathscr{L}(N_i \otimes S) = \emptyset \iff \mathscr{L}(N_j \otimes S) = \emptyset$$
  - consistency check: $\mathit{states}(P_i \otimes S)_{|S} \cup \mathit{states}(N_i \otimes S)_{|S} = S$
- Additional intersection tests in `ltlcross` (Spot 1.2):
  $$\mathscr{L}(P_i \otimes \overline{P_j}) = \emptyset \text{ if } P_j \text{ is deterministic}$$
  $$\mathscr{L}(N_i \otimes \overline{N_j}) = \emptyset \text{ if } N_j \text{ is deterministic}$$
- Once all formulas have been processed, optionally output detailed statistics in a CSV file.