2016 2021

From Spot 2.0 to Spot 2.10: What's New?

Alexandre Duret-Lutz Etienne Renault Maximilien Colange Florian Renkin
Alexandre Gbaguidi Aisse Philipp Schlehuber-Caissier Thomas Medioni
Antoine Martin Jérôme Dubois Clément Gillard Henrich Lauko

CAV'22

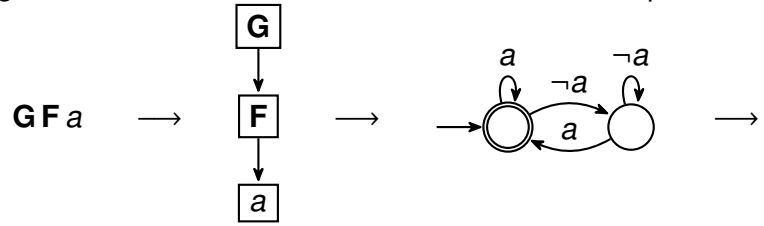
What is Spot?

A platform for manipulation of LTL formulas and ω-automata. With three interfaces.

- As a C++17 library
- Via command-line tools
- Via Python bindings

What is Spot?

A platform for manipulation of LTL formulas and ω-automata. With three interfaces.
E.g. convert an LTL formula into a neverclaim for Spin:

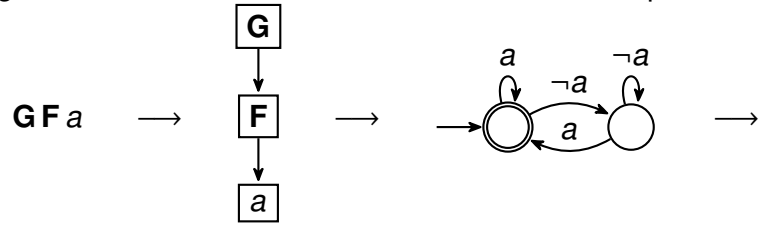



```
never { /* GFa */
accept_init:
  if
  :: (a) -> goto accept_init
  :: (!(a)) -> goto T0_S1
  fi;
T0_S1:
  if
  :: (a) -> goto accept_init
  :: (!(a)) -> goto T0_S1
  fi;
}
```

- As a C++17 library
- Via command-line tools
- Via Python bindings

What is Spot?

A platform for manipulation of LTL formulas and ω-automata. With three interfaces.
E.g. convert an LTL formula into a neverclaim for Spin:



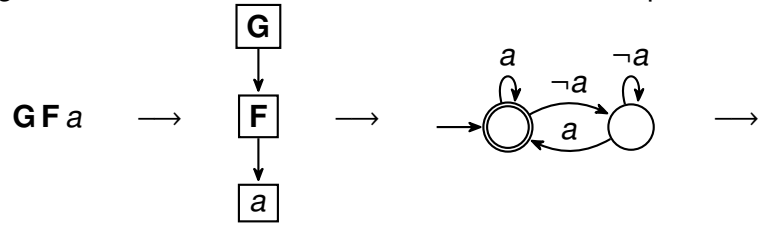

```
never { /* GFa */
accept_init:
  if
  :: (a) -> goto accept_init
  :: (!(a)) -> goto T0_S1
  fi;
T0_S1:
  if
  :: (a) -> goto accept_init
  :: (!(a)) -> goto T0_S1
  fi;
}
```

- As a C++17 library
- Via command-line tools
- Via Python bindings

```
spot::parsed_formula pf =
  spot::parse_infix_psl("GFa");
if (pf.format_errors(std::cerr))
  return 1;
spot::translator trans;
trans.set_type(spot::postprocessor::Buchi);
trans.set_pref(spot::postprocessor::SBAcc
               | spot::postprocessor::Small);
spot::twa_graph_ptr aut = trans.run(pf.f);
print_never_claim(std::cout, aut) << '\n';
```

What is Spot?

A platform for manipulation of LTL formulas and ω-automata. With three interfaces.
E.g. convert an LTL formula into a neverclaim for Spin:




```
never { /* GFa */
accept_init:
  if
  :: (a) -> goto accept_init
  :: (!(a)) -> goto T0_S1
  fi;
T0_S1:
  if
  :: (a) -> goto accept_init
  :: (!(a)) -> goto T0_S1
  fi;
}
```

- As a C++17 library

```
spot::parsed_formula pf =
  spot::parse_infix_psl("GFa");
if (pf.format_errors(std::cerr))
  return 1;
spot::translator trans;
trans.set_type(spot::postprocessor::Buchi);
trans.set_pref(spot::postprocessor::SBAcc
               | spot::postprocessor::Small);
spot::twa_graph_ptr aut = trans.run(pf.f);
print_never_claim(std::cout, aut) << '\n';
```

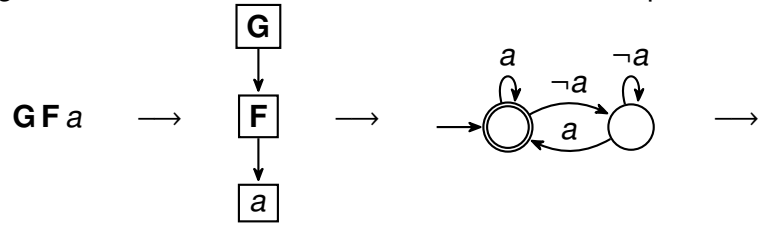
- Via command-line tools

```
% ltl2tgba --spin GFa
```

- Via Python bindings

What is Spot?

A platform for manipulation of LTL formulas and ω-automata. With three interfaces.
E.g. convert an LTL formula into a neverclaim for Spin:




```
never { /* GFa */
accept_init:
  if
  :: (a) -> goto accept_init
  :: (!(a)) -> goto T0_S1
  fi;
T0_S1:
  if
  :: (a) -> goto accept_init
  :: (!(a)) -> goto T0_S1
  fi;
}
```

- As a C++17 library

```
spot::parsed_formula pf =
  spot::parse_infix_psl("GFa");
if (pf.format_errors(std::cerr))
  return 1;
spot::translator trans;
trans.set_type(spot::postprocessor::Buchi);
trans.set_pref(spot::postprocessor::SBAcc
               | spot::postprocessor::Small);
spot::twa_graph_ptr aut = trans.run(pf.f);
print_never_claim(std::cout, aut) << '\n';
```

- Via command-line tools

```
% ltl2tgba --spin GFa
```

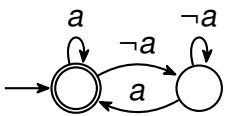
- Via Python bindings

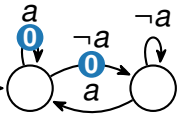
```
import spot
a = spot.translate('GFa', 'buchi', 'sbacc')
print(a.to_str('spin'))
```

- With graphical representations in Jupyter

Transition-based Emerson-Lei Automata (TELA) Internally

- Each transition is colored with a subset of {0, 1, 2, ...}
- Acceptance conditions are Boolean combinations of Inf(2) or Fin(4)...

E.g., 

is in fact stored as 

Inf(0), plus some meta-data to interpret the automaton as state-based where needed.

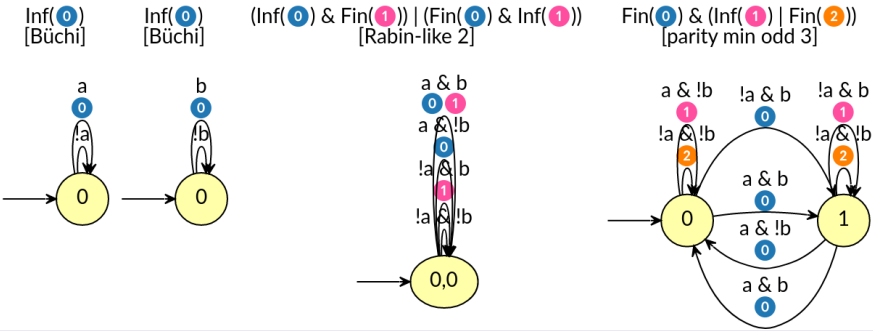
Traditional acceptance conditions (Büchi, co-Büchi, Rabin, Streett, parity, and their *generalized* variants) can all be expressed.

But arbitarily complex acceptance conditions are possible.

Example of Arbitrary Acceptance Condition

In [1]:
```
import spot
from spot.jupyter import display_inline
spot.setup()
```

In [2]:
```
a1 = spot.translate('GFa')
a2 = spot.translate('GFb')
a3 = spot.product_xor(a1, a2)
a4 = spot.acd_transform(a3, True)
display_inline(a1, a2, a3, a4)
```



Casares, Colcombet, and Fijalkow. Optimal transformations of games and automata using Muller conditions. *ICALP'21*. doi

Important Features Added Since 2.0

- Emptiness check of TELA with arbitrary acceptance.

Important Features Added Since 2.0

- Emptiness check of TELA with arbitrary acceptance.
- Translation from LTL to TELA (via Boolean decomposition of the formula).

Important Features Added Since 2.0

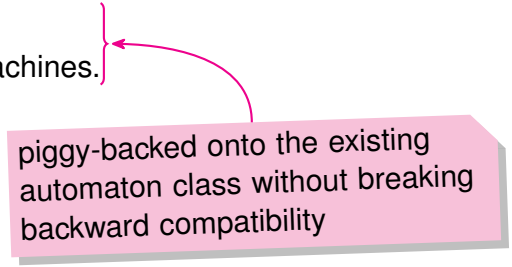
- Emptiness check of TELA with arbitrary acceptance.
- Translation from LTL to TELA (via Boolean decomposition of the formula).
- Several acceptance transformations and simplifications.


Renkin, Duret-Lutz, and Pommellet. Practical “paritizing” of Emerson-Lei automata. *ATVA'20*. doi


Casares et al. Practical applications of the alternating cycle decomposition. *TACAS'22*. doi

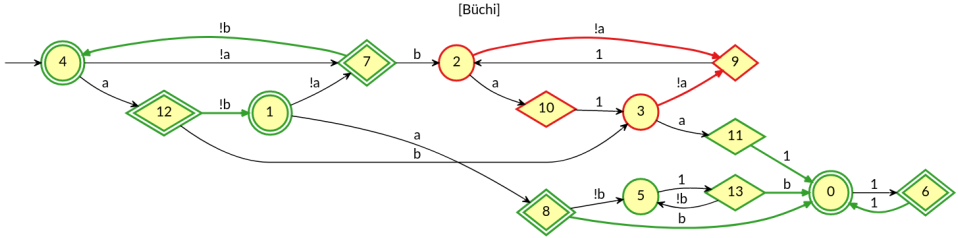
Important Features Added Since 2.0

- Emptiness check of TELA with arbitrary acceptance.
- Translation from LTL to TELA (via Boolean decomposition of the formula).
- Several acceptance transformations and simplifications.
- Support for alternating automata.
- Support for games and mealy machines.

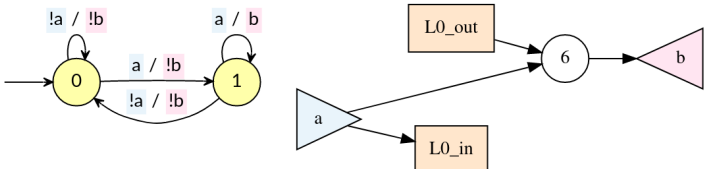


Support for Games and Mealy Machines

In [2]:
```
g = spot.automaton("ltlsynt --outs=b -f 'F(a&Xa)<->Fb' --print-game-hoa|")
spot.solve_game(g)
spot.highlight_strategy(g)
display(g)
```



In [3]:
```
m = spot.solved_game_to_separated_mealy(g)
spot.reduce_mealy_here(m, True)
aig = spot.mealy_machine_to_aig(m, "isop")
display_inline(m, aig.show('h'))
```



Typical Use-Cases

Are you a teacher?

- Use Jupyter to do exercices about LTL, ω-automata, games...
- Illustrate algorithms graphically. ‣ www
- Build a four-line model-checker. ‣ www

Are you a researcher?

- Answer questions about LTL formulas ‣ www or automata.
- Get tools for performing benchmarks.
- Prototype constructions in Python.
- Compare your algorithms to those in Spot (e.g. next talk!)

Are you a tool developer?

- Use Spot as a library to leverage existing algorithms.
- Implement new algorithms on top of existing data-structure.
- Use Python bindings for quick prototyping and debugging.
- Use Spot to test your tool ‣ ltlcross ‣ autcross.

Typical Use-Cases

Are you a teacher?

- Use Jupyter to do exercices about LTL, ω-automata, games...
- Illustrate algorithms graphically. ‣ www
- Build a four-line model-checker. ‣ www

Are you a paper author?

- Please write Spot's version number!

Are you a researcher?

- Answer questions about LTL formulas ‣ www or automata.
- Get tools for performing benchmarks.
- Prototype constructions in Python.
- Compare your algorithms to those in Spot (e.g. next talk!)

Are you a tool developer?

- Use Spot as a library to leverage existing algorithms.
- Implement new algorithms on top of existing data-structure.
- Use Python bindings for quick prototyping and debugging.
- Use Spot to test your tool ‣ ltlcross ‣ autcross.

Availability & Demo

- Licensed as GPLv3.
- Source code, documentation, examples, at

  `https://spot.lrde.epita.fr/`
- Packages for Debian, RedHat, Conda (Linux & MacOS).
- Third-party packages exist for Arch, FreeBSD.

Tool demonstration tomorrow (Wednesday) at 13:00–13:30 in the Taub entrance floor lobby.