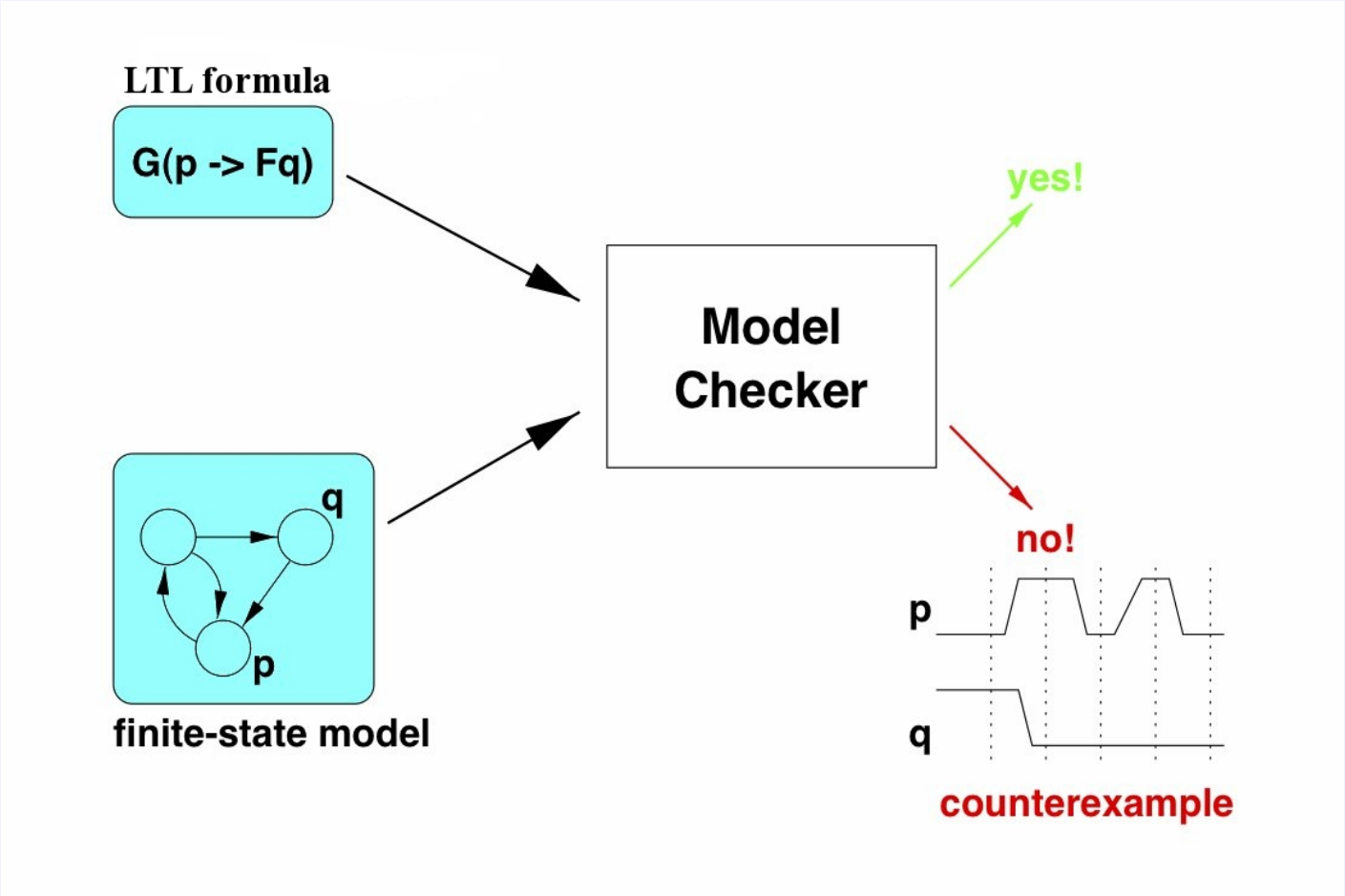
LTL model checking of stuttering-insensitive properties

Ala Eddine BEN SALEM

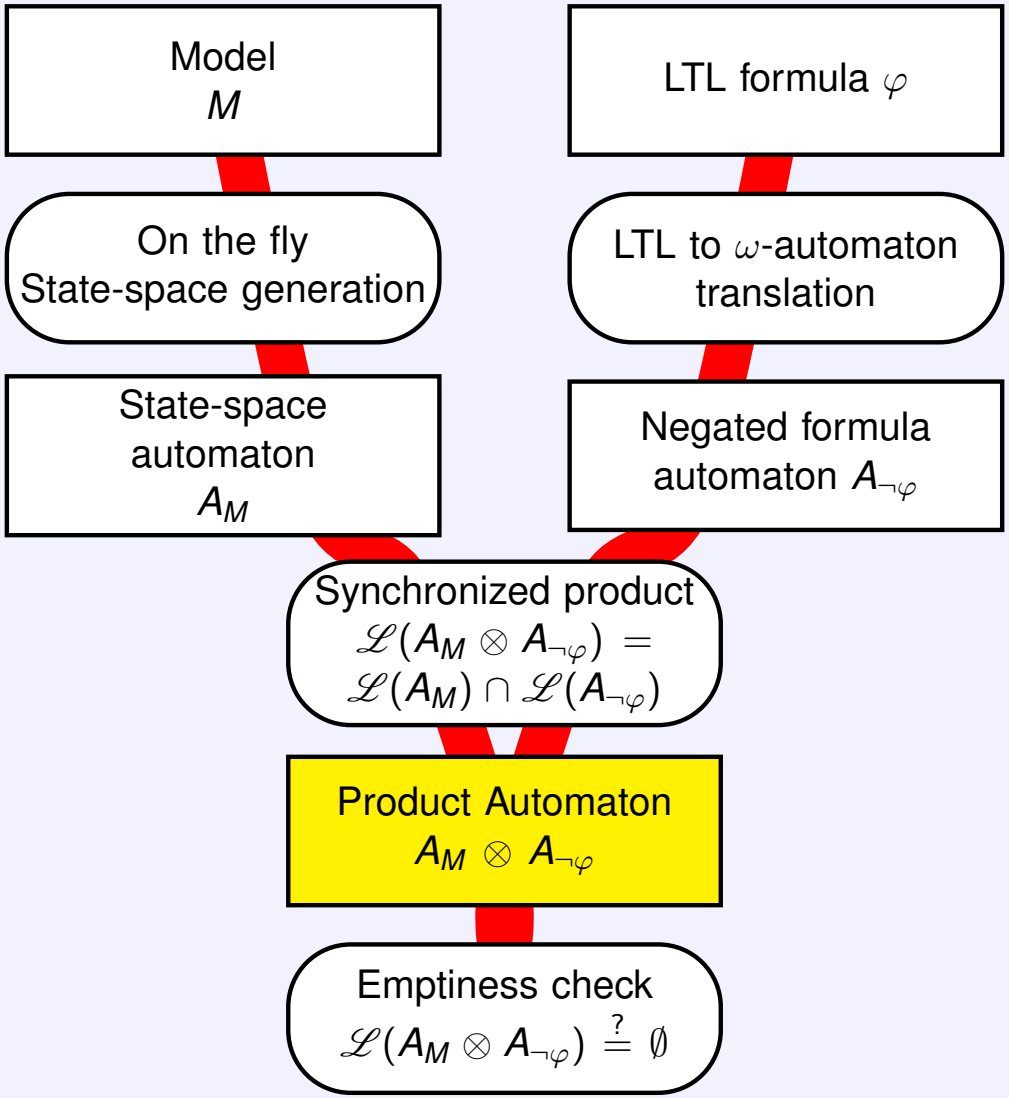
LRDE/LIP6

04 July 2012

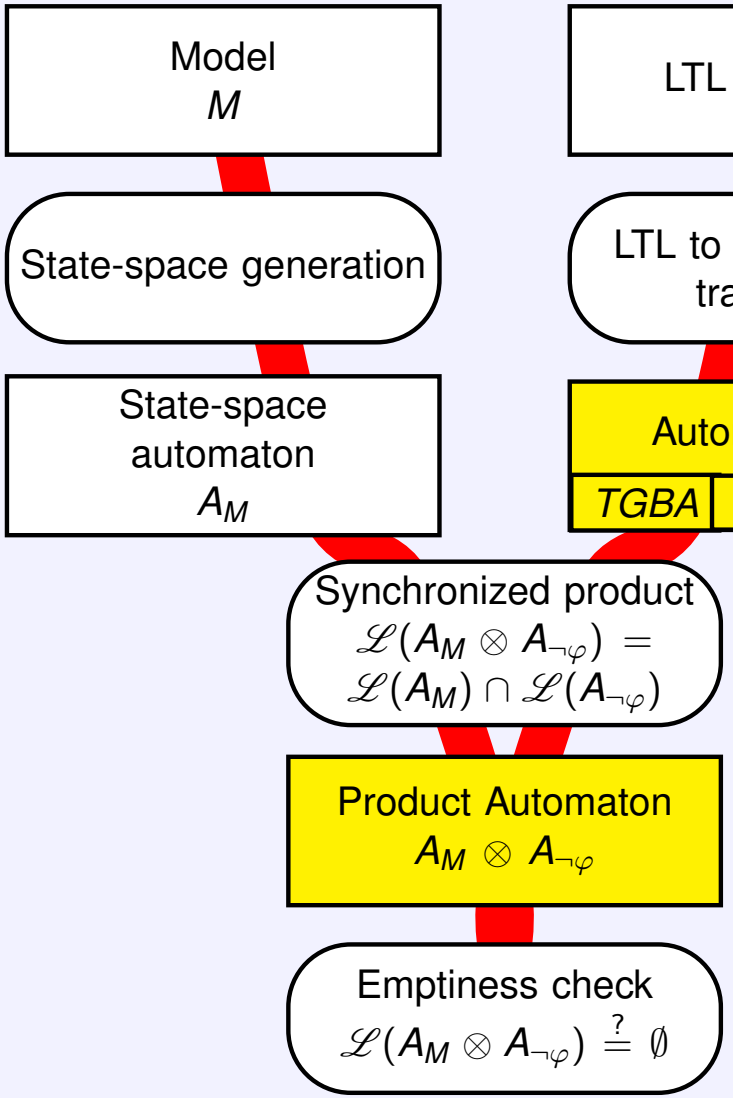
Model Checking



Automata-Theoretic Approach to Model Checking



Automata-Theoretic Approach to Model Checking

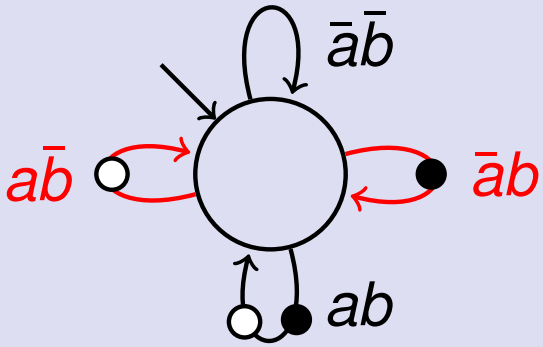


There are different types of Automata:

- **TGBA**: Transition-based Generalized Büchi Automata
- **BA**: Büchi Automata
- **TA**: Testing Automata (stuttering-insensitive)

Approach 1: TGBA

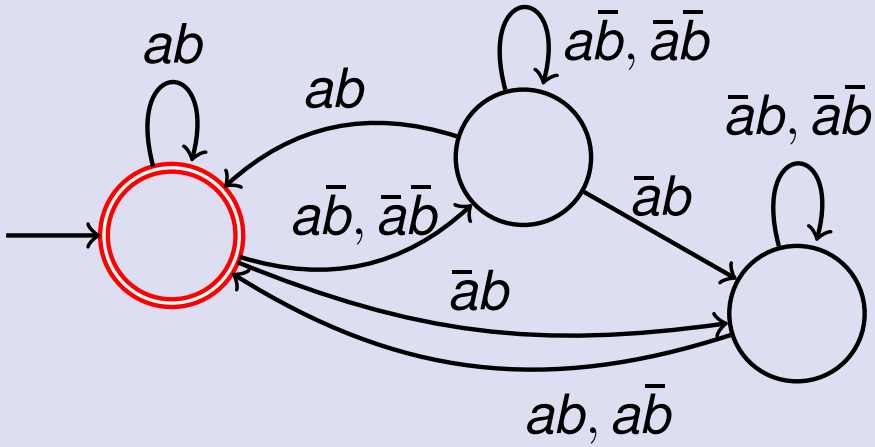
TGBA for the LTL property $\varphi = \mathrm{G}\,\mathrm{F}\,a \wedge \mathrm{G}\,\mathrm{F}\,b$ (Weak-fairness)



- Let AP = the set of *atomic proposition*.
- A TGBA over the alphabet $K = 2^{AP}$ is a tuple $\langle S, I, R, F \rangle$:
  - S is finite set of states,
  - $I \subseteq S$ is the set of initial states,
  - F is a finite set of acceptance conditions,
  - $R \subseteq S \times 2^K \times 2^F \times S$ is the transition relation.
- An infinite run of a TGBA is accepting if it visits each accepting condition from F ($\bullet$, $\circ$,...) infinitely often.

Approach 2: BA

BA recognizing LTL property $\varphi = \mathrm{G}\,\mathrm{F}\,a \wedge \mathrm{G}\,\mathrm{F}\,b$



Obtained from a TGBA by degeneralization

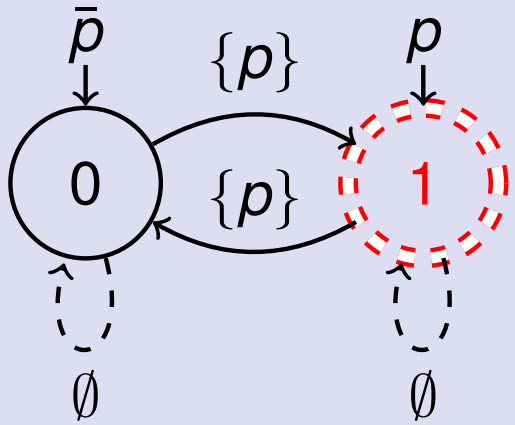
- Has only one acceptance condition that is state-based.
- A BA over the alphabet $K = 2^{AP}$ is a tuple $\langle S, I, R, F \rangle$:
  - $F \subseteq S$ **is a finite set of acceptance states**
  - $R \subseteq S \times 2^K \times S$ is the transition relation
- An infinite run of a BA is accepting if it visits at least one acceptance state infinitely often.

Approach 3: TA (only stuttering-insensitive)

TA recognizing LTL property F G p

Model Execution = $\bar{p}\ \bar{p}\ p\ p\ \bar{p}\ p\ p\ p\ p \ldots$

TA Run = 0 0 1 1 0 1 1 1 1 ...



A TA over the alphabet $K = 2^{AP}$ is a tuple $\langle S, I, U, R, F, G \rangle$:

- $R \subseteq S \times \mathbf{K} \times S$: each transition (s, k, d) is labeled by the set of **atomic propositions that change** between s and d,
- $F \subseteq S$ is a set of Büchi acceptance states,
- $G \subseteq S$ **is a set of livelock acceptance states**.

A second way to accept an infinite run: reaches a livelock acceptance state and from that point only stuttering.

Preliminary work: Experimental comparison of the three approaches

Hypothesis: LTL\ X formulas (*stuttering-insensitive*)

Experimental evaluation comparing the three approaches: TGBA, BA and TA.

Results [Ben Salem 2011]:

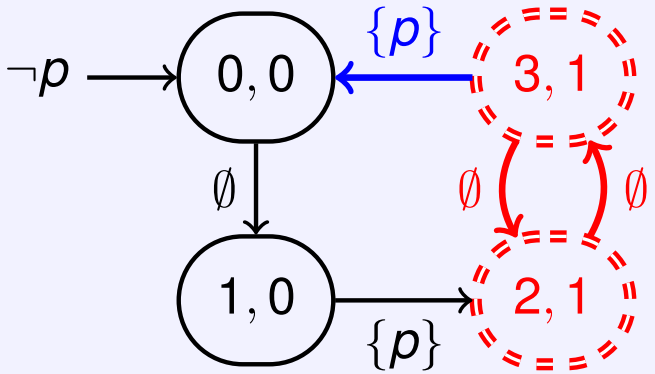
- Verified properties (complete exploration of the product):
  - TA requires **two-pass emptiness check**
  - It is therefore better to use the **TGBA** approach .
- Violated properties (partial exploration of the product):
  - **TA** approach is the most efficient to detect counterexample
- TGBA is more efficient than BA in all cases

Contributions

1. Enhancing TA emptiness check to avoids a second pass when it is possible
2. *Single-pass Testing Automata* (STA):
   - a transformation of TA that never requires a second pass
   - add an artificial livelock state (the only one)
3. *Transition-based Generalized Testing Automata* (TGTA):
   - new automaton that combines benefits from TA and TGBA
   - no two-pass emptiness check like TA
   - no artificial state like STA

Why does TA emptiness check require two passes ?

- Two kinds of accepting SCC: Büchi or livelock (accepted if composed **only by stuttering-transitions** $\emptyset$)
- first pass may miss to detect livelock-accepting SCCs (depending on order to explore the transitions of $(3, 1)$)



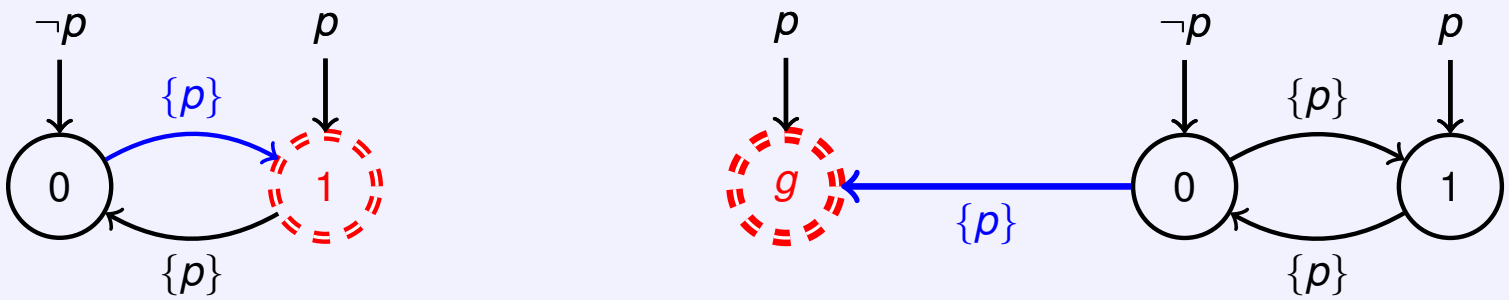
Product between a model and a TA of (F G p). The red SCC is livelock-accepting.

- Problem: mixing of non-stuttering and stuttering transitions in the same SCC

Single-pass Testing Automata (STA)

We transform a TA into a STA by:

- adding a unique livelock-acceptance state g and
- adding a transition (s, k, g) for any transition (s, k, s') that goes into a livelock-acceptance state s' in TA

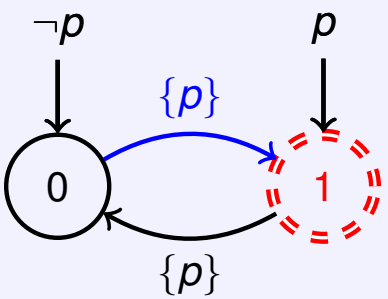


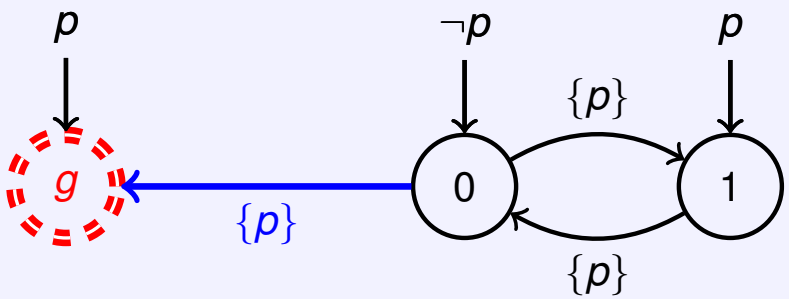
Transfomation of TA (F G p) into STA

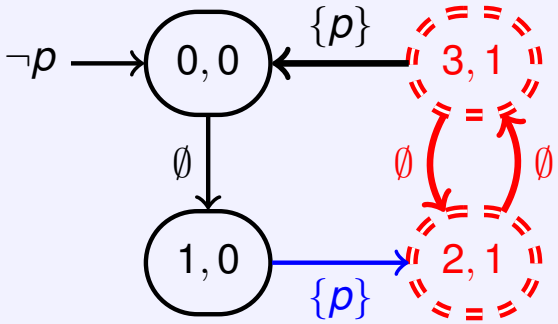
Single-pass Testing Automata (STA)

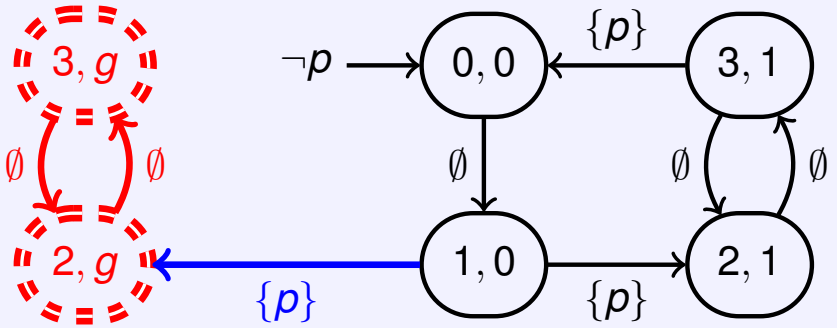
We transform a TA into a STA by:

- adding a unique livelock-acceptance state g and
- adding a transition (s, k, g) for any transition (s, k, s') that goes into a livelock-acceptance state s' in TA







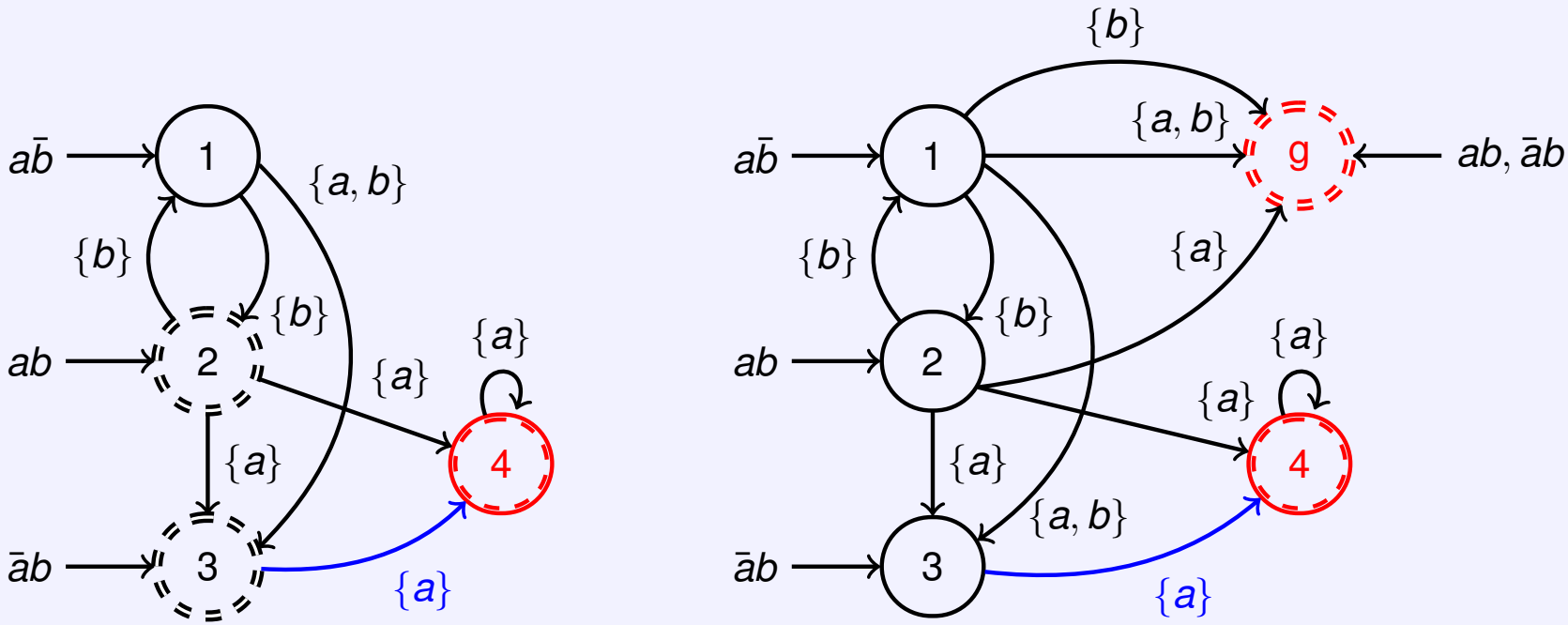


Impact of STA on the product: single-pass emptiness check

STA optimization

During the TA to STA transformation:

- don't add transition (s, k, g) for transition (s, k, s') where s' is both livelock and Büchi accepting,
- because in the product, any SCC containing s' is accepting



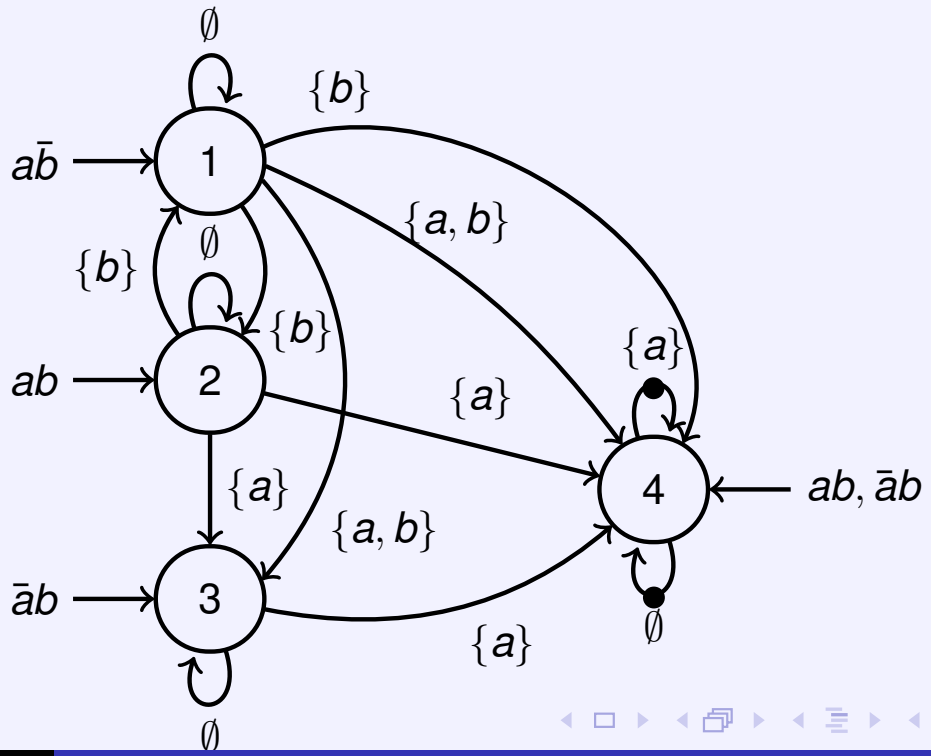
Transformation of TA (recognizing $a\,\mathrm{U}\,\mathrm{G}\,b$) into optimized STA. The state 4 is both livelock and Büchi accepting

TGTA a new kind of automaton

TGTA combines ideas from TGBA and TA:

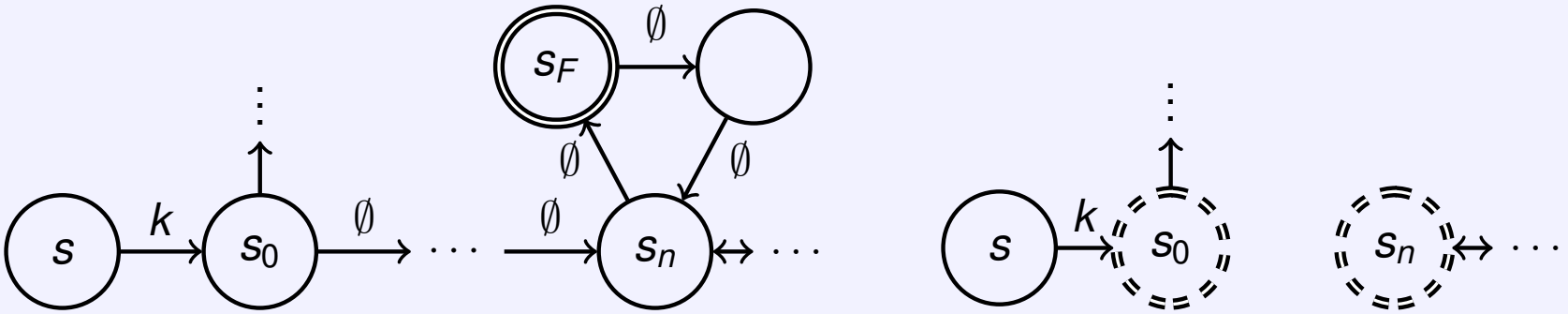
- From TGBA:
  - transition-based generalized acceptance conditions,
  - and a one-pass emptiness-check (the same algorithm)
- From TA:
  - reduction of stuttering-transitions
  - without adding livelock-acceptance (because two passes)

TGTA of $(a \,\mathrm{U}\, \mathrm{G}\, b)$:



Reduction of stuttering-transitions in TGTA

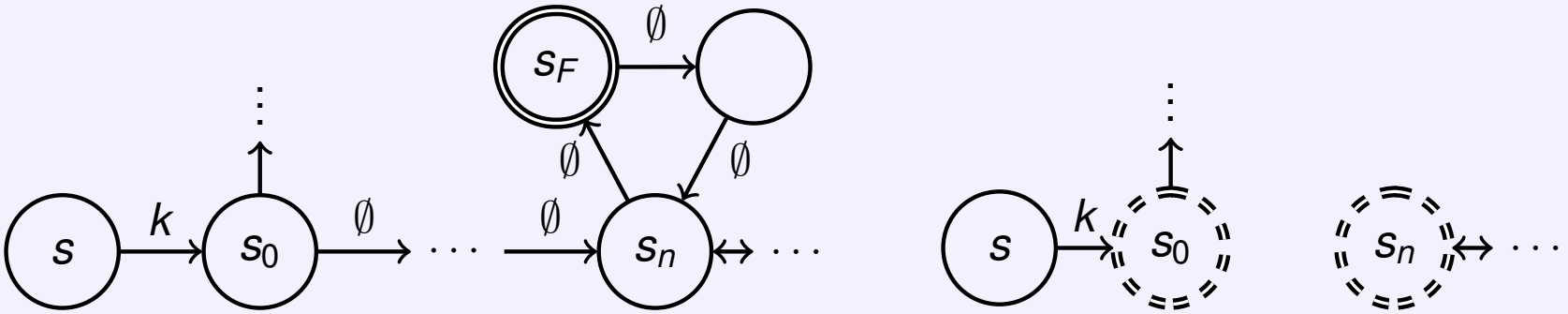
TGTA reduction does not add livelock-accepting states (like a TA reduction).



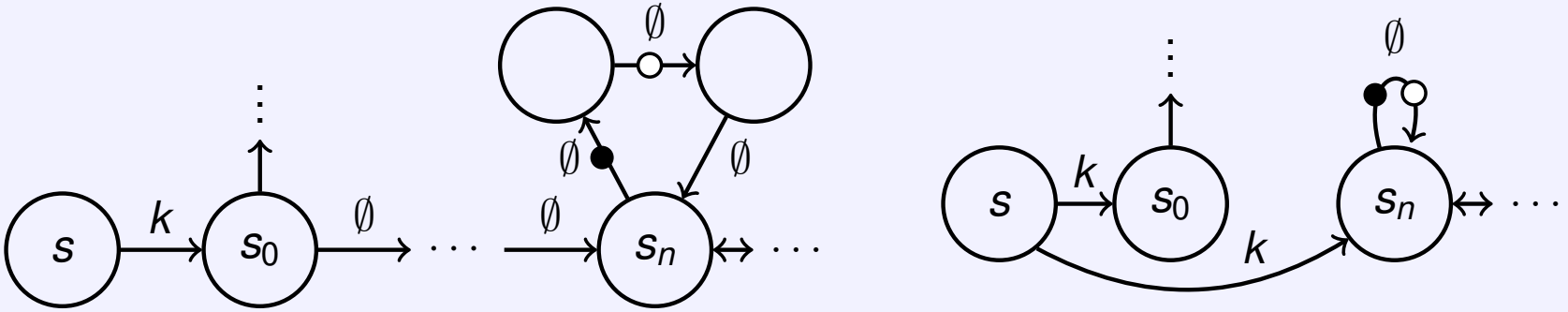
Reduction of stuttering-transitions in TA.

Reduction of stuttering-transitions in TGTA

TGTA reduction does not add livelock-accepting states (like a TA reduction).

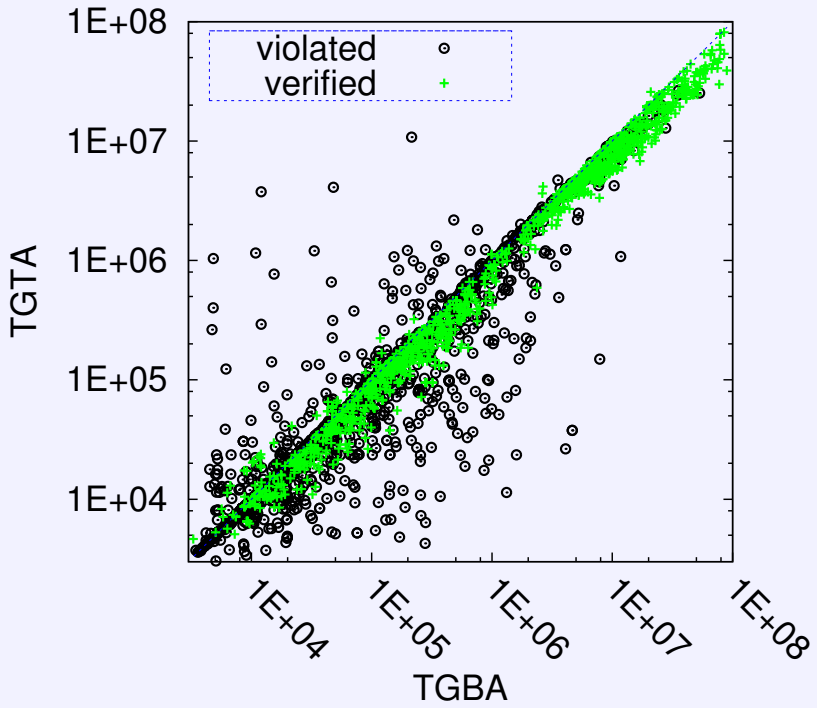


Reduction of stuttering-transitions in TA.



Reduction of stuttering-transitions in TGTA.

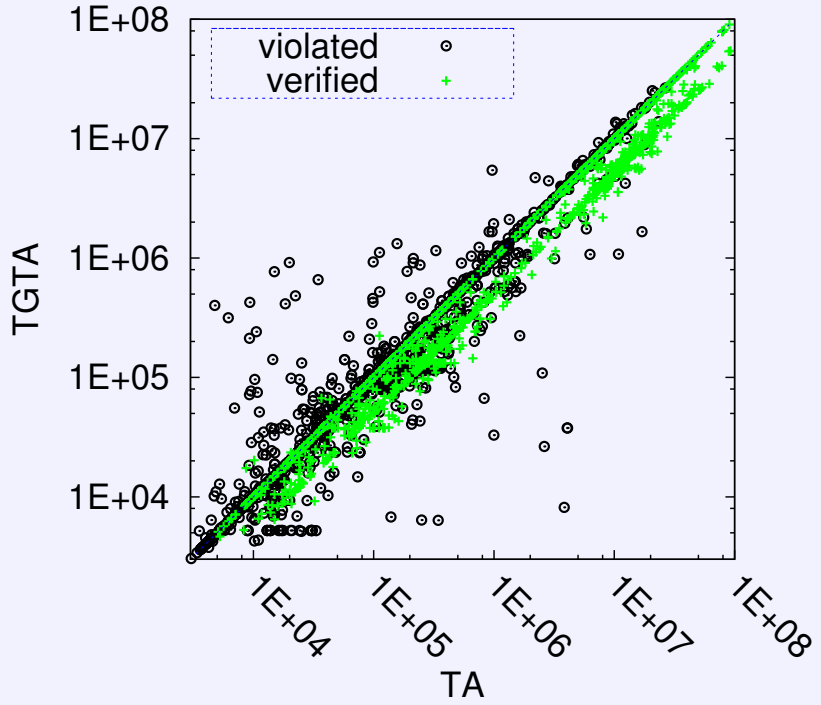
Experimental evaluation of TGTA against TGBA



Number of transitions explored by the emptiness check of TGTA against TGBA. Axes in logarithmic scale

- Verified properties (green crosses): TGTA is more efficient
- Violated properties (black circles): harder to interpret

Experimental evaluation of TGTA against TA



Number of transitions explored by the emptiness check of TGTA against TA. (Axes in logarithmic scale)

- Verified properties: TGTA more efficient, because TA requires two-pass
- Violated properties: same problem as for TGTA against TGBA

Conclusion

- We improved the model cheking of stuttering-insensitive properties
- with some contributions: enhancing TA emtiness check, proposing STA and TGTA
- TGTA is our most significant contribution [Ben Salem 2012]
- Our benchmarks show that TGTA outperform TA and TGBA

We plan additional work to:

- enable symbolic model checking with TGTA
- provide direct conversion of LTL to TGTA
- combine partial order reduction with TGTA

Publications

- [Ben Salem 2011] A.-E. B. Salem, A. Duret-Lutz, and F. Kordon. Generalized Büchi automata versus testing automata for model checking. In Proc. of SUMo'11 , vol. 726, pp. 65–79. CEUR.

- [Ben Salem 2012] A. E. Ben Salem, A. Duret-Lutz, and F. Kordon. Model Checking using Generalized Testing Automata. Transactions on Petri Nets and Other Models of Concurrency (ToPNoC) , ?:?, 2012. (à paraitre)

Questions