Two-Automaton Emptiness Check in Spot

Clément Gillard

LRDE
EPITA Research and Development Laboratory

Seminar – 2017-07-04

Two-Automaton Emptiness Check in Spot

1. Spot and the model checking toolchain
2. Existing implementations
3. New implementation
4. Conclusion

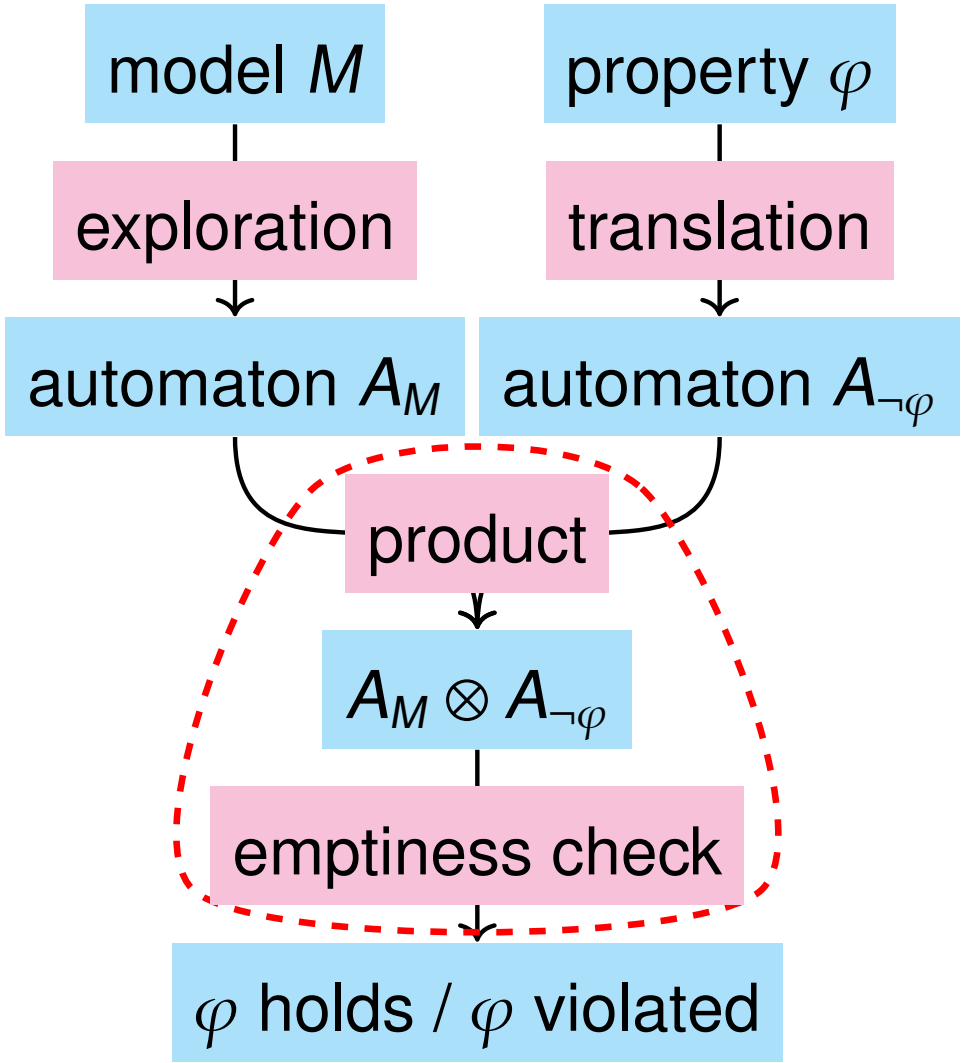
Spot and the model checking toolchain

1. Spot and the model checking toolchain
2. Existing implementations
3. New implementation
4. Conclusion

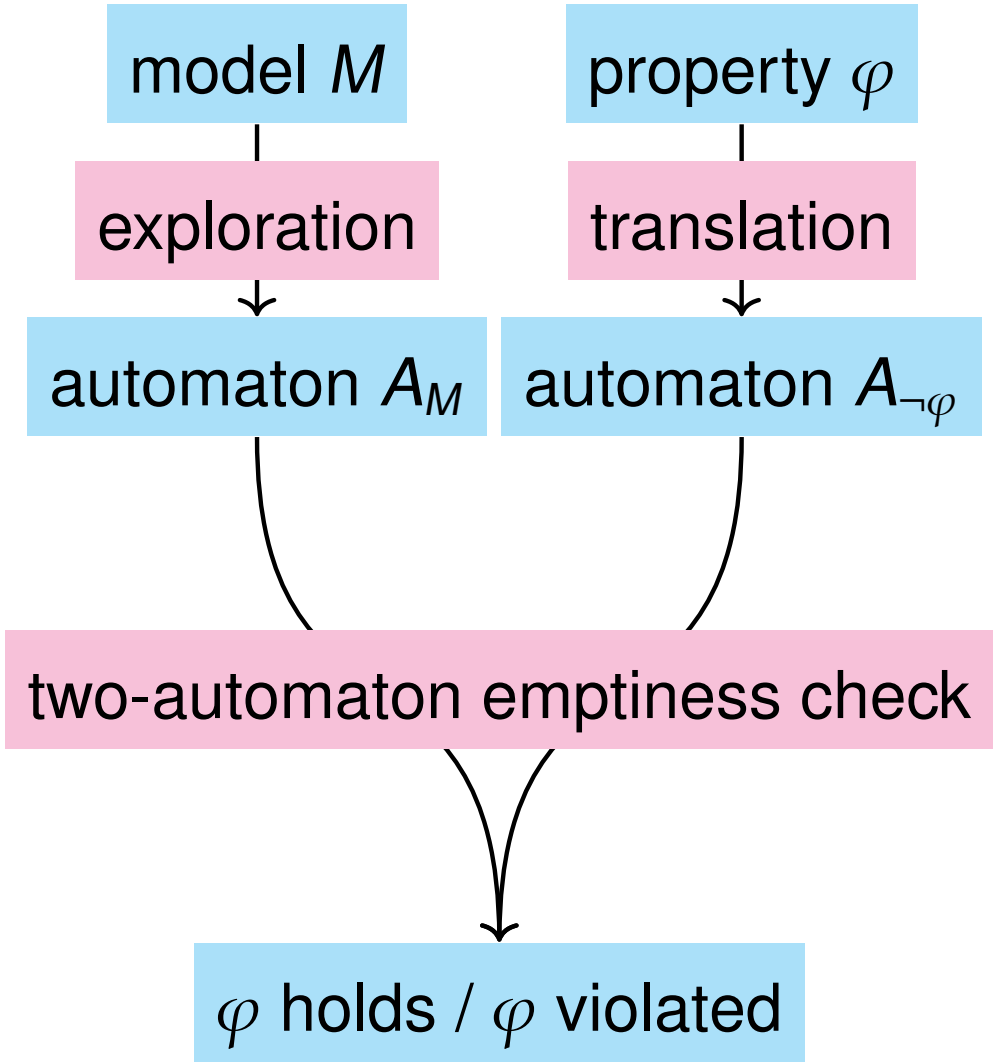
Spot

- C++ library [1]
- ω-automata for model checking
- on-the-fly and explicit automata
- Kripke structures, strength decomposition [2] …

The model checking toolchain



The model checking toolchain



Existing implementations

1. Spot and the model checking toolchain
2. Existing implementations
3. New implementation
4. Conclusion

Existing code

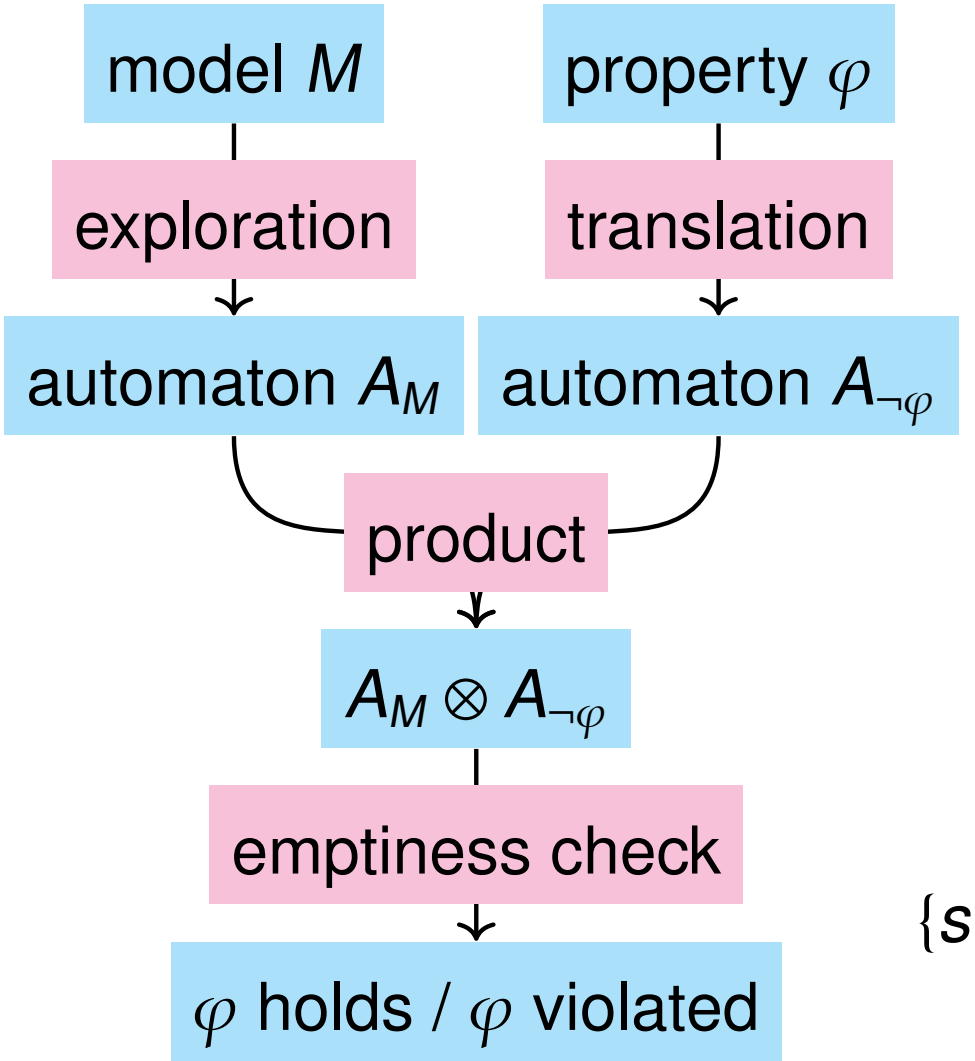
In `twa::intersects`:

```
return !otf_product(a, b)->is_empty();
```

`otf_product` on-the-fly product, uses on-the-fly interface, returns an on-the-fly automaton, optimized on Kripke structures,

`is_empty` Couvreur [3], uses on-the-fly or explicit interface, optimized on expliciteness and strength.

Existing code



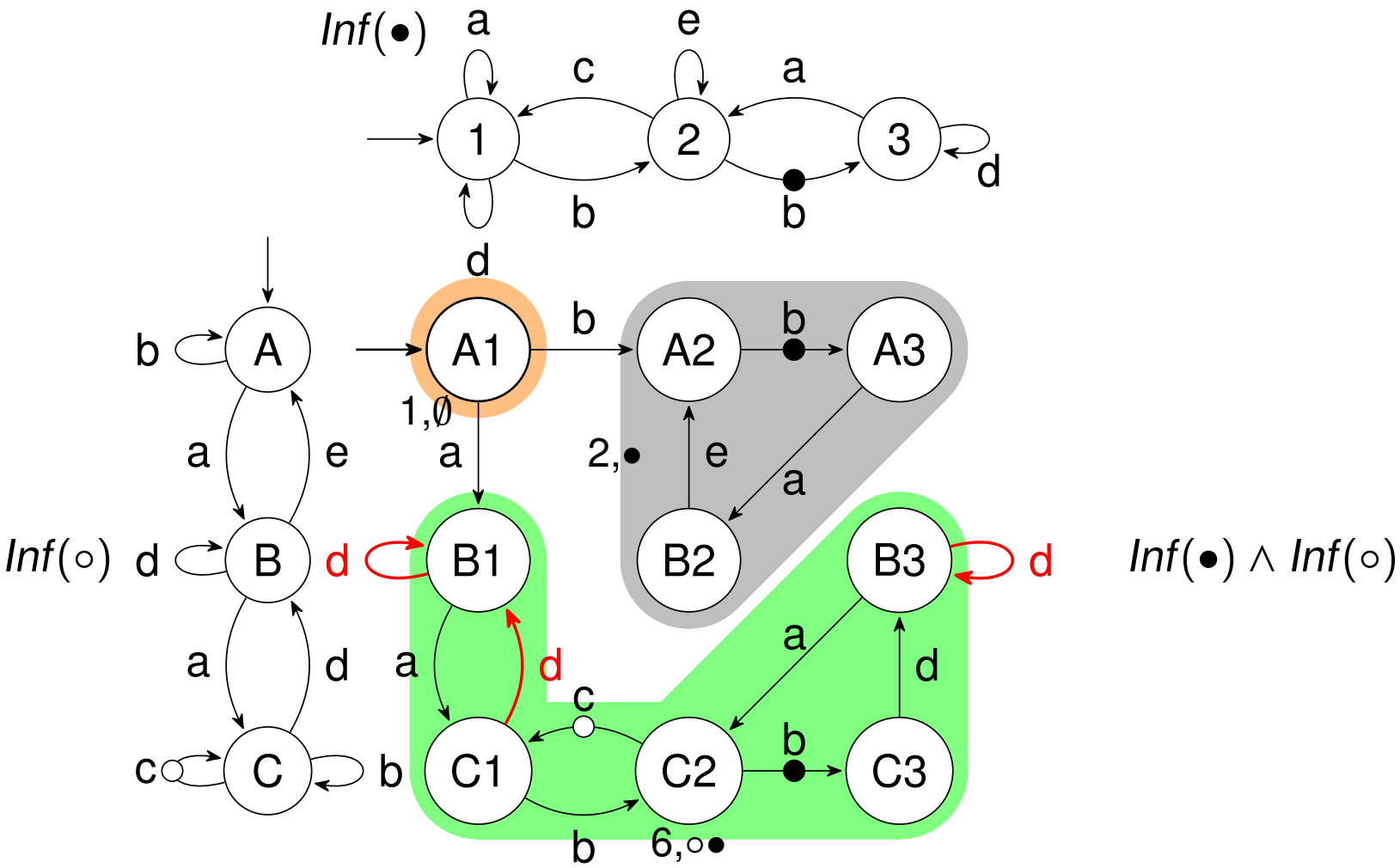
$\{otf, kripke\}$

$\{otf, explicit\}$
$\times$
$\{strong, weak, terminal\}$

New implementation

1. Spot and the model checking toolchain
2. Existing implementations
3. New implementation
4. Conclusion

Unrolling of the algorithm



Optimizations

Explicit automata:

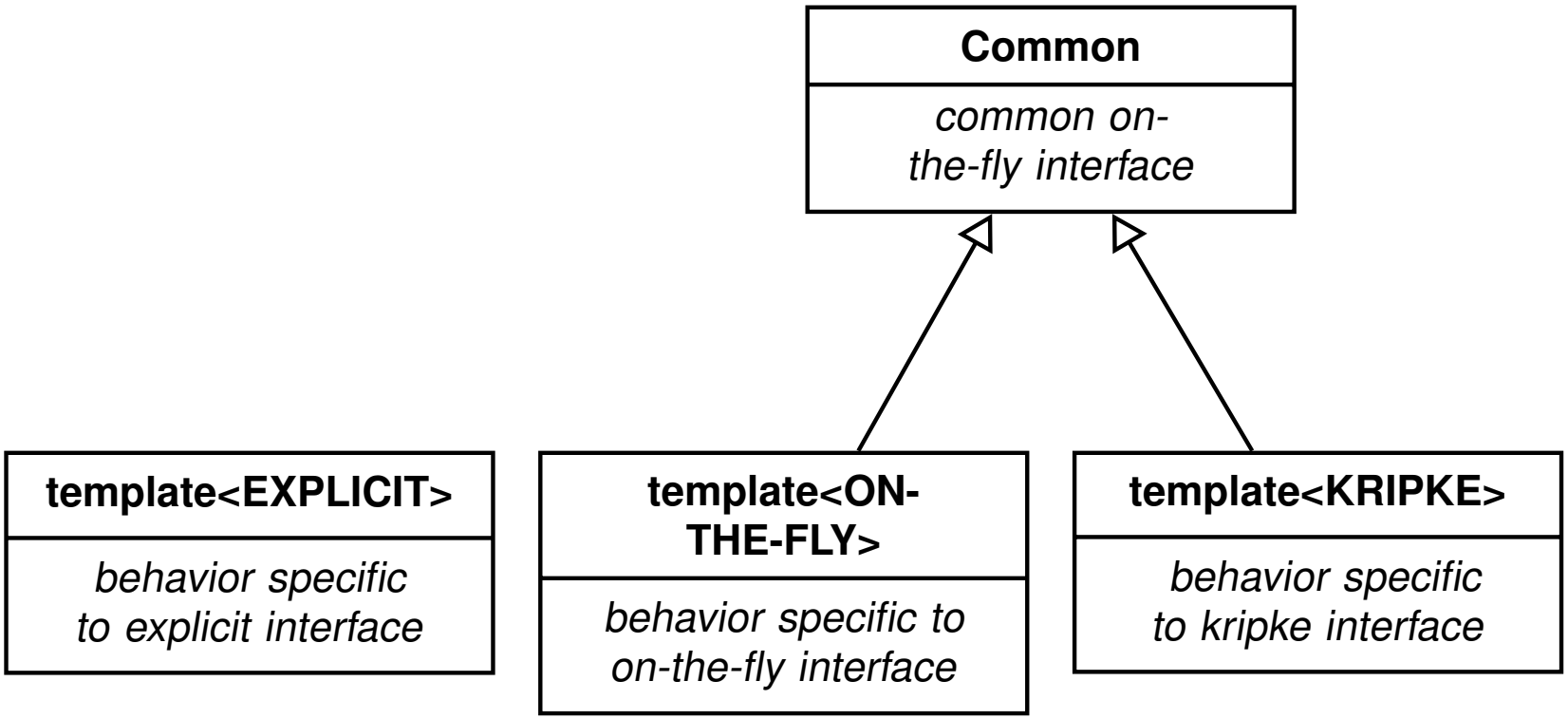
- template specialization of iterator and states
- reproduce behavior of on-the-fly iterators with explicit interface
- same with states
- improvements in structure size and method calls

Optimizations

Kripke and Fair-Kripke structures:

- very similar to on-the-fly
- “No dynamic dispatch!”
- solution: inheritance of common behavior in templates
- improvements in method calls

Optimizations



Optimizations

Strength of SCCs:

- lighter structures
- what you store is not what you use
- code uses STRONG-STRONG, structures stores STRONG-STRONG, WEAK-STRONG, or is unused
- conversions lose data, operations gain data
- improvements in structure size and usage

Optimizations

$$\{explicit, on\text{-}the\text{-}fly, kripke\}$$
$$\times$$
$$\{explicit, on\text{-}the\text{-}fly\}$$
$$\times$$
$$\{strong - strong, weak - strong, weak - weak\}$$

18 instances

Conclusion

1. Spot and the model checking toolchain
2. Existing implementations
3. New implementation
4. Conclusion

Benchmarks

- Explicitness:
  - 50 pairs
  - 200 states
  - random combination of 16 acceptance sets
  - 10 boolean variables

  two-automaton emptiness check: 1595s
  `otf_product` + `is_empty`: 1750s
- Strength: no significant result
- Kripke: not yet

Conclusion

What remains to do:

- return counter example

Questions?


Alexandre Duret-Lutz, Alexandre Lewkowicz, Amaury Fauchille, Thibaud Michaud, Etienne Renault, and Laurent Xu.
Spot 2.0 — a framework for LTL and ω-automata manipulation.
In *Proceedings of the 14th International Symposium on Automated Technology for Verification and Analysis (ATVA'16)*, volume 9938 of *Lecture Notes in Computer Science*, pages 122–129. Springer, October 2016.


Etienne Renault, Alexandre Duret-Lutz, Fabrice Kordon, and Denis Poitrenaud.
Strength-based decomposition of the property Büchi automaton for faster model checking.
In Nir Piterman and Scott A. Smolka, editors, *Proceedings of the 19th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'13)*, volume 7795 of

Lecture Notes in Computer Science, pages 580–593. Springer, March 2013.

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