Simulation Reductions for TGBA

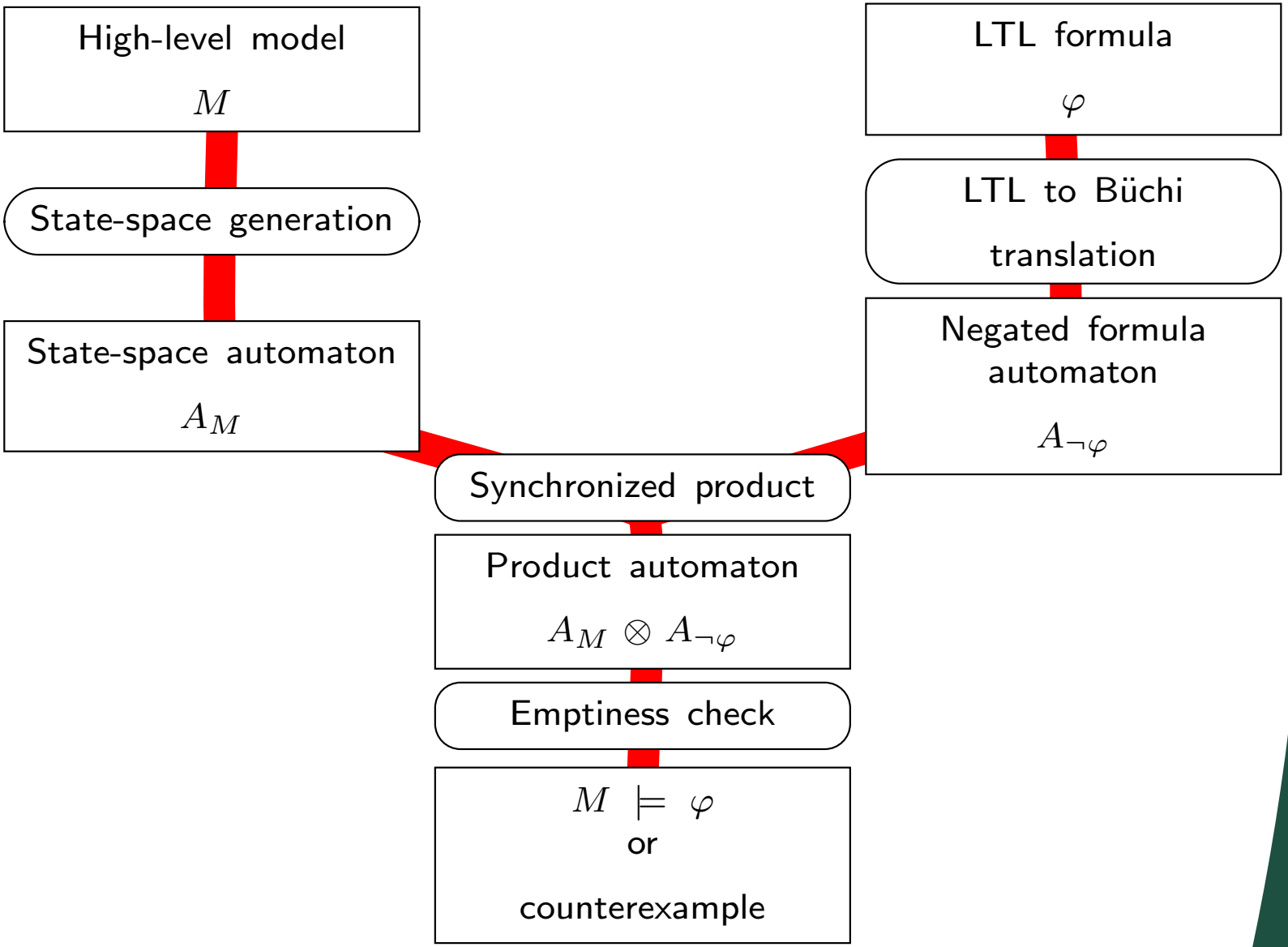
Thomas Badie

LRDE
Laboratoire de Recherche et Développement de l'EPITA

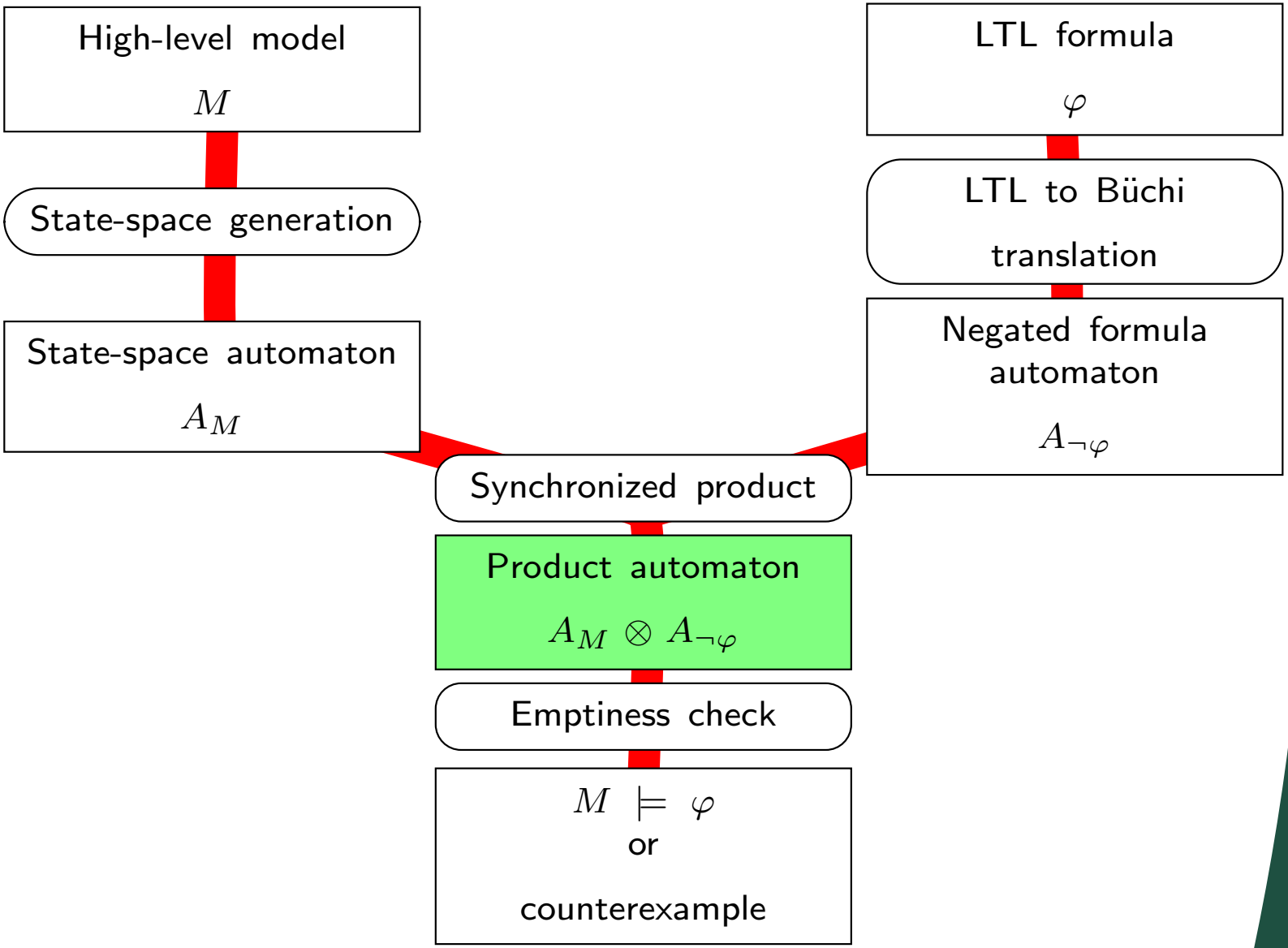
February 6, 2013

http://lrde.epita.fr/

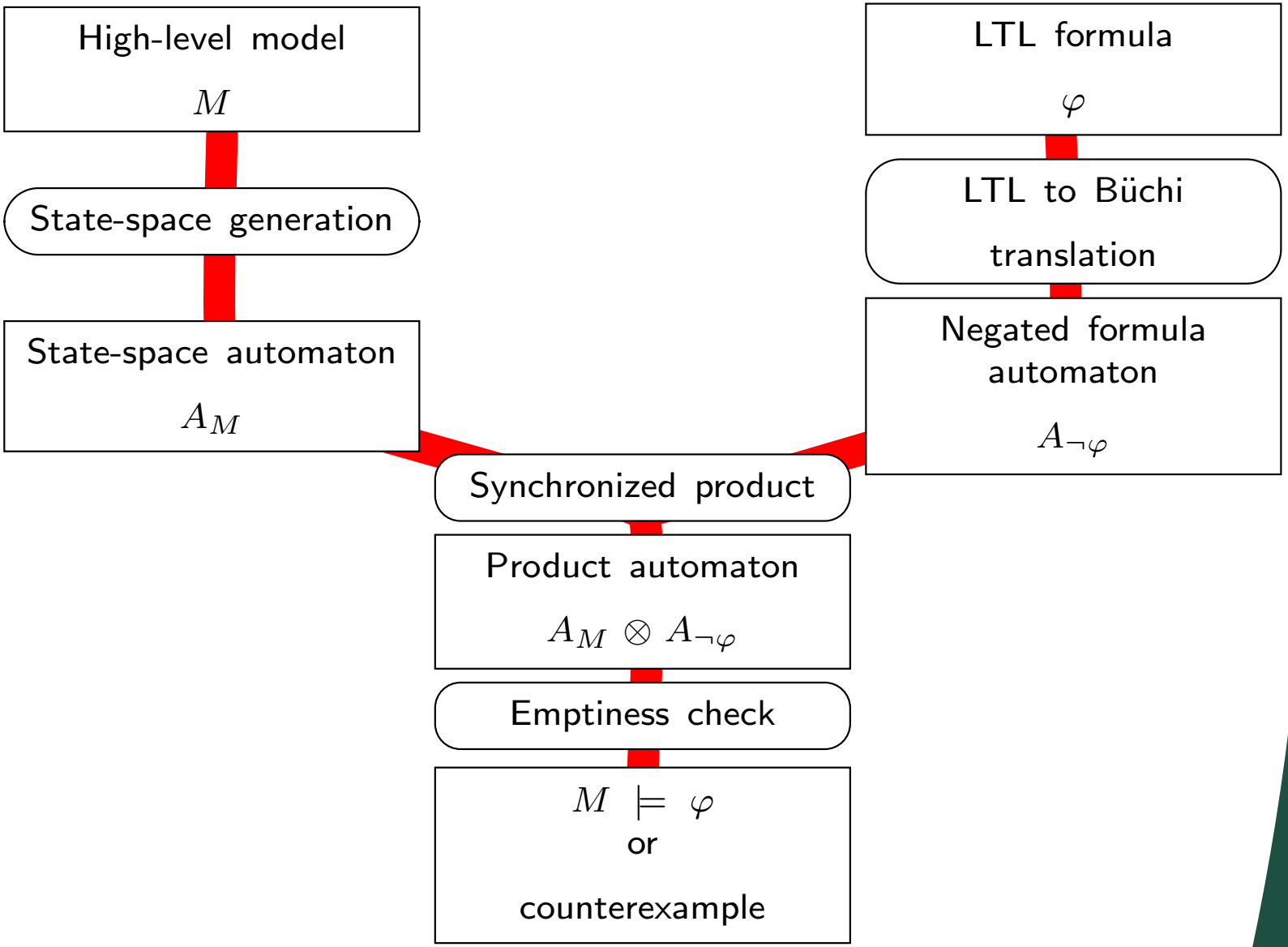
SPOT and Model Checking



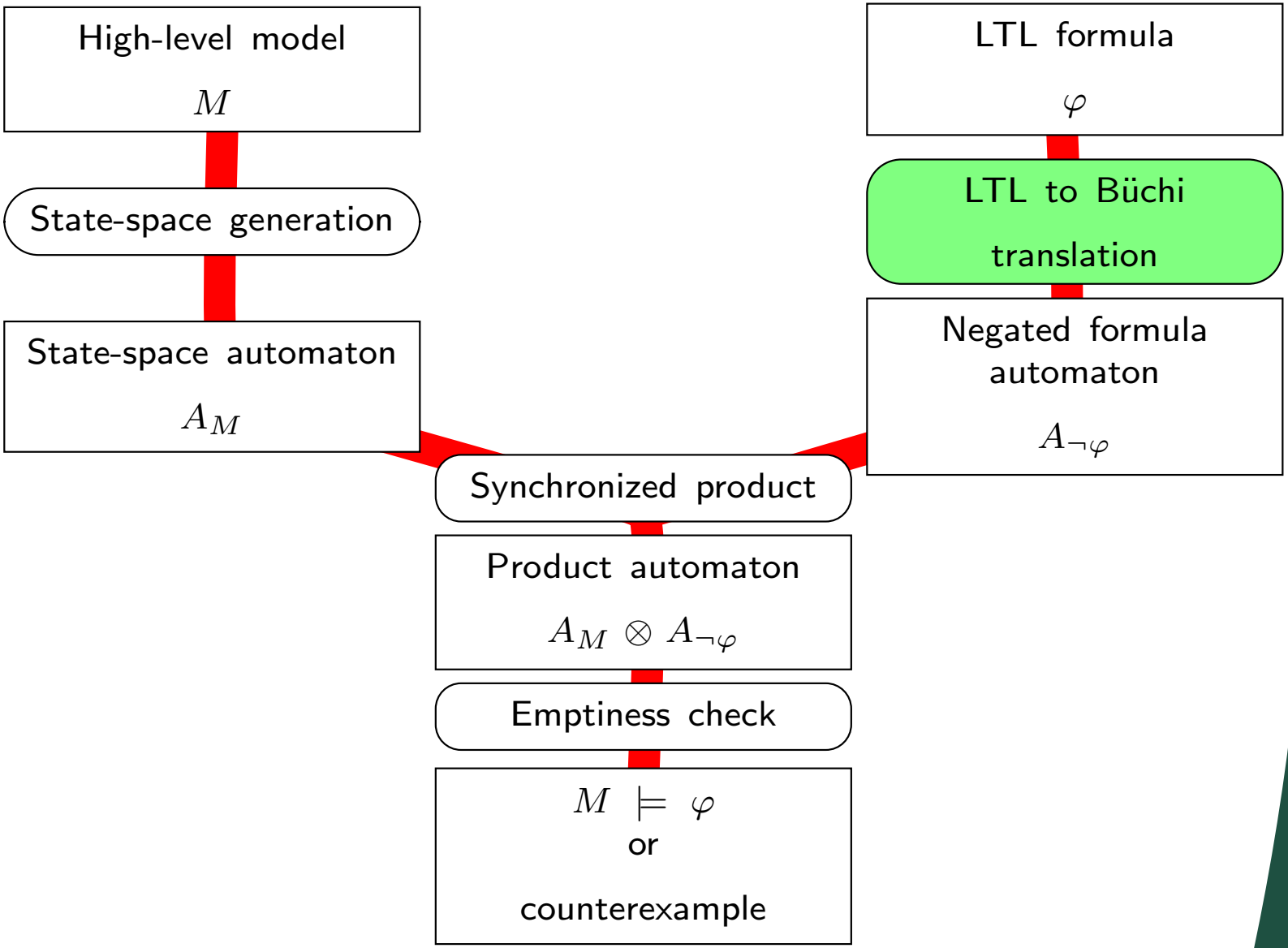
SPOT and Model Checking



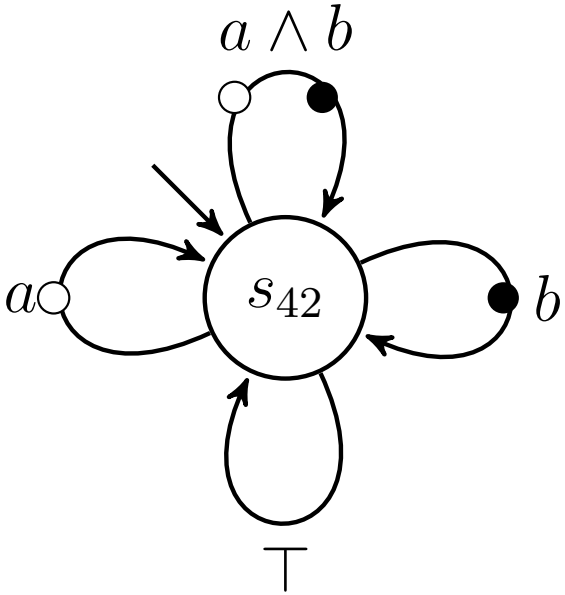
SPOT and Model Checking



SPOT and Model Checking

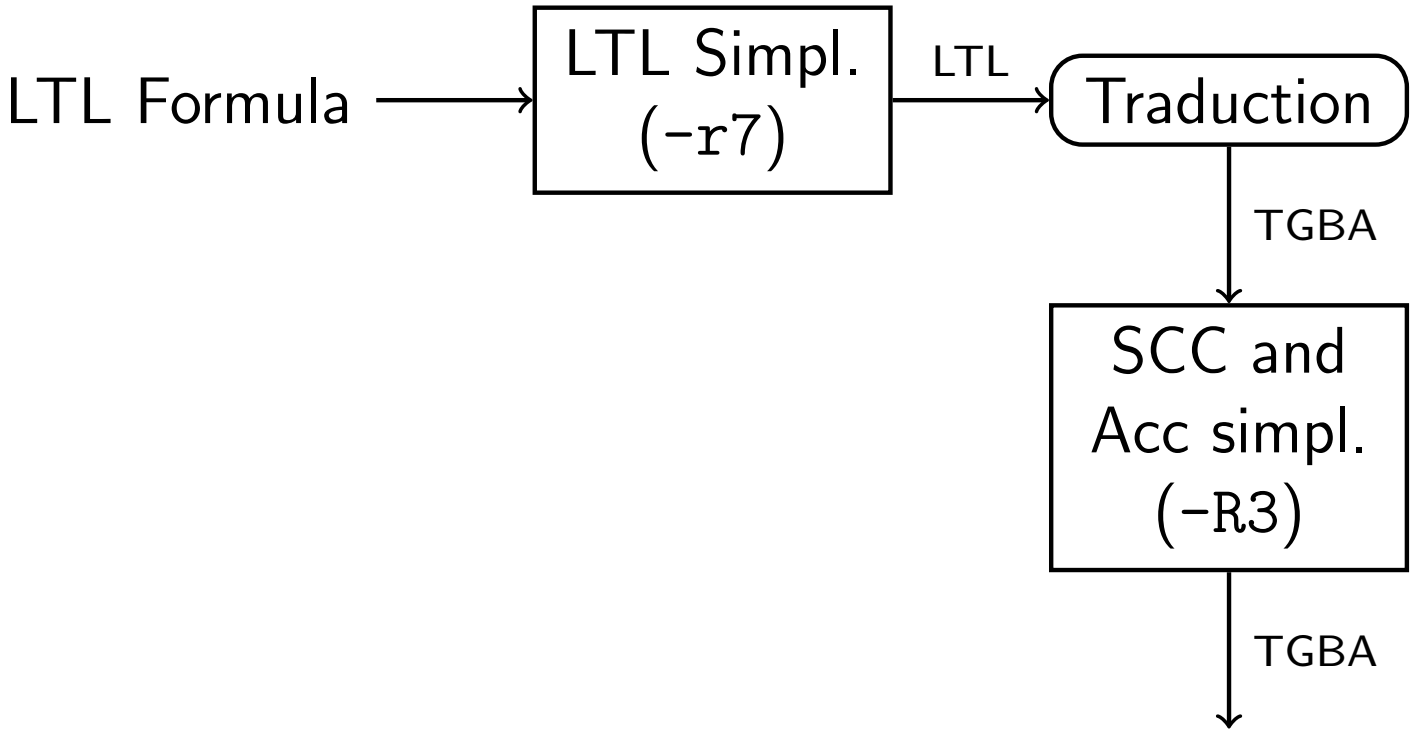


TGBA for $\mathsf{G}\,\mathsf{F}\,a \wedge \mathsf{G}\,\mathsf{F}\,b$

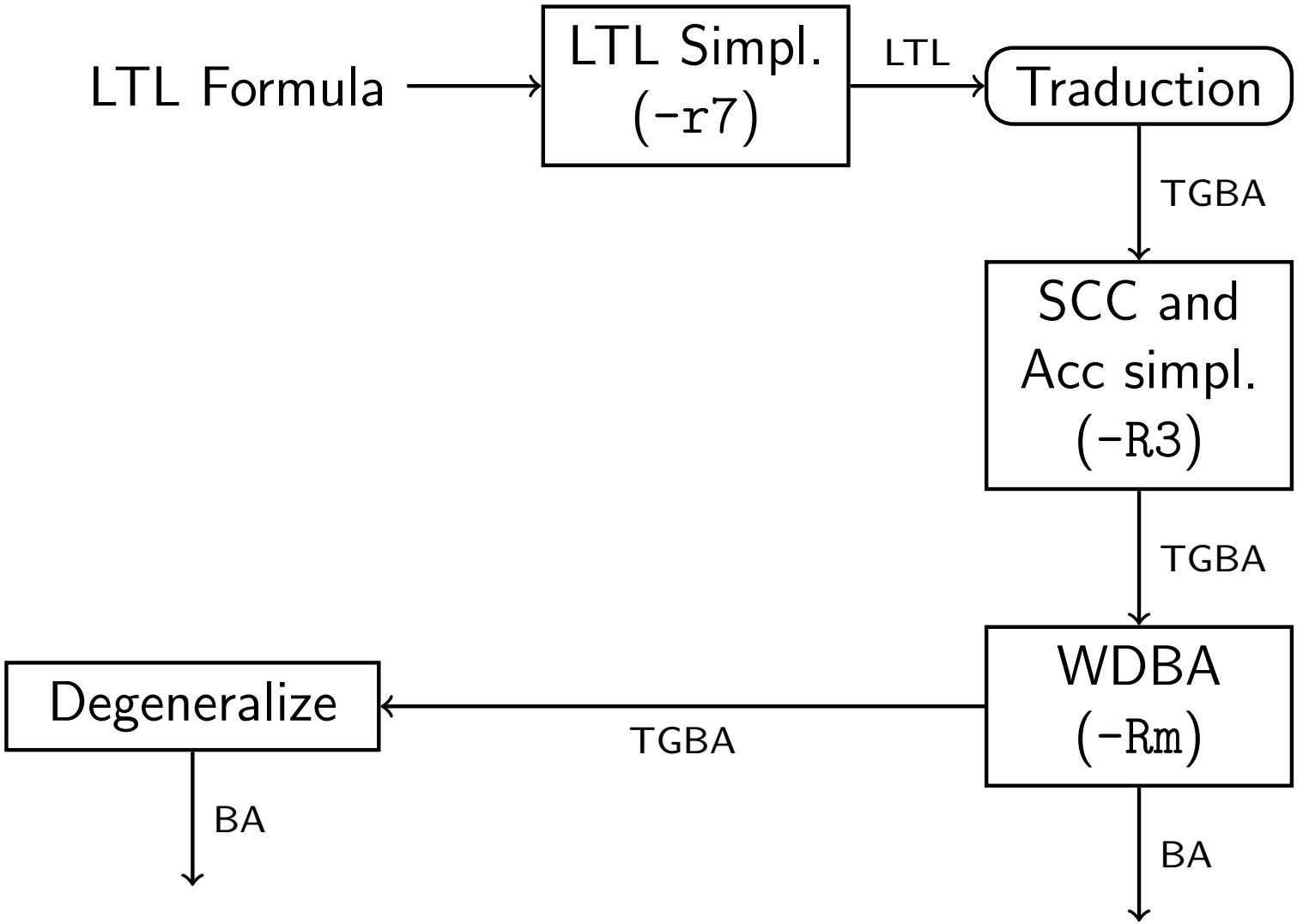


a: 000100010001...
b: 010101010101...

LTL to Büchi Translation

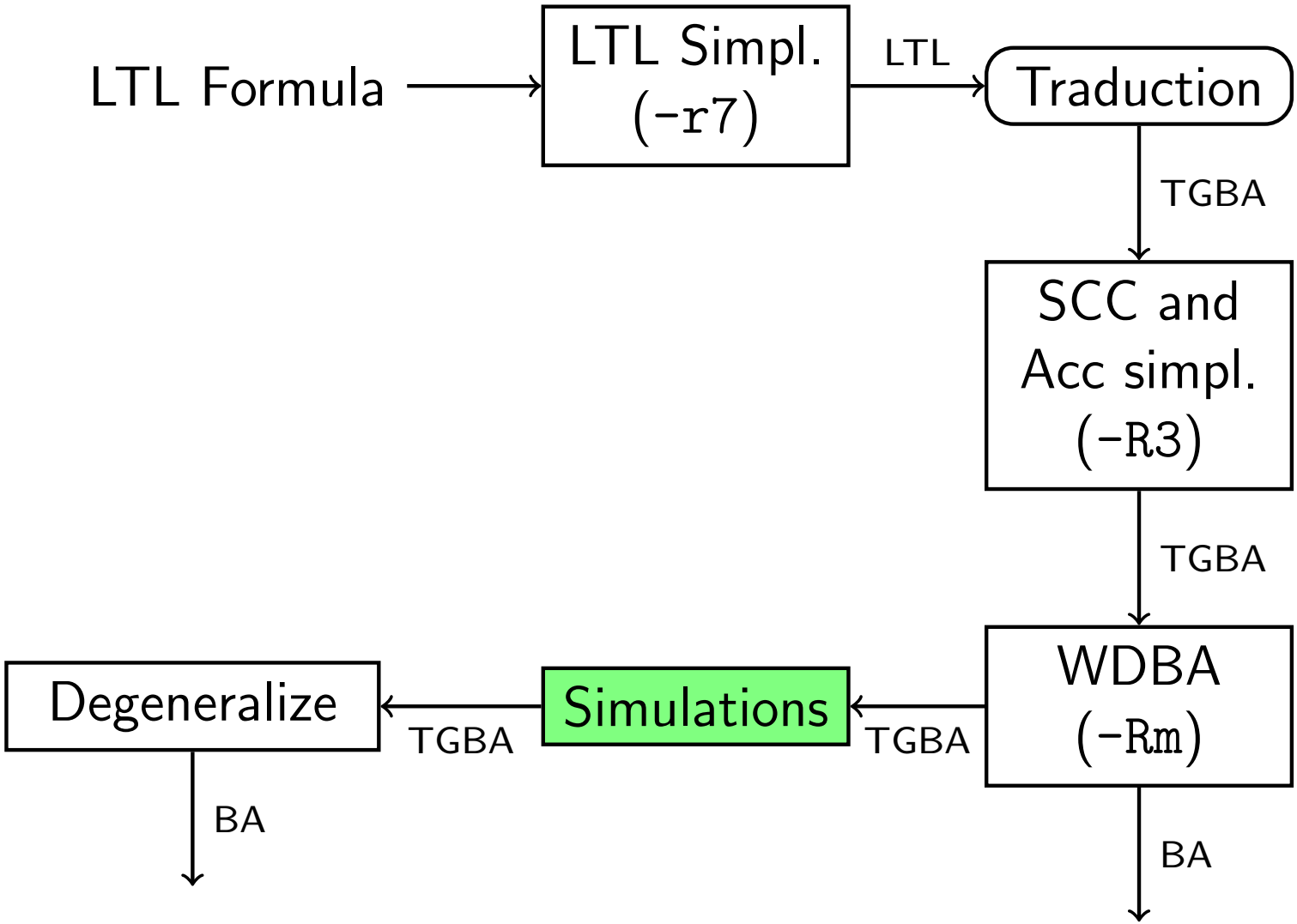


LTL to Büchi Translation



Thomas Badie

LTL to Büchi Translation



Outline

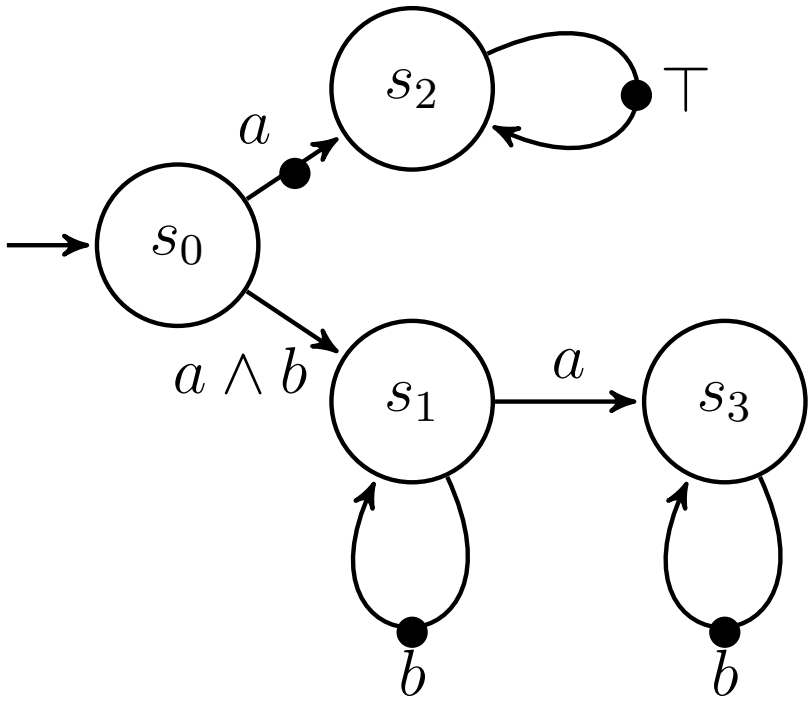
Direct Simulation

Reverse Simulation

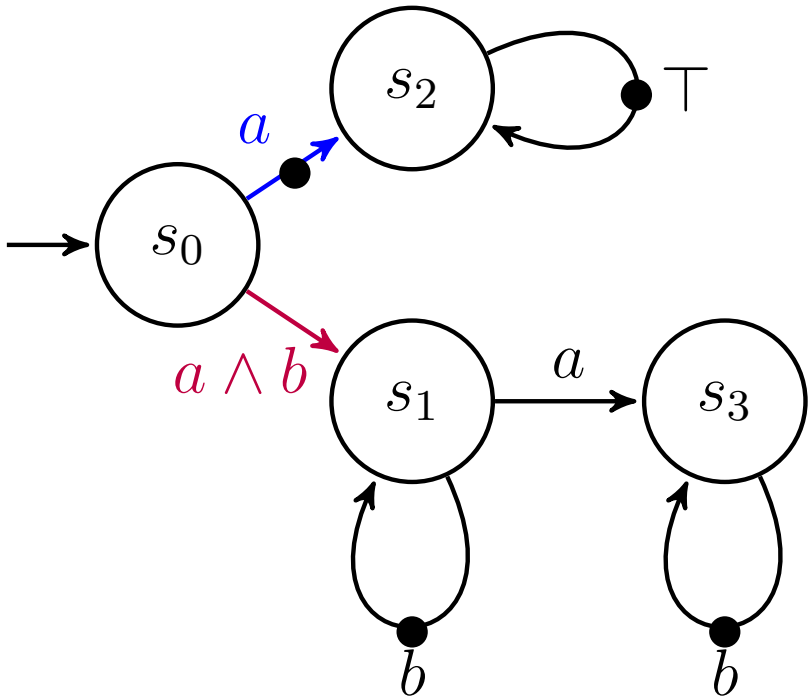
Iterated Simulation

Don't Care Simulation

Direct Simulation = Suffix Inclusion



Direct Simulation with Signatures

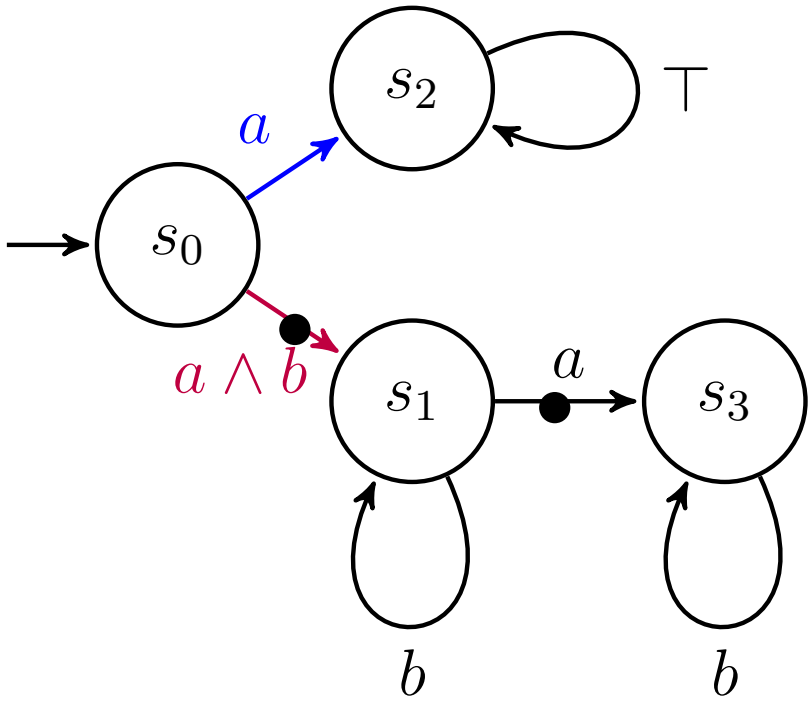


Simulation Relation

- t_1.cond $\Rightarrow$ t_2.cond
- t_1.acc $\Leftarrow$ t_2.acc
- $\mathcal{C}(t_1.\text{dest}) \Rightarrow \mathcal{C}(t_2.\text{dest})$

Direct Simulation with Signatures

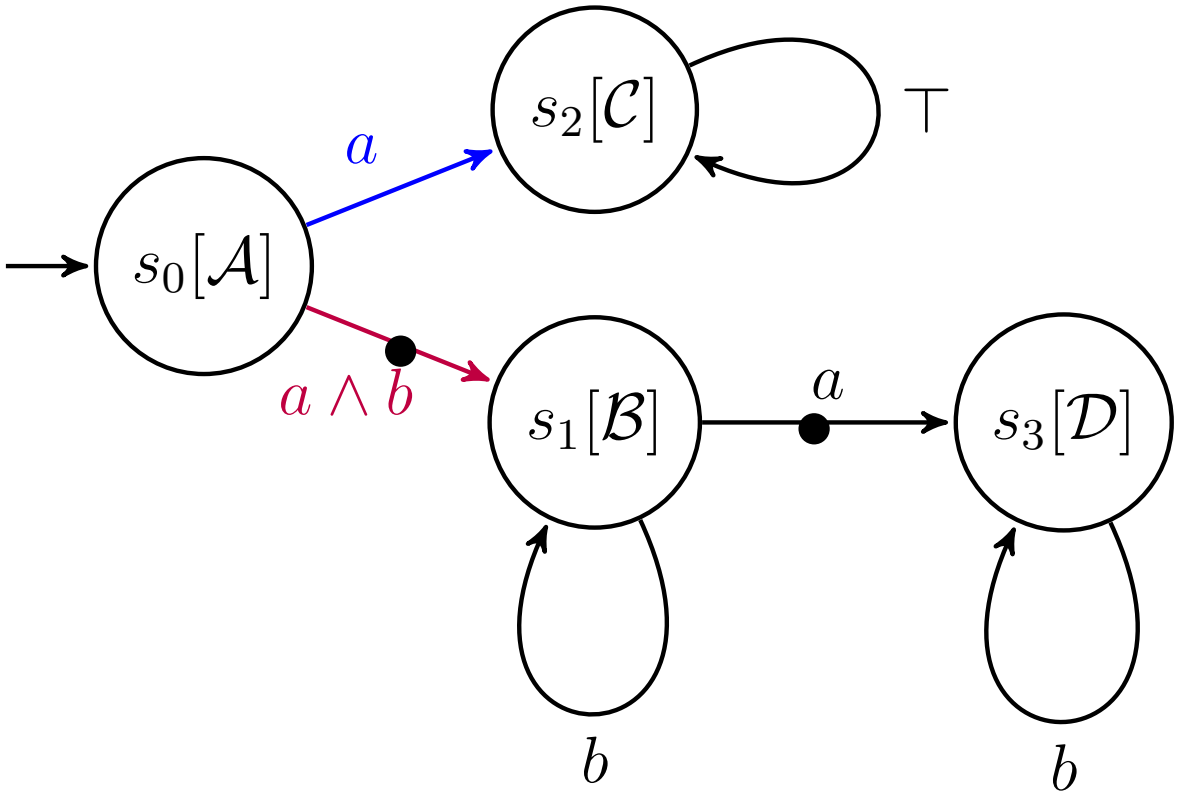
Promises Automaton.



Simulation Relation

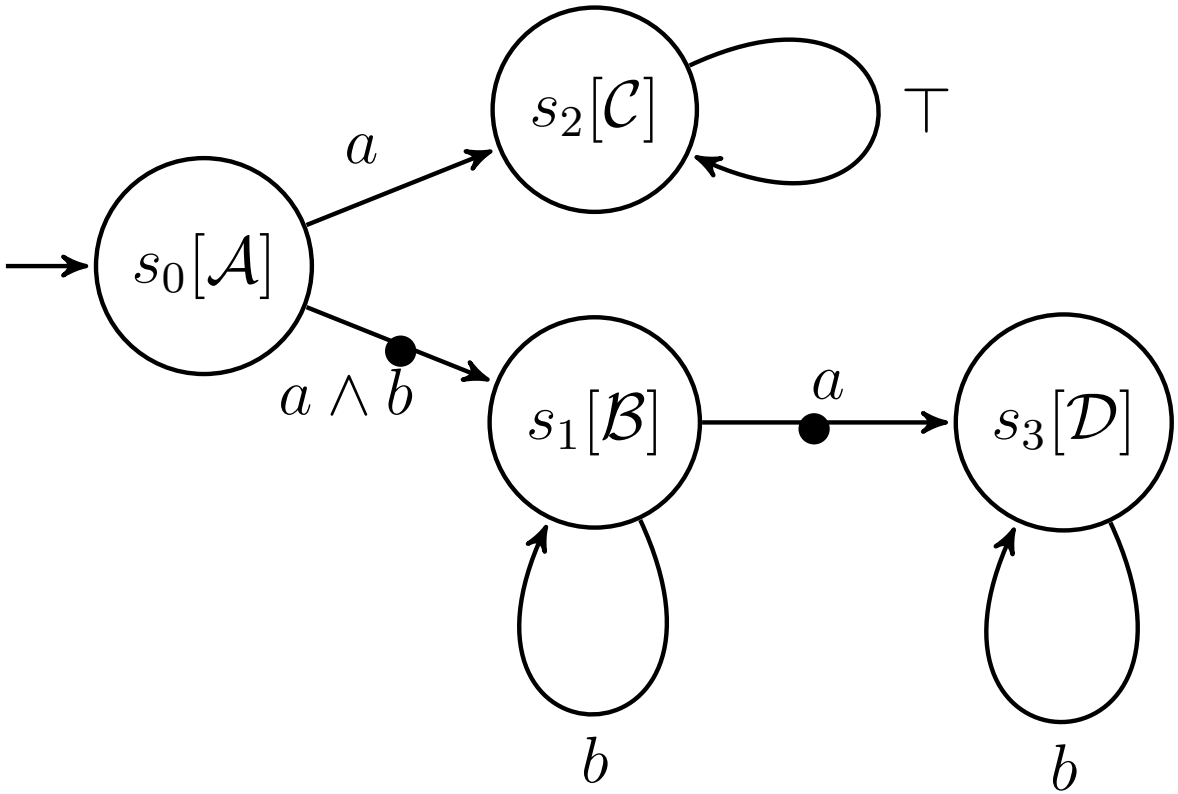
- t_1.cond $\Rightarrow$ t_2.cond
- t_1.acc $\Rightarrow$ t_2.acc
- $\mathcal{C}(t_1.\text{dest}) \Rightarrow \mathcal{C}(t_2.\text{dest})$

Example



$$\mathsf{sig}(s_0) = (a \wedge \mathcal{C}) \vee (a \wedge b \wedge \bullet \wedge \mathcal{B} \wedge \mathcal{C})$$

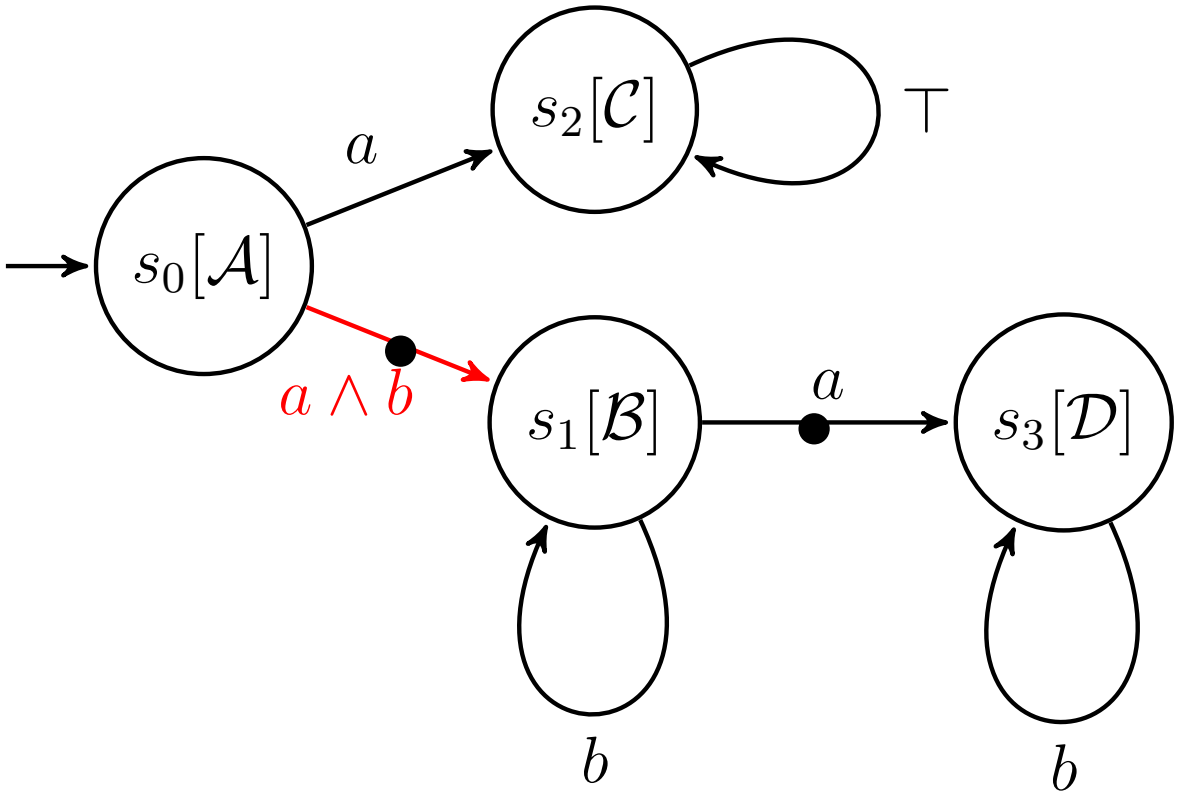
Example



$$\mathsf{sig}(s_0) = (a \wedge \mathcal{C}) \vee (a \wedge b \wedge \bullet \wedge \mathcal{B} \wedge \mathcal{C})$$

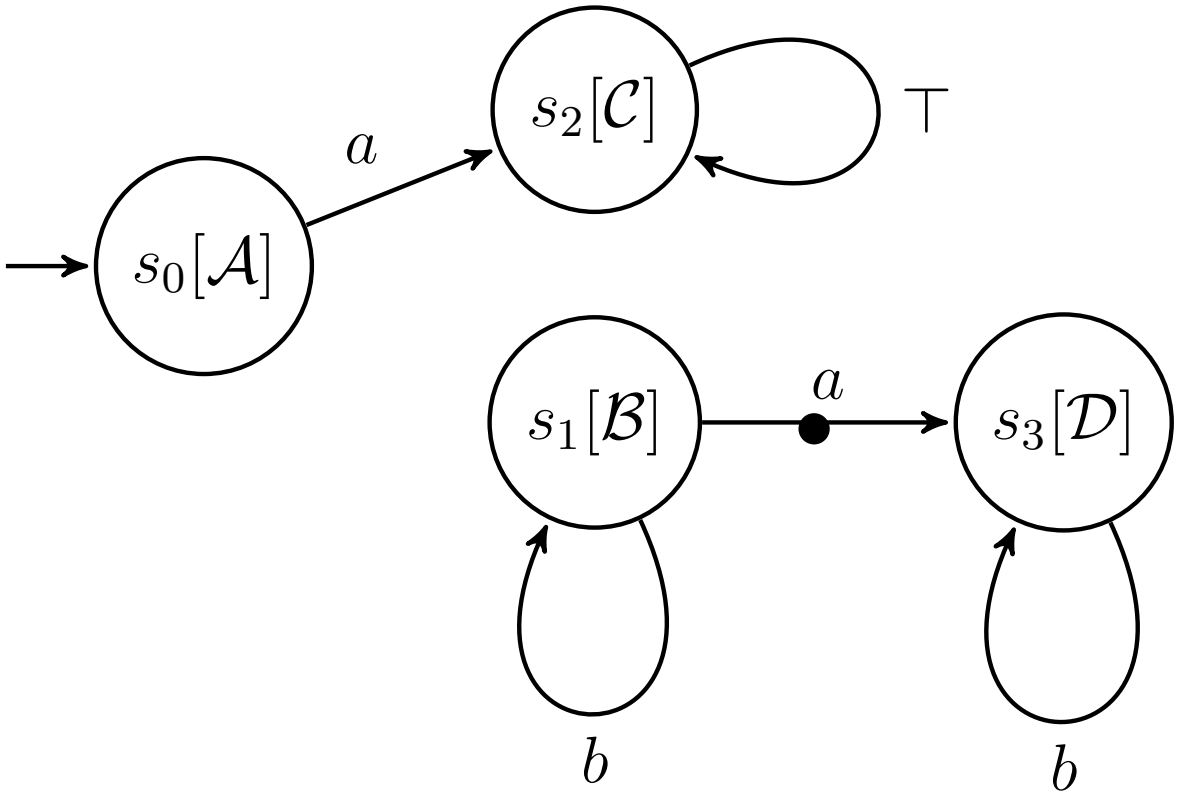
Thomas Badie

Example



$$\mathsf{sig}(s_0) = (a \wedge \mathcal{C}) \vee (a \wedge b \wedge \bullet \wedge \mathcal{B} \wedge \mathcal{C})$$

Example



$$\mathsf{sig}(s_0) = (a \wedge \mathcal{C})$$

Algorithm

- Translate the TGBA into a Promise Automaton;

Algorithm

- Translate the TGBA into a Promise Automaton;
- Compute the signature of each state;

Algorithm

- Translate the TGBA into a Promise Automaton;
- Compute the signature of each state;
- Put states into classes according to their signatures;

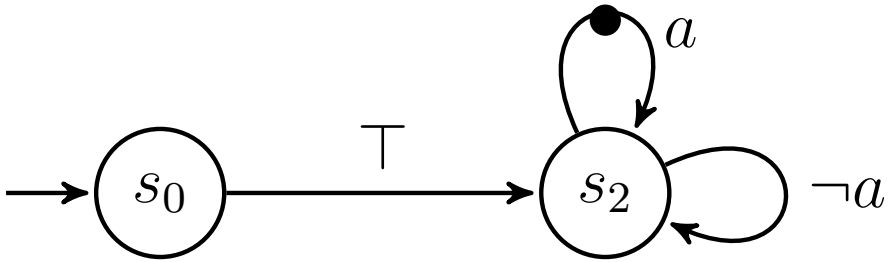
Algorithm

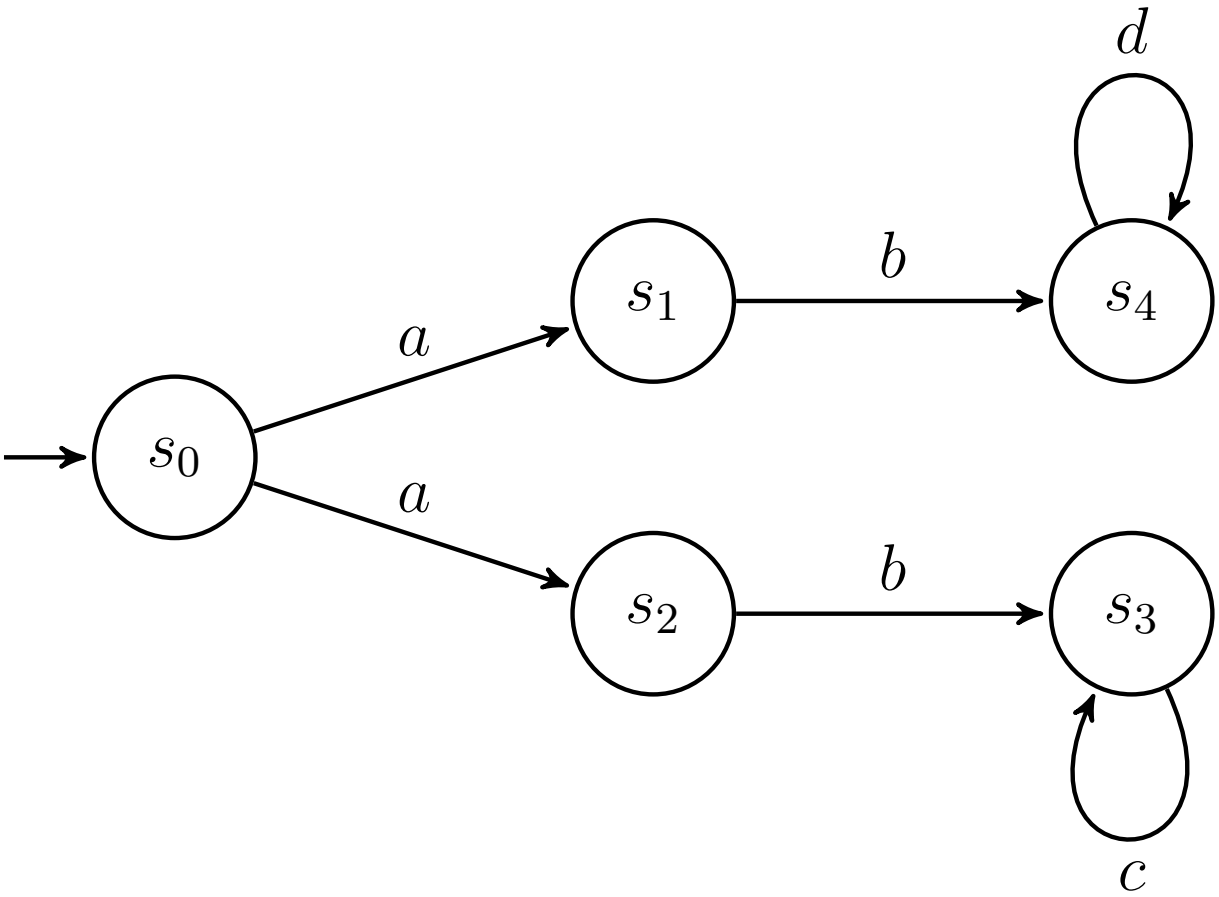
- Translate the TGBA into a Promise Automaton;
- Compute the signature of each state;
- Put states into classes according to their signatures;
- Compute an implication relation between the classes;

Algorithm

- Translate the TGBA into a Promise Automaton;
- Compute the signature of each state;
- Put states into classes according to their signatures;
- Compute an implication relation between the classes;
- Iterate until fixpoint.

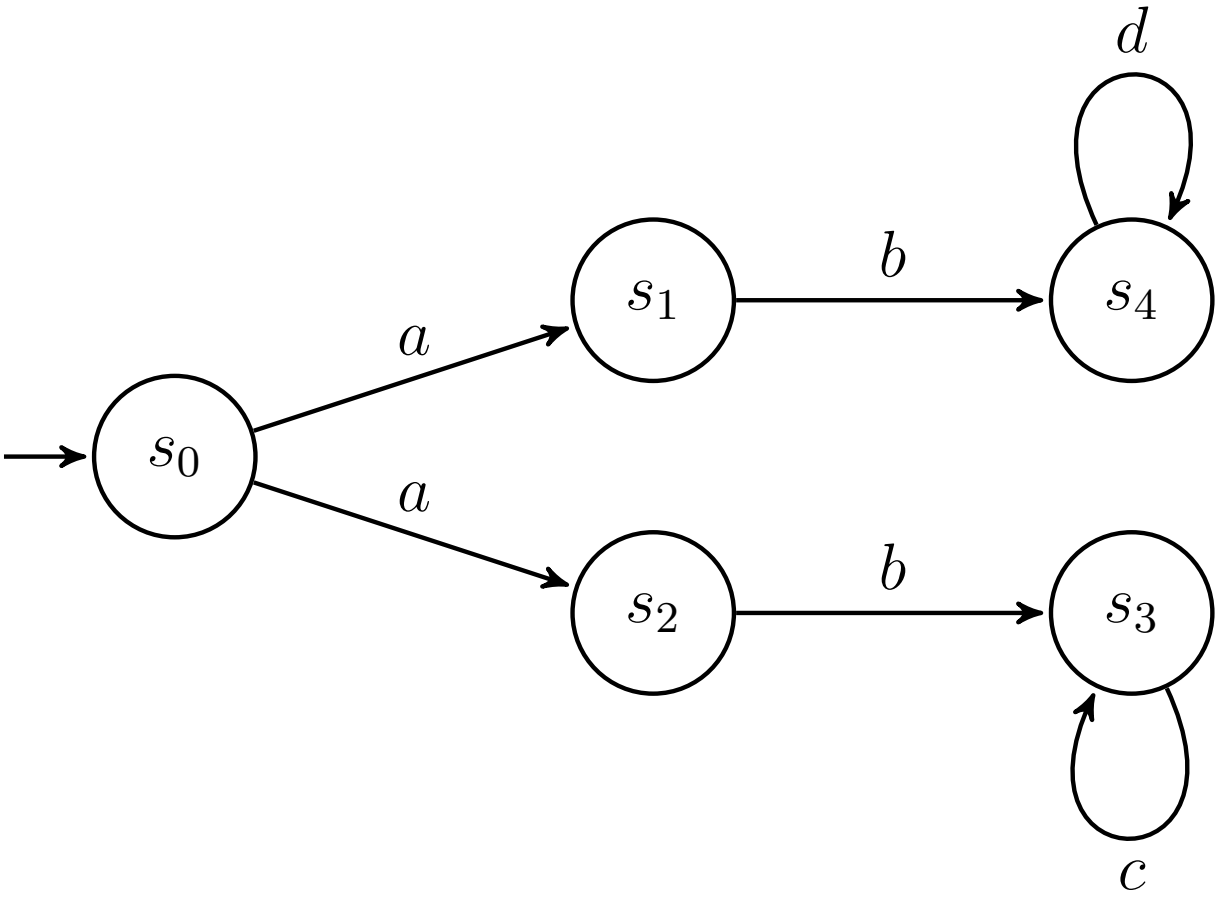
Simulation limits





Thomas Badie

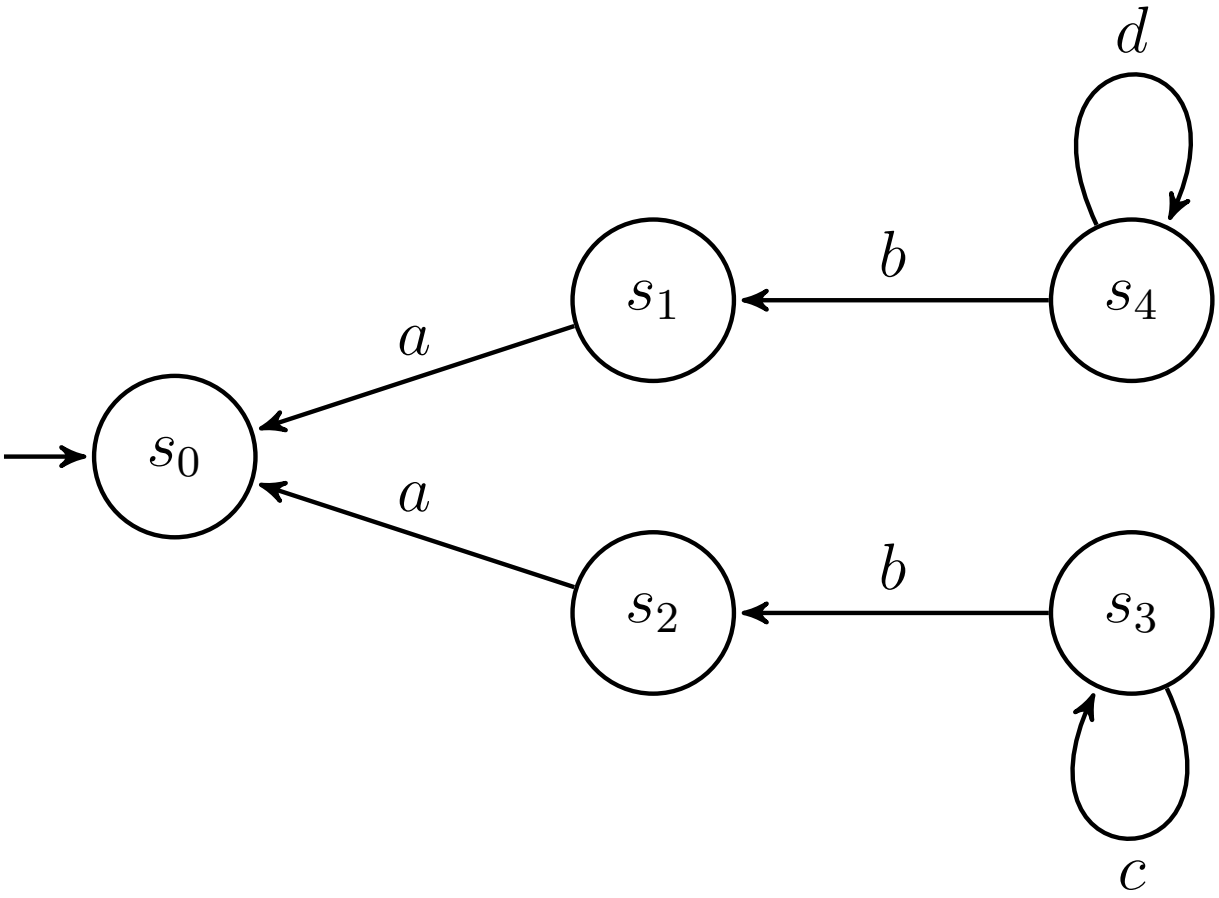
Reverse Simulation = Prefix Inclusion



Transposing a TGBA is complex

- SPOT does not support several initial states;
- A TGBA does not have final states.

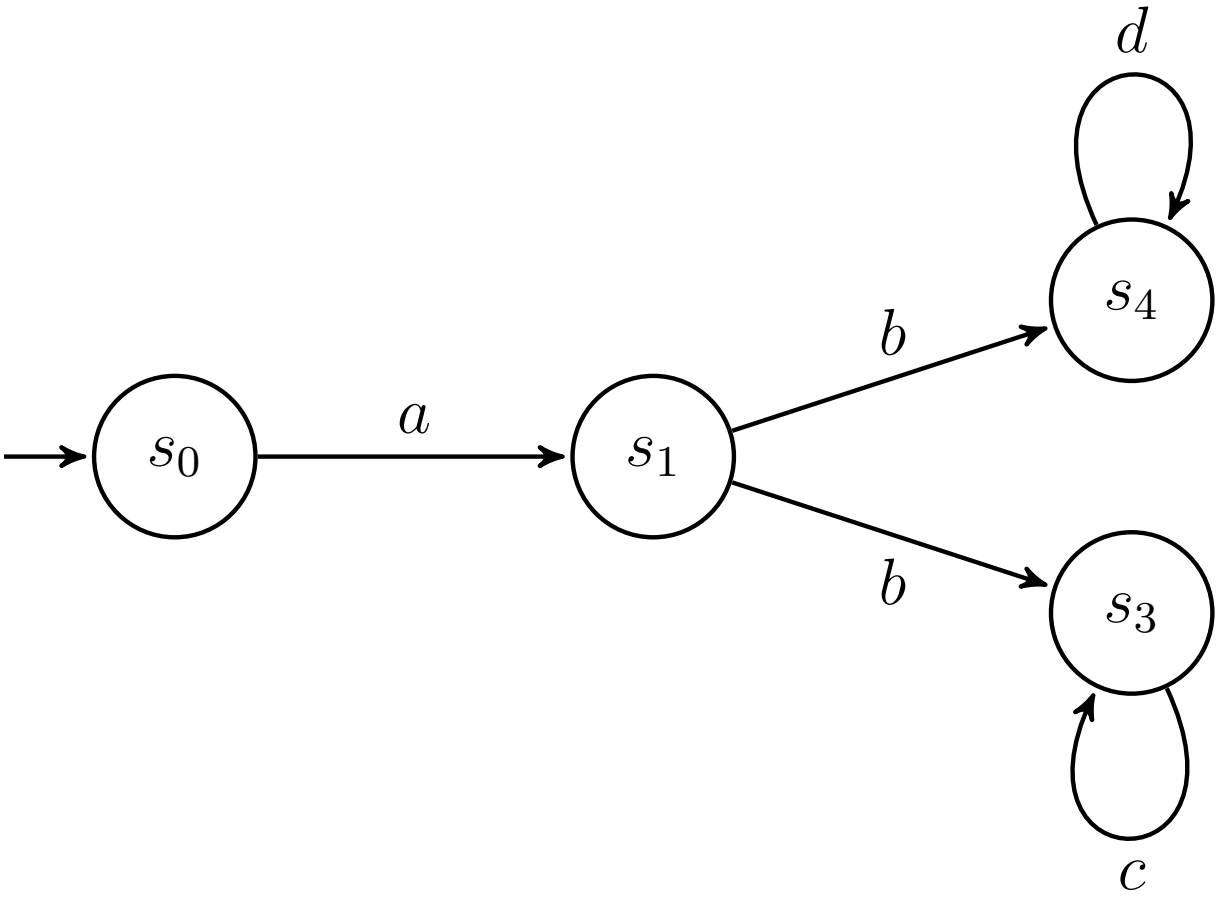
Reverse Simulation = Prefix Inclusion



Transposing a TGBA is complex

- SPOT does not support several initial states;
- A TGBA does not have final states.

Reverse Simulation = Prefix Inclusion



Transposing a TGBA is complex

- SPOT does not support several initial states;
- A TGBA does not have final states.

Iterated Simulation

Idea

- Reverse Simulation reduces cases the simulation can not. And vice-versa;
- We can run the two until we reach a fixpoint;
- Adding `scc_filter` allows to remove some useless SCC left by the reverse simulation.

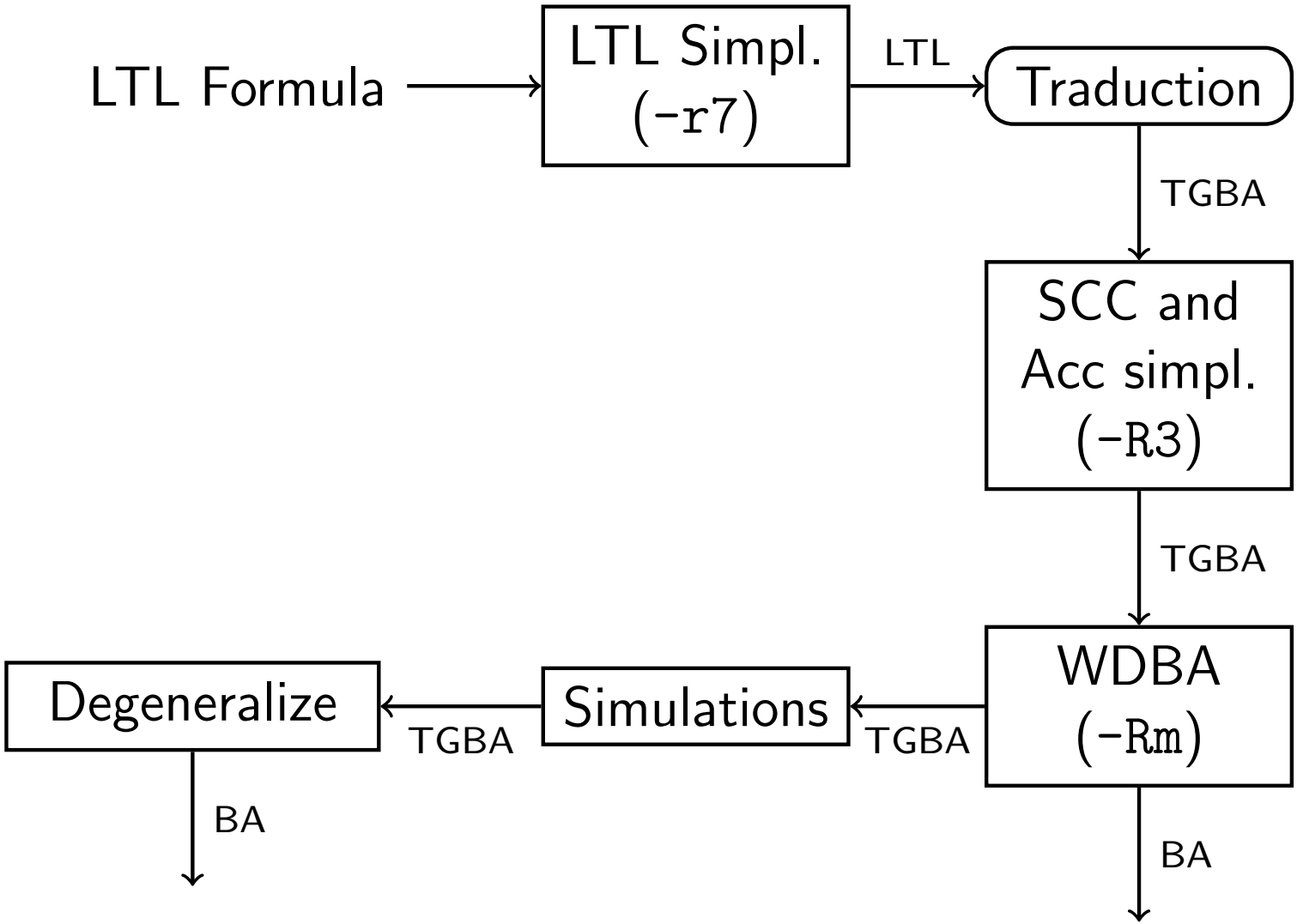
Benchmark Parameters

What we test

- We test with options `-r7`, `-R3`, `-Rm` ; On 188 formulae from the literature [1, 2, 3] and we present a cumulative result;
- Algorithms:

No Sim	No more options than the ones described above;
RDS	The Direct Simulation;
RRS	The Reverse Simulation;
RIS	The Iterated Simulation.

LTL to Büchi Translation

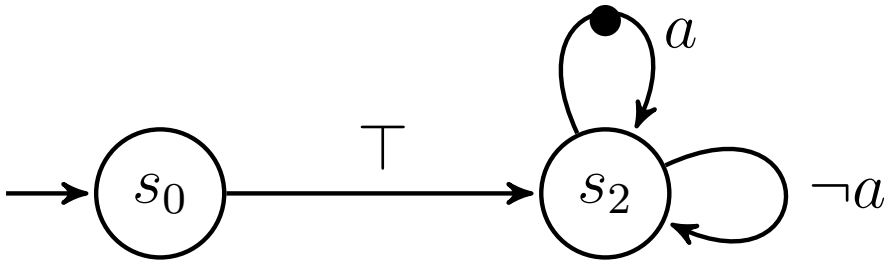


Size

Algo. Spot	Automata St.	Tr.	ND St.	ND	Product states	trans.	time
TGBA							
No Sim	702	1901	233	70	140167	6371627	3.46
RDS	676	1761	120	51	134677	4917433	3.45
RRS	668	1688	185	67	133369	5352279	3.43
RIS	659	1638	102	51	131507	4668891	3.73
BA							
No Sim	768	2156	293	70	153364	7345375	3.60
RDS	742	2015	142	51	147727	5405417	3.57
RRS	732	1918	224	67	146159	6145500	3.50
RIS	721	1854	122	51	143810	5118850	3.71

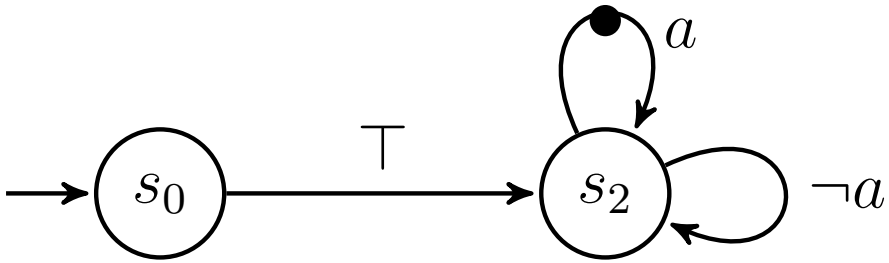
Don't Care (Direct) Simulation

- Acceptance conditions on transitions that are out of a SCC do not contribute to the language. We can modify them;



Don't Care (Direct) Simulation

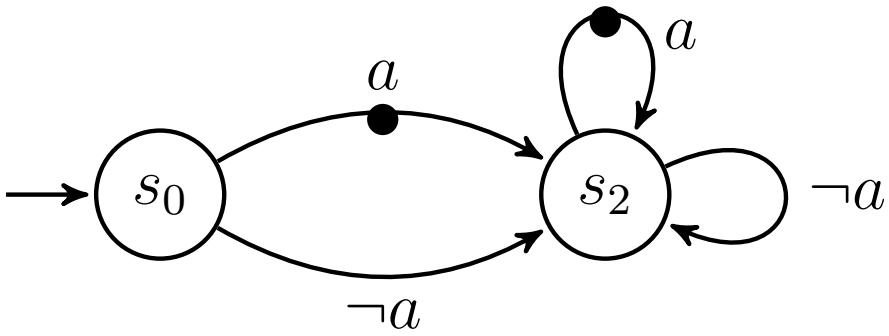
- Acceptance conditions on transitions that are out of a SCC do not contribute to the language. We can modify them;



- Adding or removing an acceptance condition is not enough in that case. We need to split the transition;

Don't Care (Direct) Simulation

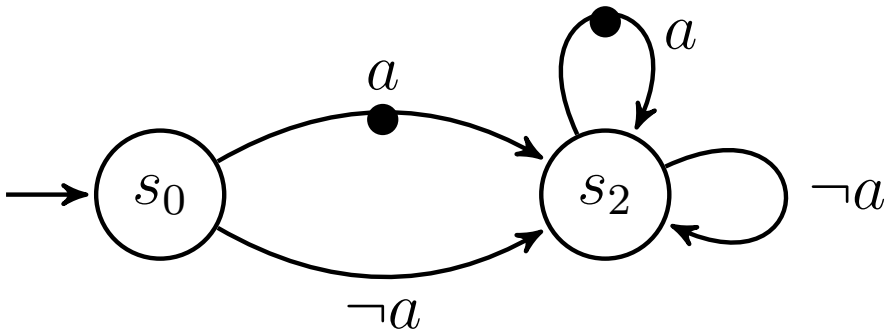
- Acceptance conditions on transitions that are out of a SCC do not contribute to the language. We can modify them;



- Adding or removing an acceptance condition is not enough in that case. We need to split the transition;

Don't Care (Direct) Simulation

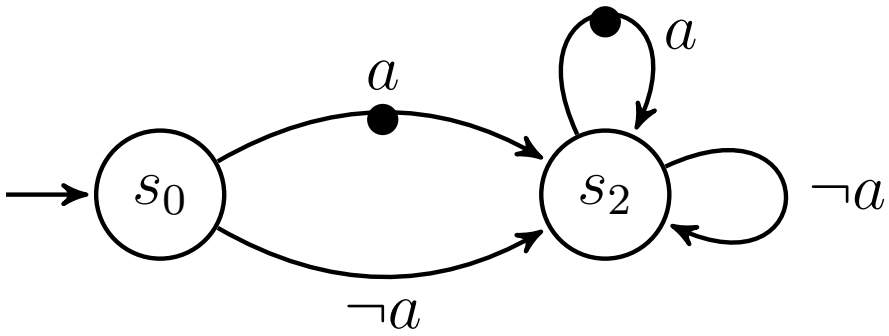
- Acceptance conditions on transitions that are out of a SCC do not contribute to the language. We can modify them;



- Adding or removing an acceptance condition is not enough in that case. We need to split the transition;
- We can detect which states could imply another, and find what must change;

Don't Care (Direct) Simulation

- Acceptance conditions on transitions that are out of a SCC do not contribute to the language. We can modify them;



- Adding or removing an acceptance condition is not enough in that case. We need to split the transition;
- We can detect which states could imply another, and find what must change;
- Interaction between different changes is complicated. We brute force to keep the best result.

Size

Algo. Spot	Automata St.	Tr.	ND St.	ND	Product states	trans.	time
TGBA							
No Sim	702	1901	233	70	140167	6371627	3.46
RDS	676	1761	120	51	134677	4917433	3.45
RRS	668	1688	185	67	133369	5352279	3.43
RIS	659	1638	102	51	131507	4668891	3.73
RDCS	**676**	**1750**	**120**	**51**	**134677**	**4917433**	**5.18**
RDCIS	**659**	**1614**	**93**	**51**	**131507**	**4644451**	**5.52**
BA							
No Sim	768	2156	293	70	153364	7345375	3.60
RDS	742	2015	142	51	147727	5405417	3.57
RRS	732	1918	224	67	146159	6145500	3.50
RIS	721	1854	122	51	143810	5118850	3.71
RDCS	**766**	**2080**	**151**	**51**	**152463**	**5551454**	**5.19**
RDCIS	**744**	**1899**	**121**	**51**	**148334**	**5230622**	**5.58**

Conclusion

The work done

- The direct simulation was already implemented;
- RDS, RRS, RIS are integrated in SPOT 1.0 and in the web interface (`http://spot.lip6.fr/ltl2tgba.html`);
- RDCS, RDCIS are ready to be integrated.

The future work

- Don't care simulation is not perfect yet;
- Experiment the simulations on the BA (after *degeneralize*);
- Work on Delayed Simulation.

Question?

Bibliography I

Matthew B. Dwyer, George S. Avrunin, and James C. Corbett.

Property specification patterns for finite-state verification.

In Mark Ardis, editor, *Proceedings of the 2nd Workshop on Formal Methods in Software Practice (FMSP'98)*, pages 7–15, New York, March 1998. ACM Press.

Kousha Etessami and Gerard J. Holzmann.

Optimizing Büchi automata.

In C. Palamidessi, editor, *Proceedings of the 11th International Conference on Concurrency Theory (Concur'00)*, volume 1877 of *Lecture Notes in Computer Science*, pages 153–167, Pennsylvania, USA, 2000. Springer-Verlag.

Bibliography II

Fabio Somenzi and Roderick Bloem.

Efficient Büchi automata for LTL formulæ.

In *Proceedings of the 12th International Conference on Computer Aided Verification (CAV'00)*, volume 1855 of *Lecture Notes in Computer Science*, pages 247–263, Chicago, Illinois, USA, 2000. Springer-Verlag.