

Spot 2.0
A C++ library for model checking and
 ω -automata manipulation

Alexandre DURET-LUTZ Alexandre LEWKOWICZ Amaury FAUCHILLE
Étienne RENAULT Laurent Xu Thibaud MICHAUD

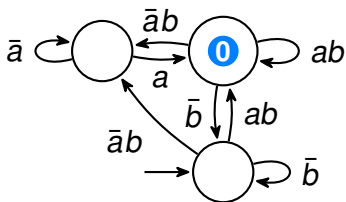
Monday, October 17th

A Generalized View of ω -Automata

ω -Automata in Spot have:

- ▶ a single initial state,
- ▶ transitions labeled by Boolean formulas over atomic prop.,
- ▶ acceptance marks (0, 1, 2, ...) that can label either states or transitions,
- ▶ an acceptance condition that tells which marks have to be seen infinitely often or finitely often for a run to be accepted.

Minimal Büchi automaton for $\mathbf{GF} a \wedge \mathbf{GF} b$:



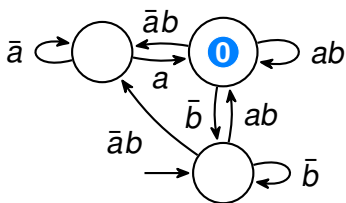
Inf(0)

A Generalized View of ω -Automata

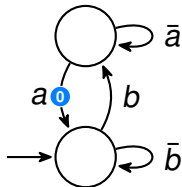
ω -Automata in Spot have:

- ▶ a single initial state,
- ▶ transitions labeled by Boolean formulas over atomic prop.,
- ▶ acceptance marks (0, 1, 2, ...) that can label either states or transitions,
- ▶ an acceptance condition that tells which marks have to be seen infinitely often or finitely often for a run to be accepted.

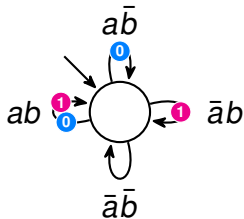
Minimal automata for $\mathbf{GF} a \wedge \mathbf{GF} b$:



Inf(0)



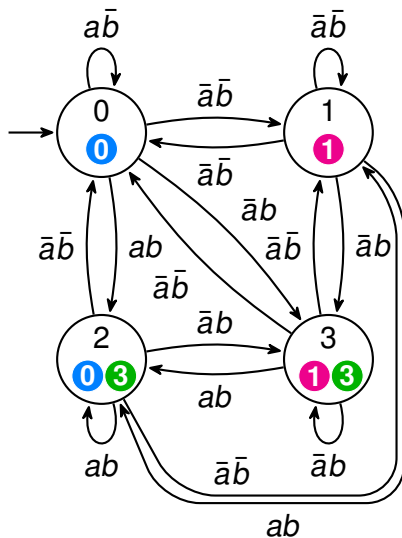
Inf(0)



Inf(0) \wedge Inf(1)

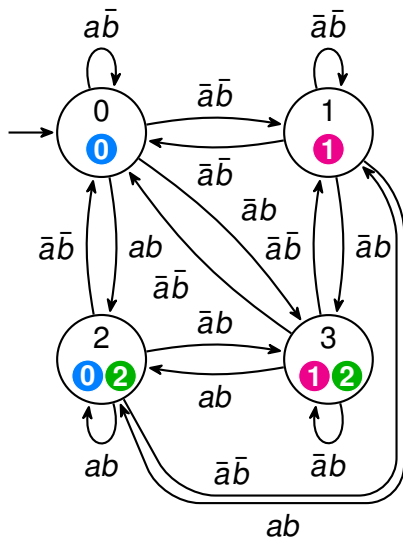
A Rabin Automaton for $\mathbf{GF} a \rightarrow \mathbf{GF} b$

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



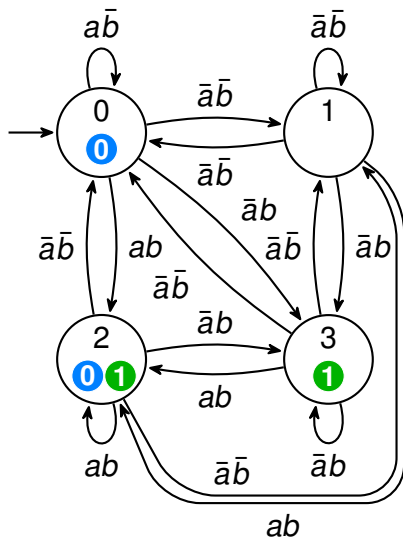
An ω -Automaton for $\mathbf{GF} a \rightarrow \mathbf{GF} b$

$$(\text{Fin}(\mathbf{0}) \wedge \text{Inf}(\mathbf{1})) \vee \text{Inf}(\mathbf{2})$$



A Streett Automaton for $\mathbf{GF} a \rightarrow \mathbf{GF} b$

Fin(**0**) \vee Inf(**1**)



Support for the Hanoi Omega-Automata Format

<http://adl.github.io/hoaf/support.html>

`ltl2dstar 0.5.3` output DRA or DSA, can also input BA

`ltl3ba 1.1.2` output BA, TGBA, or VWAA

`ltl3dra 0.2.2` output DRA, TGDRA or MMAA

`Rabinizer 3.1` output DRA, TDRA, GDRA, or TGDRA

`PRISM 4.3`

input deterministic automata for probabilistic model checking;
(generalized) Rabin for MDP; any acceptance for CTMC/DTMC

`Spot since 1.99.2`

can input/output anything that is not alternating; can convert
from other formats; has several transformations

`jhoafparser` and `cpphoafparser`

two parsers with pretty printers, and convenient transformations



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*

Spot's Services for Temporal Logic Formulas

For LTL formulas (or the linear fragment of PSL):

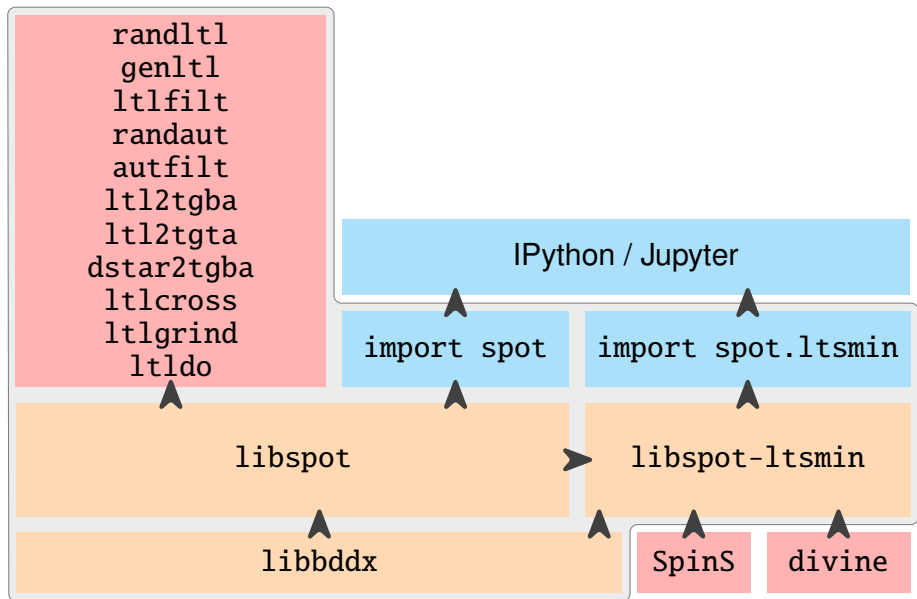
- ▶ Parsers, printers
- ▶ Simplifications, rewritings
- ▶ Implication, equivalence checks
- ▶ Stutter-invariance checks
- ▶ Filtering by properties
- ▶ Random formula generation
- ▶ Translation to Transition-based Generalized Büchi automata

Spot's Services for ω -automata

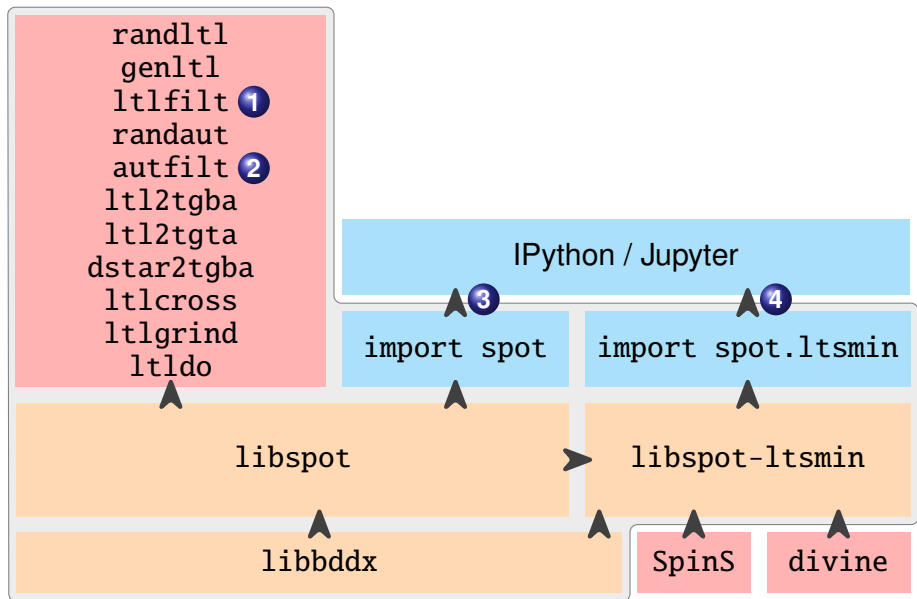
For ω -automata:

- ▶ Parsers, printers
- ▶ Simplifications: acceptance pruning, SCC-based mark simplifications, simulation-based reductions, minimization for obligation properties, ...
- ▶ Acceptance conversions
- ▶ Boolean operations: sum, product (on-the-fly if desired), complement
- ▶ Determinization (to Parity automata)
- ▶ Emptiness checks
- ▶ SAT-based minimization of deterministic automata (with arbitrary input and output acceptance)
- ▶ Random automata generation

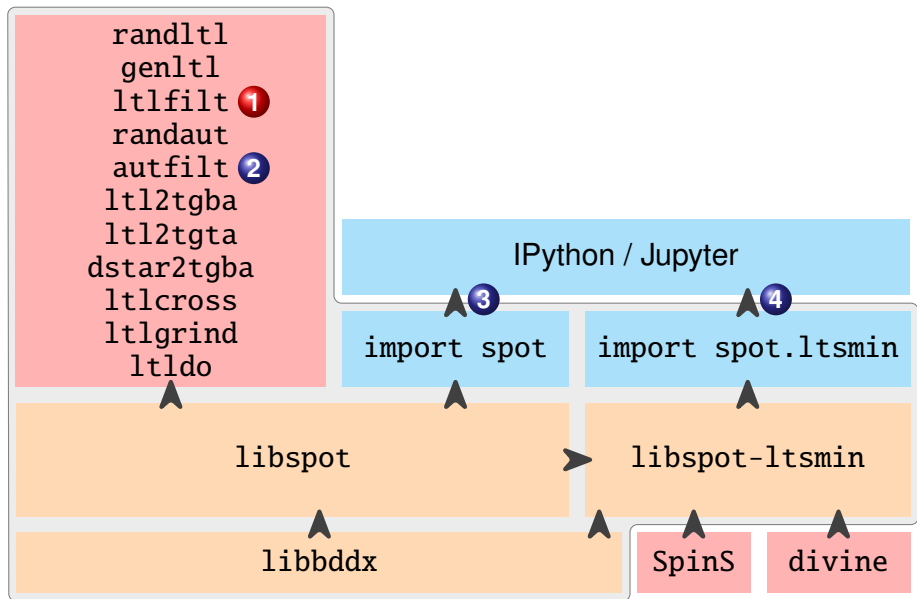
Spot's Architecture



Spot's Architecture



Spot's Architecture



ltlfi1t — Rewriting Formulas

Operator rewritings

```
$ ltlfi1t --unabbreviate=iGF -f 'G(a -> Fb)'  
0 R (!a | (1 U b))
```

ltlfilter — Rewriting Formulas

Operator rewritings

```
$ ltlfilter --unabbreviate=iGF -f 'G(a -> Fb)'  
⊙ R (!a | (1 U b))
```

```
$ ltlfilter --unabbreviate=iGFR -f 'G(a -> Fb)'  
(!a | (1 U b)) W ⊙
```

ltlfilt — Rewriting Formulas

Operator rewritings

```
$ ltlfilt --unabbreviate=iGF -f 'G(a -> Fb)'  
⊙ R (!a | (1 U b))
```

```
$ ltlfilt --unabbreviate=iGFR -f 'G(a -> Fb)'  
(!a | (1 U b)) W ⊙
```

```
$ ltlfilt --unabbreviate=iGFRW -f 'G(a -> Fb)'  
!(1 U !(a | (1 U b)))
```

ltlfilt — Rewriting Formulas

Operator rewritings

```
$ ltlfilt --unabbreviate=iGF -f 'G(a -> Fb)'  
⊙ R (!a | (1 U b))
```

```
$ ltlfilt --unabbreviate=iGFR -f 'G(a -> Fb)'  
(!a | (1 U b)) W ⊙
```

```
$ ltlfilt --unabbreviate=iGFRW -f 'G(a -> Fb)'  
!(1 U !(a | (1 U b)))
```

Simplifications

```
$ ltlfilt --simplify -f '!(1 U !(a | (1 U b)))'  
G(!a | Fb)
```


ltlfilt — Answering Simple Questions

Is $a \cup (b \cup a)$ equivalent to $b \cup a$?

```
$ ltlfilt -f 'a U (b U a)' --equivalent-to 'b U a'  
a U (b U a)
```

ltlfilt — Answering Simple Questions

Is $a \text{ U } (b \text{ U } a)$ equivalent to $b \text{ U } a$?

```
$ ltlfilt -f 'a U (b U a)' --equivalent-to 'b U a'  
a U (b U a)
```

Which of these formulas are stutter-invariant?

```
$ ltlfilt -f 'G(a | X(a -> b))' -f 'G(a | X(a <-> b))' \  
--stutter-invariant  
G(a | X(a -> b))
```



ltlfilt — Answering Simple Questions

Is $a \mathbf{U} (b \mathbf{U} a)$ equivalent to $b \mathbf{U} a$?

```
$ ltlfilt -f 'a U (b U a)' --equivalent-to 'b U a'  
a U (b U a)
```

Which of these formulas are stutter-invariant?

```
$ ltlfilt -f 'G(a | X(a -> b))' -f 'G(a | X(a <-> b))' \  
--stutter-invariant  
G(a | X(a -> b))
```

Give an \mathbf{X} -free formula for $\mathbf{G}(a \vee \mathbf{X}(a \rightarrow b))$

```
$ ltlfilt -f 'G(a | X(a -> b))' --remove-x --simplify  
G(a | (!a & (!a U (a & (!a | b))) & (!!b U a) | (b U a))) |  
(b & (b U (!b & (!a | b))) & (!!a U !b) | (a U !b))) | (!!a |  
b) & (G!a | Ga) & (G!b | Gb)) | (!b & (!!a U b) | (a U b)))
```

 K. Etessami. A note on a question of Peled and Wilke regarding stutter-invariant LTL. *Information Processing Letters*, 75(6):261–263, 2000

ltlflt — Random Generation with Constraints

Build 10 pathological safety formulas

```
$ randltl -n -1 --tree-size=10..13 a b |  
  ltlflt --simplify --safety --uniq |  
  ltlflt --invert-match --syntactic-safety -n 10
```

```
F(!a | Ga)  
(!b & X((b W Xb) R b)) | (b & X(!b M X!b) U !b))  
G(b U XXb)  
(((!b & XGa) | (b & XF!a)) W b) R a  
((b W a) M b) | (!a R X!a)  
F(b | X!b)  
G((b & XF!b) | (!b & XGb))  
Xa U (Gb | Ga)  
Fa R X!a  
F(b | G!b)
```

ltlflt — Random Generation with Constraints

Build 10 pathological safety formulas

```
$ randltl -n -1 --tree-size=10..13 a b |  
  ltlflt --simplify --safety --uniq |  
  ltlflt --invert-match --syntactic-safety -n 10
```

F(!a | Ga)

(!b & X((b W Xb) R b)) | (b & X(!b M X!b) U !b))

G(b U XXb)

(((!b & XGa) | (b & XF!a)) W b) R a

((b W a) M b) | (!a R X!a)

F(b | X!b)

G((b & XF!b) | (!b & XGb))

Xa U (Gb | Ga)

Fa R X!a

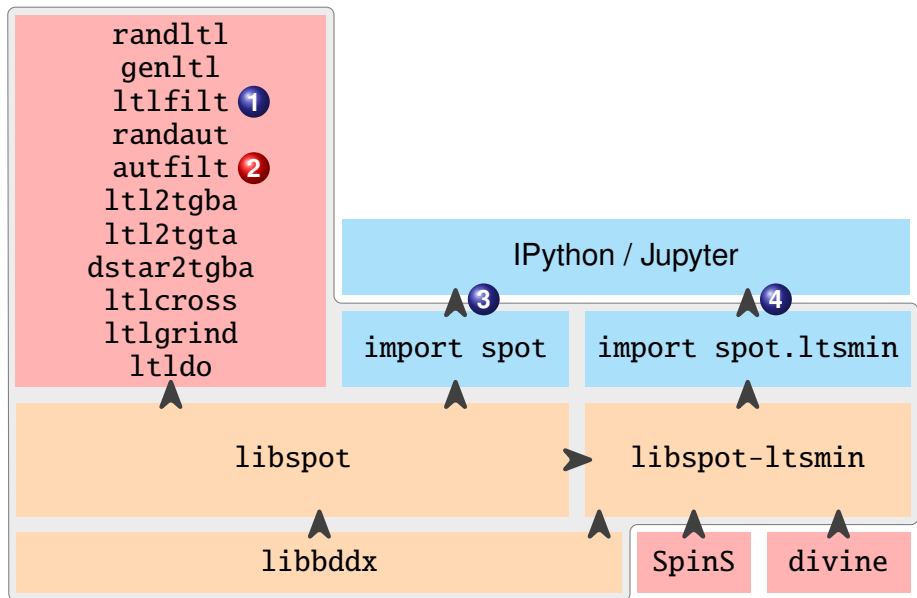
F(b | G!b)

ltlflt — Random Generation with Constraints

Build 10 pathological safety formulas not equivalent to \top or \perp

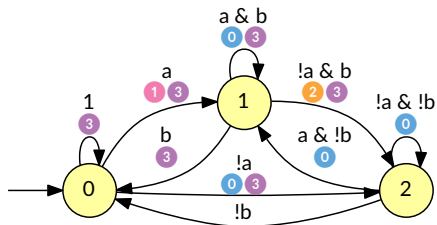
```
$ randltl -n -1 --tree-size=10..13 a b |  
  ltlflt --simplify --safety --uniq |  
  ltlflt --invert-match --syntactic-safety |  
  ltlflt --invert-match --equivalent-to=1 |  
  ltlflt --invert-match --equivalent-to=0  
  
(!b & X((b W Xb) R b)) | (b & X(!b M X!b) U !b))  
G(b U XXb)  
(((!b & XGa) | (b & XF!a)) W b) R a  
((b W a) M b) | (!a R X!a)  
Xa U (Gb | Ga)  
Fa R X!a  
(a & (!a U b)) | (!a & (a R !b))  
G(((a & b) | (!a & !b)) & (a M b)) R a  
b & ((!b & F!a) | (b & Ga))  
b & (!a | (b M Xb))
```

Spot's Architecture



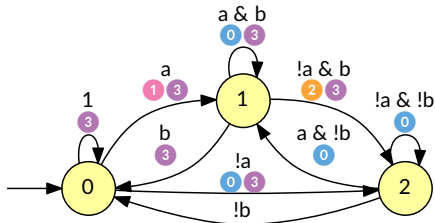
autfilt — Acceptance Transformations

$(\text{Fin}(1) \ \& \ \text{Fin}(3) \ \& \ \text{Inf}(0)) \mid (\text{Inf}(2) \ \& \ \text{Inf}(3)) \mid \text{Inf}(1)$



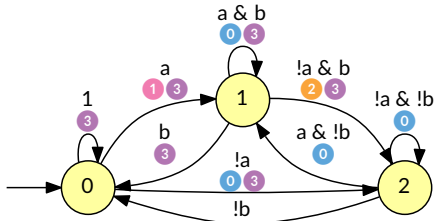
autfilt — Acceptance Transformations

$(\text{Fin}(1) \ \& \ \text{Fin}(3) \ \& \ \text{Inf}(0)) \mid (\text{Inf}(2) \ \& \ \text{Inf}(3)) \mid \text{Inf}(1)$



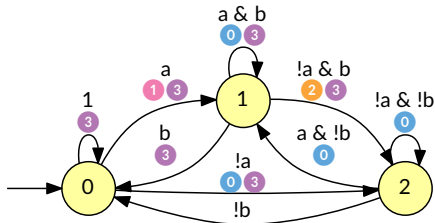
```
$ autfilt --cnf-acceptance example.hoa > output.hoa
```

$(\text{Inf}(0) \mid \text{Inf}(1) \mid \text{Inf}(3)) \ \& \ (\text{Fin}(3) \mid \text{Inf}(1) \mid \text{Inf}(2))$



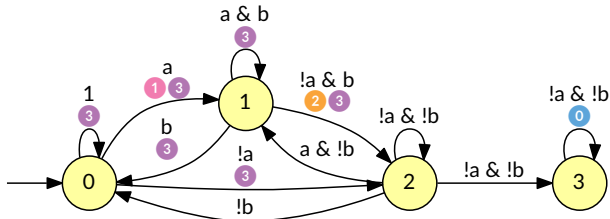
autfilt — Acceptance Transformations

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



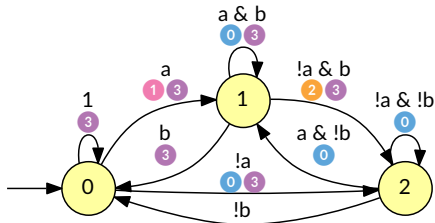
```
$ autfilt --remove-fin example.hoa > output.hoa
```

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



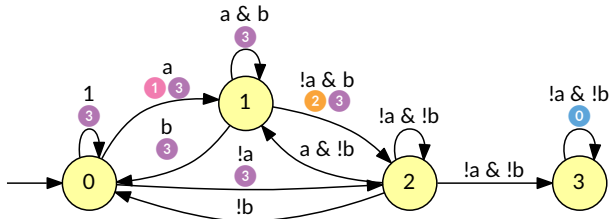
autfilt — Acceptance Transformations

$(\text{Fin}(1) \ \& \ \text{Fin}(3) \ \& \ \text{Inf}(0)) \mid (\text{Inf}(2) \ \& \ \text{Inf}(3)) \mid \text{Inf}(1)$



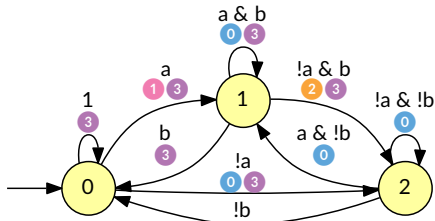
```
$ autfilt --remove-fin --cnf-acc example.hoa > output.hoa
```

$(\text{Inf}(0) \mid \text{Inf}(1) \mid \text{Inf}(2)) \ \& \ (\text{Inf}(0) \mid \text{Inf}(1) \mid \text{Inf}(3))$



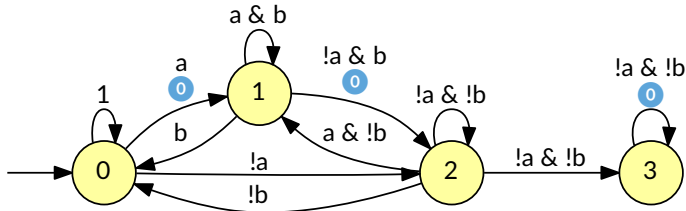
autfilt — Acceptance Transformations

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



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

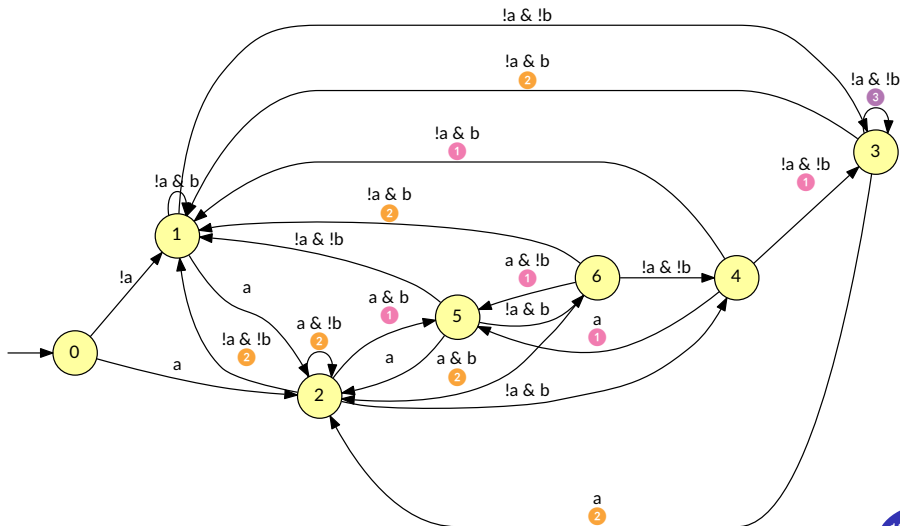
Inf(0)



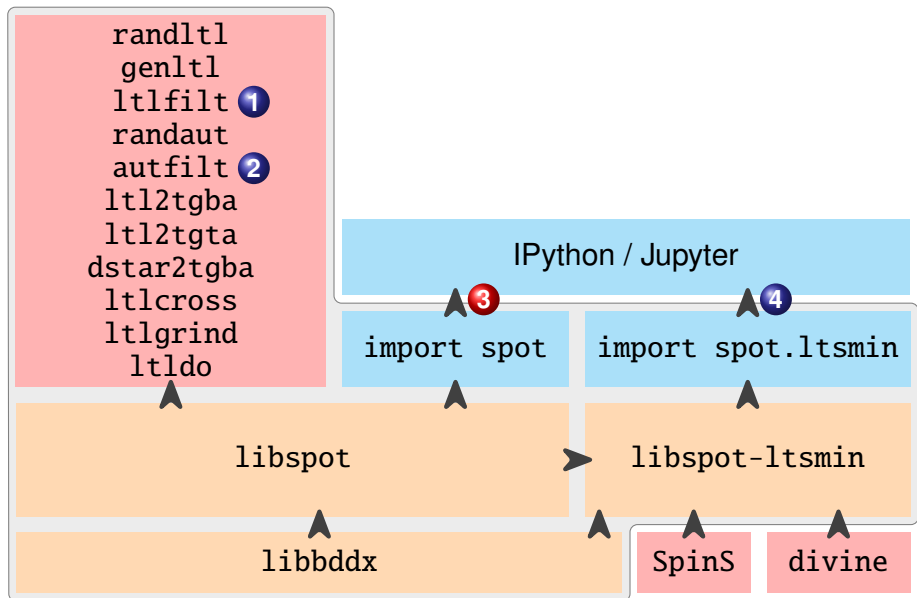
autfilt — Determinization

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

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

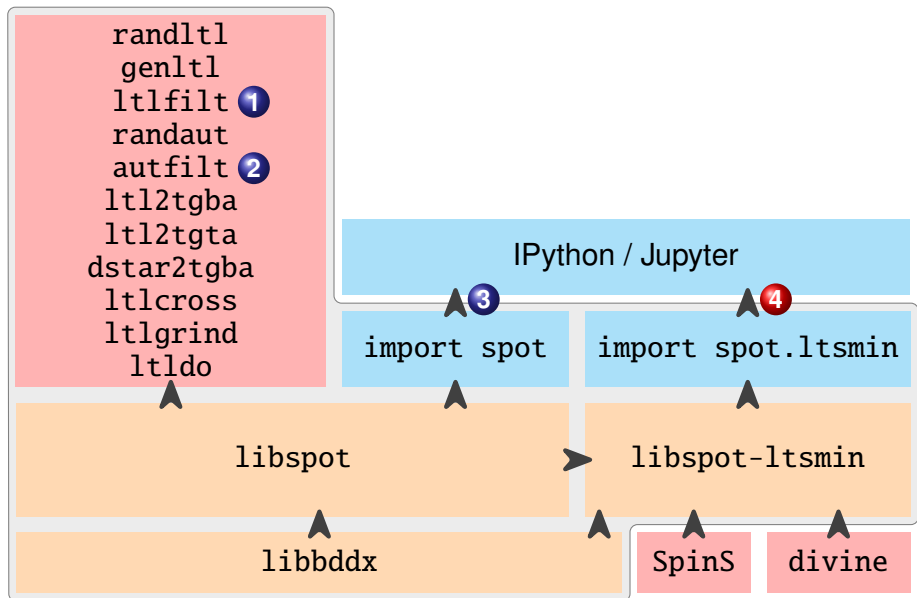


Spot's Architecture



Live demo.

Spot's Architecture



Automata-theoretic LTL model checking

High-level
model M

On-the-fly generation
of state-space automaton
 A_M

Custom Model Checker

SPOT

On-the-fly
synchronized product
 $\mathcal{L}(A_M \otimes A_{\neg\varphi}) =$
 $\mathcal{L}(A_M) \cap \mathcal{L}(A_{\neg\varphi})$

Emptiness check
 $\mathcal{L}(A_M \otimes A_{\neg\varphi}) \stackrel{?}{=} \emptyset$

LTL
property φ

LTL
translation

Negated
property
automaton $A_{\neg\varphi}$

$M \models \varphi$
or coun-
terexample

Live demo.

Conclusion & Availability

Summary:

- ▶ C++ library for model checking & ω -automata manipulation
- ▶ Support for generic acceptance conditions
- ▶ Shell commands & Python bindings
- ▶ Interface with several tools

Availability:

- ▶ <https://spot.lrde.epita.fr/> (new site)
- ▶ License GNU GPL v3+
- ▶ Source tarball or Debian packages
- ▶ Live Jupyter installation at
<http://spot-sandbox.lrde.epita.fr/>

Future work:

- ▶ More uses of generic acceptance condition