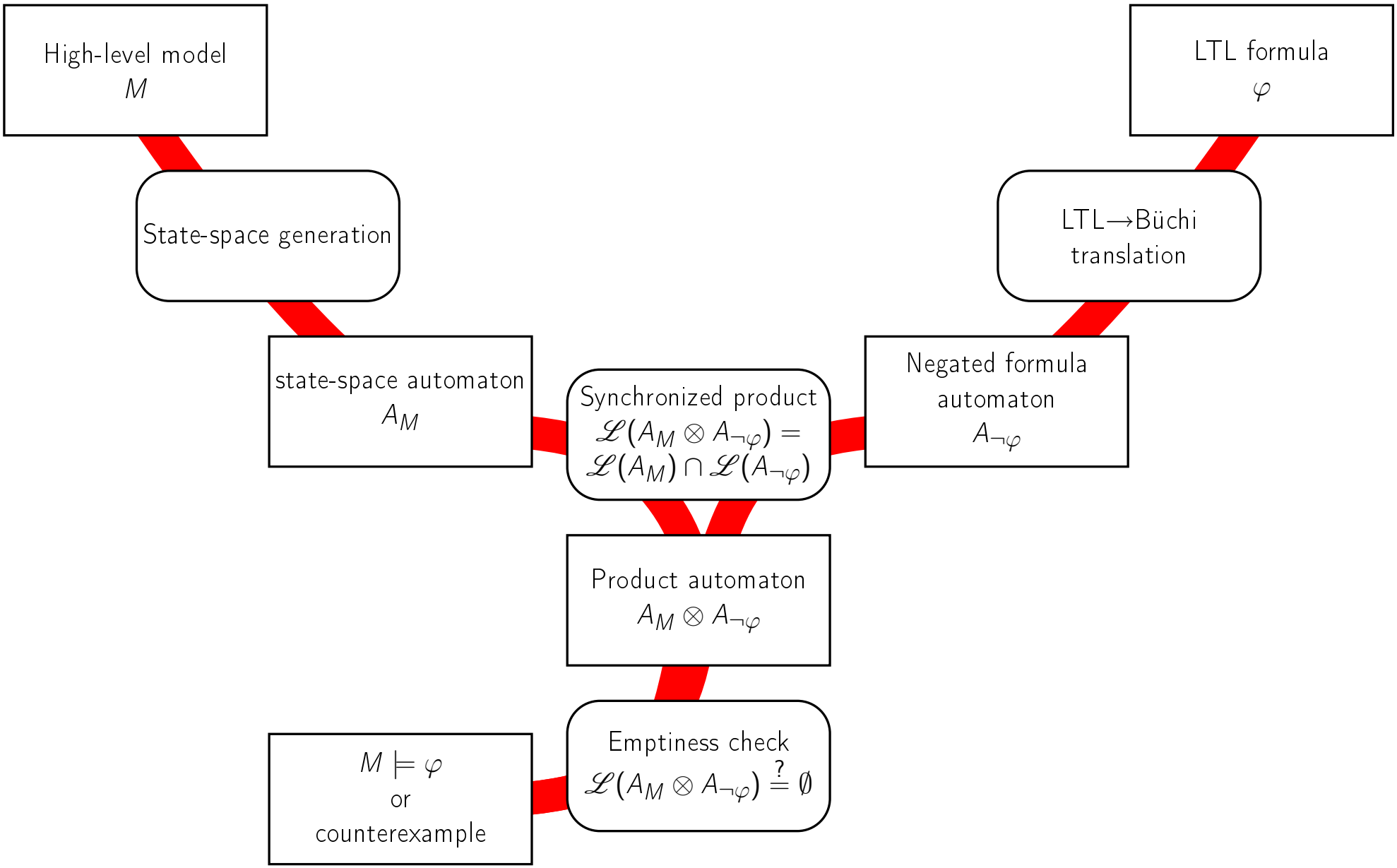
On-the-fly Emptiness Check of Transition-based Streett Automata

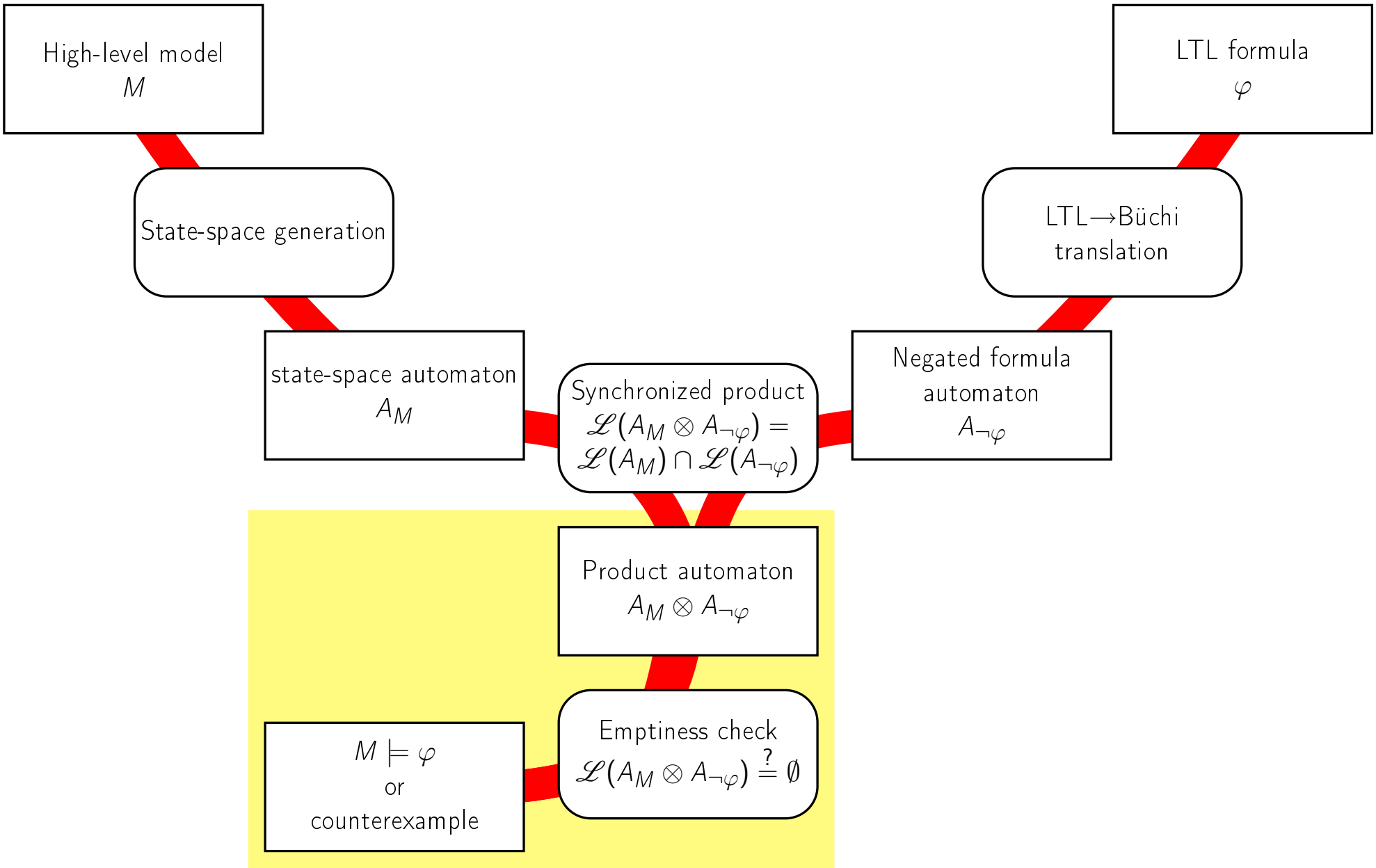
Alexandre Duret-Lutz, Denis Poitrenaud, Jean-Michel Couvreur

15th October 2009

Automata Theoretic Approach to Model Checking



Automata Theoretic Approach to Model Checking



Büchi Automata

They comes in various flavors:

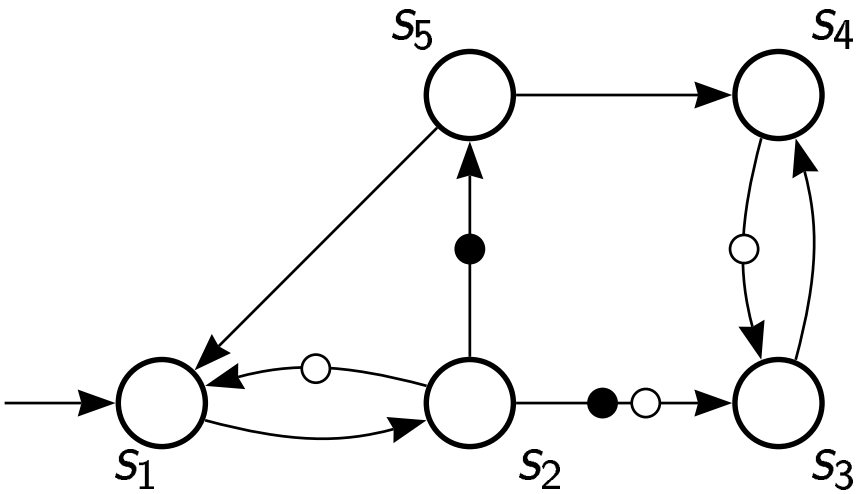
- Labels on states or transitions (ignored by emptiness check)
- Acceptance conditions on states or transitions
- Single acceptance set, or generalized acceptance conditions

Büchi Automata

They comes in various flavors:

- Labels on states or transitions (ignored by emptiness check)
- Acceptance conditions on states or transitions
- Single acceptance set, or generalized acceptance conditions

We focus on Transition-based Generalized Büchi Automata (TGBA).

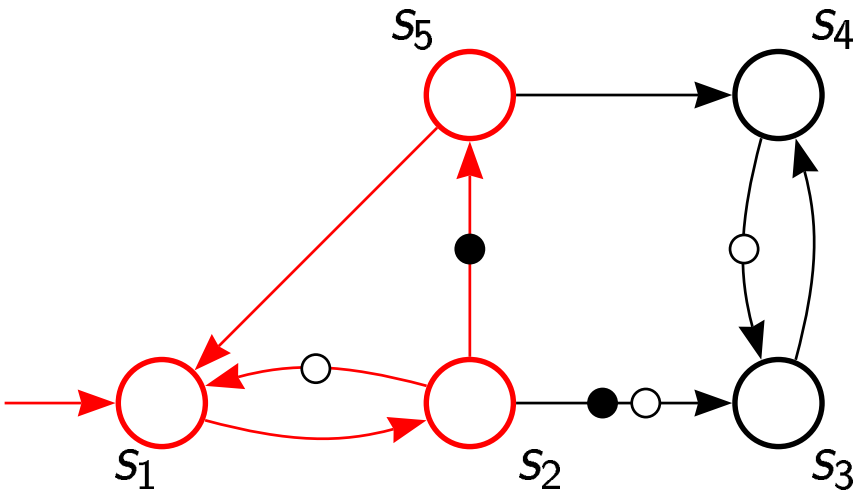


Büchi Automata

They comes in various flavors:

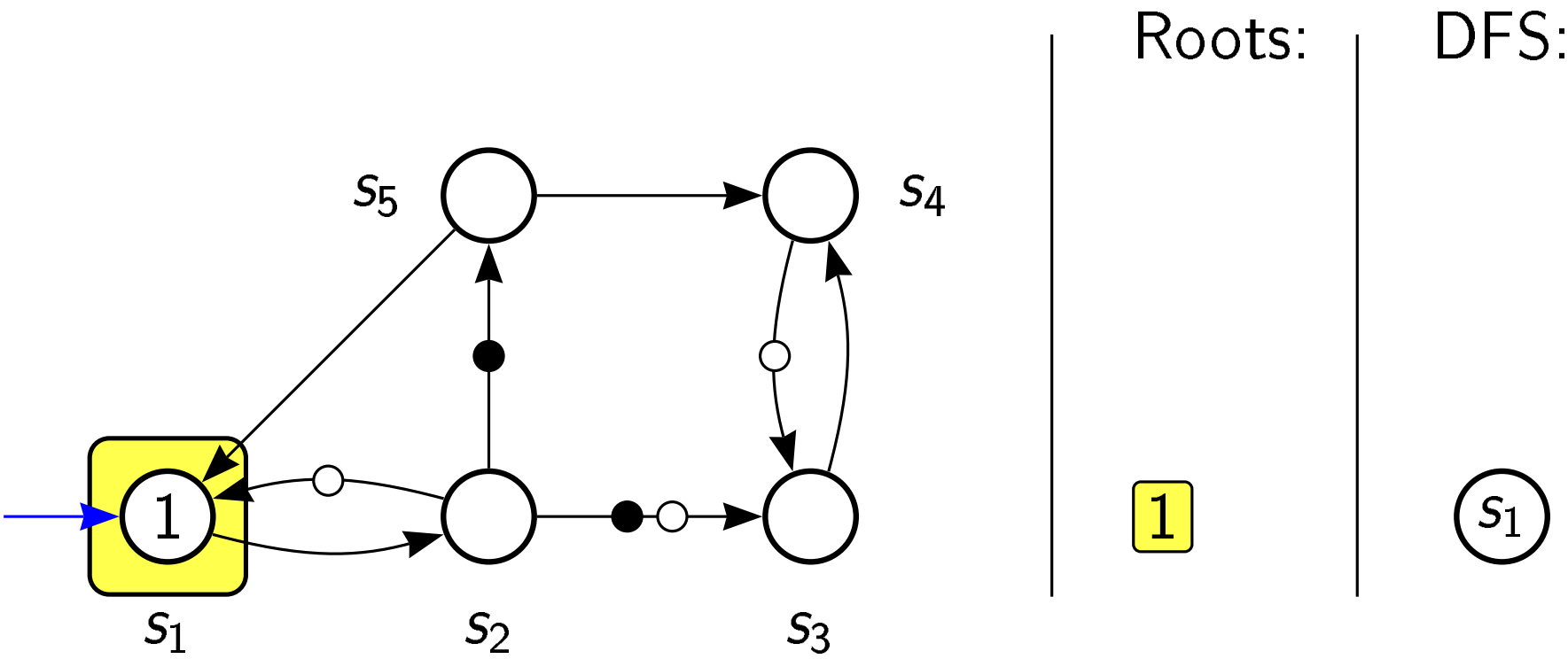
- Labels on states or transitions (ignored by emptiness check)
- Acceptance conditions on states or transitions
- Single acceptance set, or generalized acceptance conditions

We focus on Transition-based Generalized Büchi Automata (TGBA).



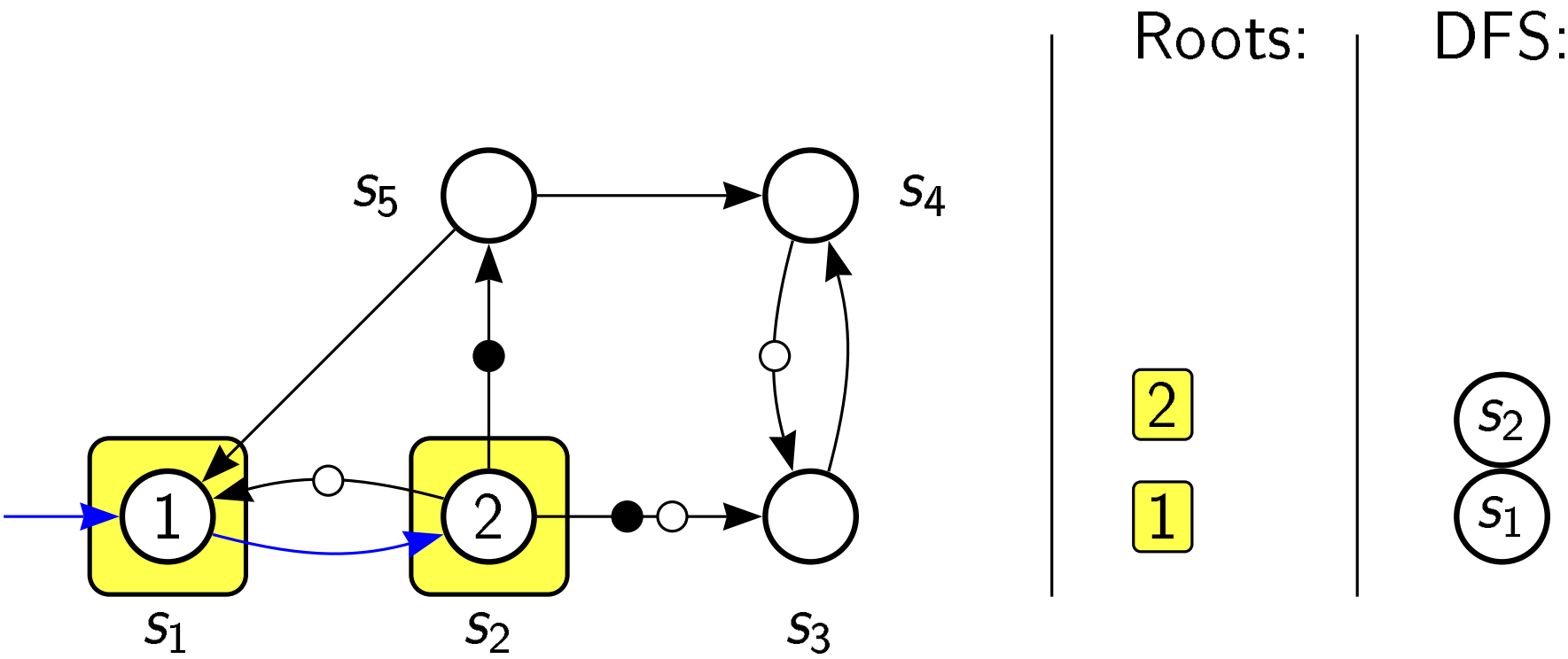
An infinite run of this automaton is accepting if it visits a transition from each accepting set infinitely often.

SCC-Based Emptiness Check for TGBA



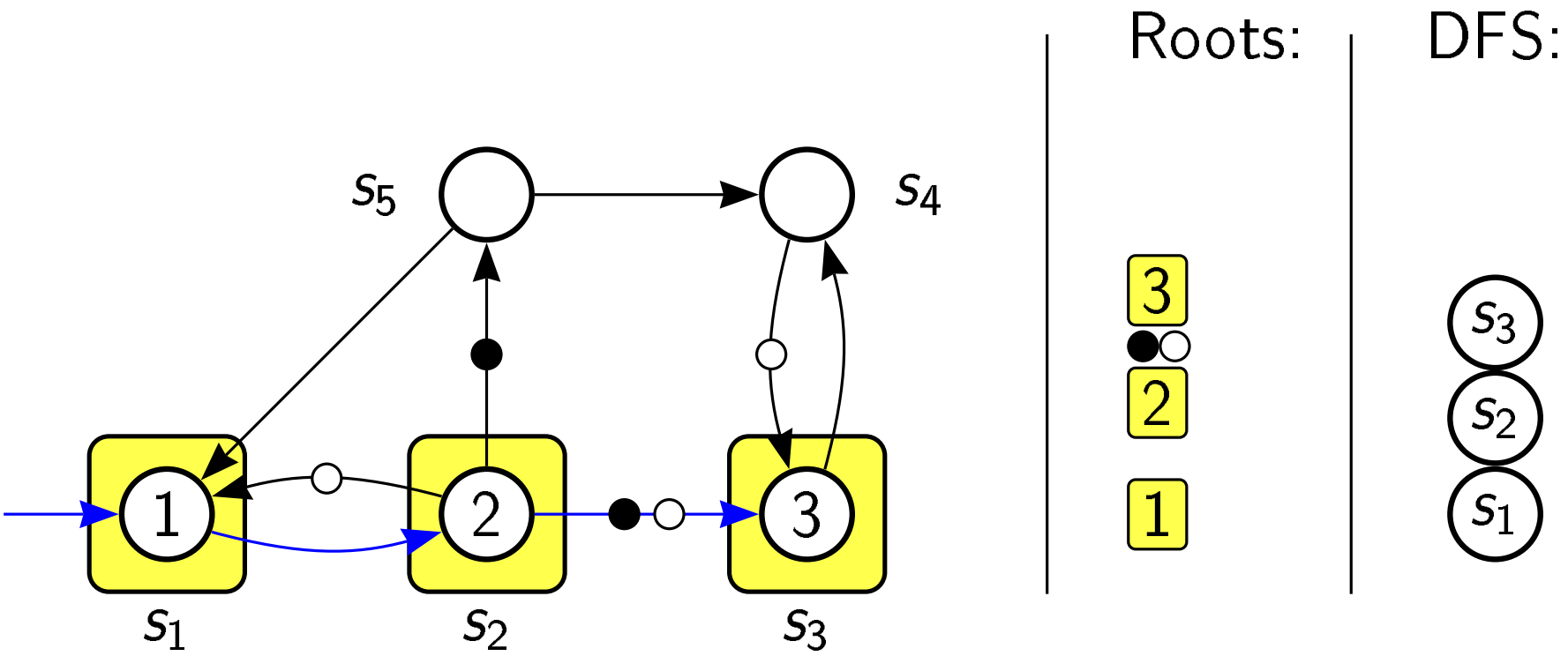
(From: *On-the-fly Verification of Temporal Logic*. Jean-Michel Couvreur. FM'99. LNCS 1708.)

SCC-Based Emptiness Check for TGBA



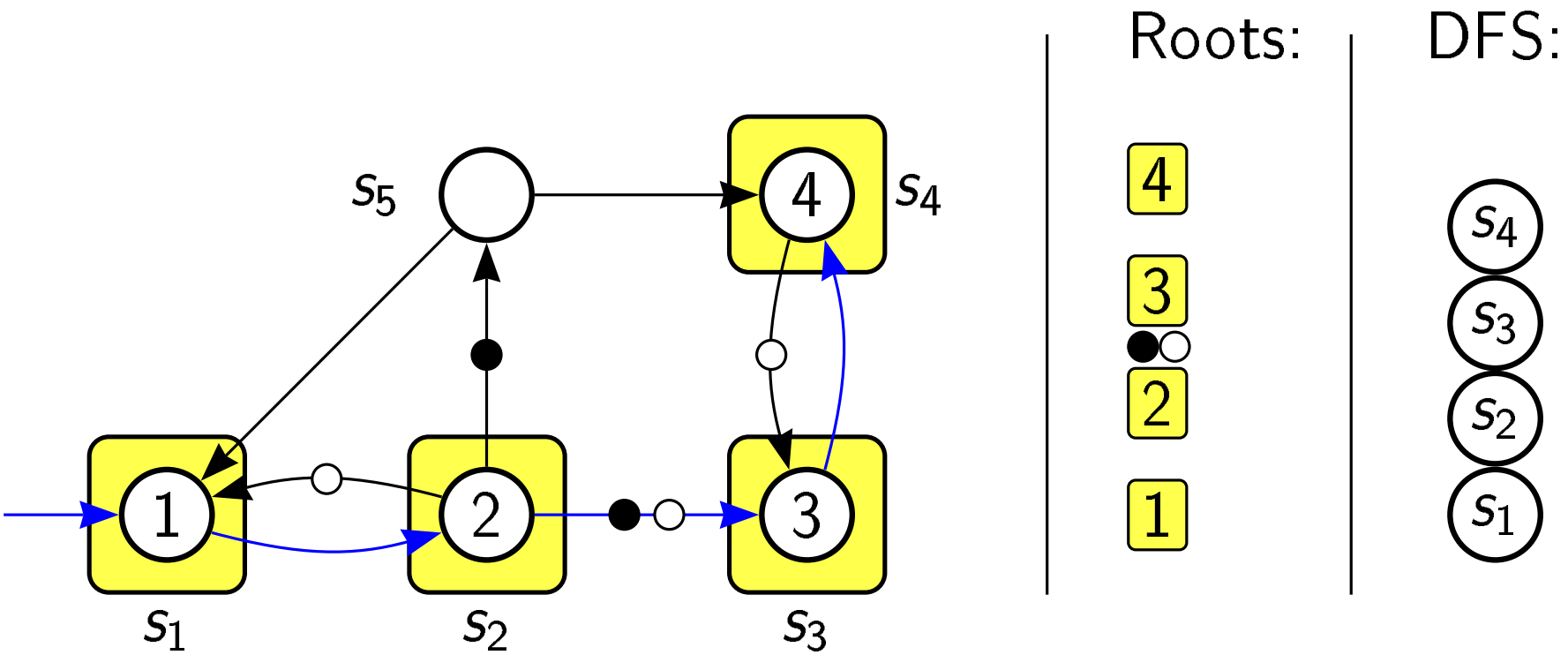
(From: *On-the-fly Verification of Temporal Logic*. Jean-Michel Couvreur. FM'99. LNCS 1708.)

SCC-Based Emptiness Check for TGBA



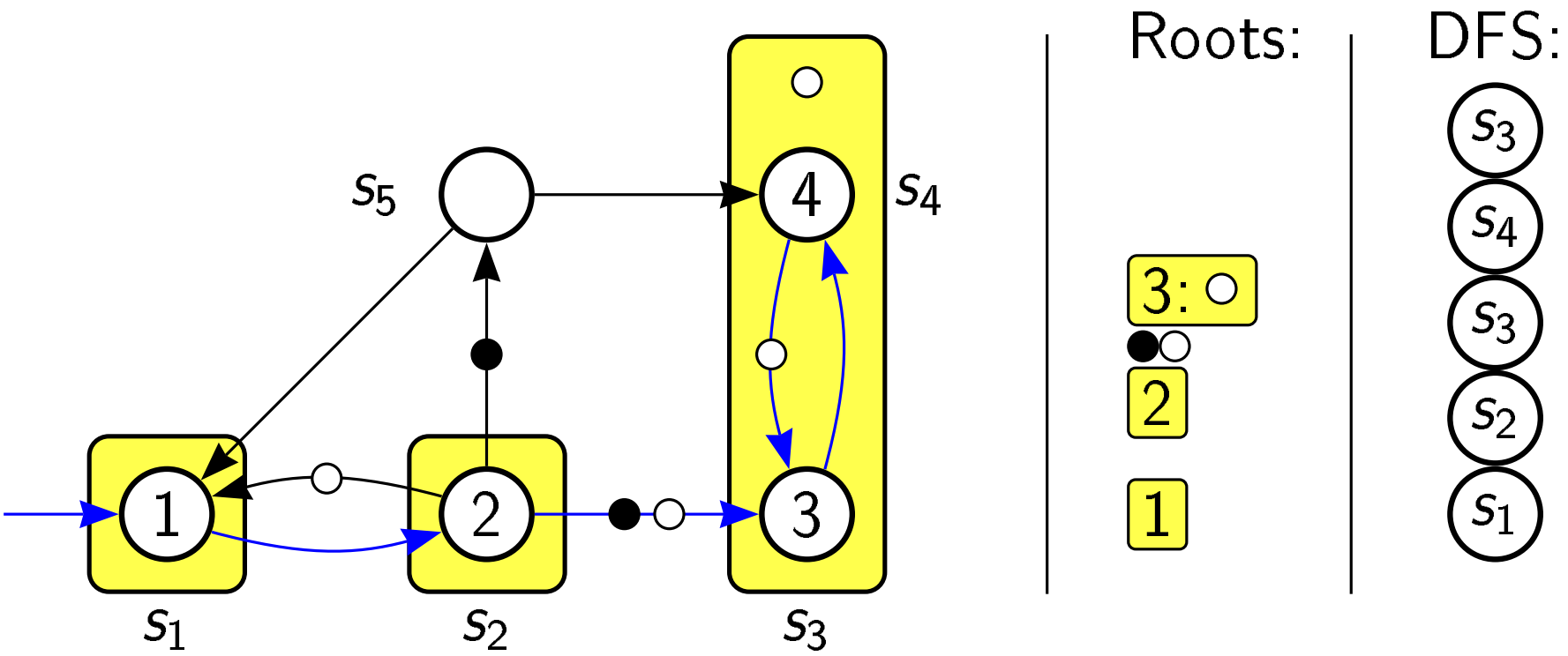
(From: *On-the-fly Verification of Temporal Logic*. Jean-Michel Couvreur. FM'99. LNCS 1708.)

SCC-Based Emptiness Check for TGBA



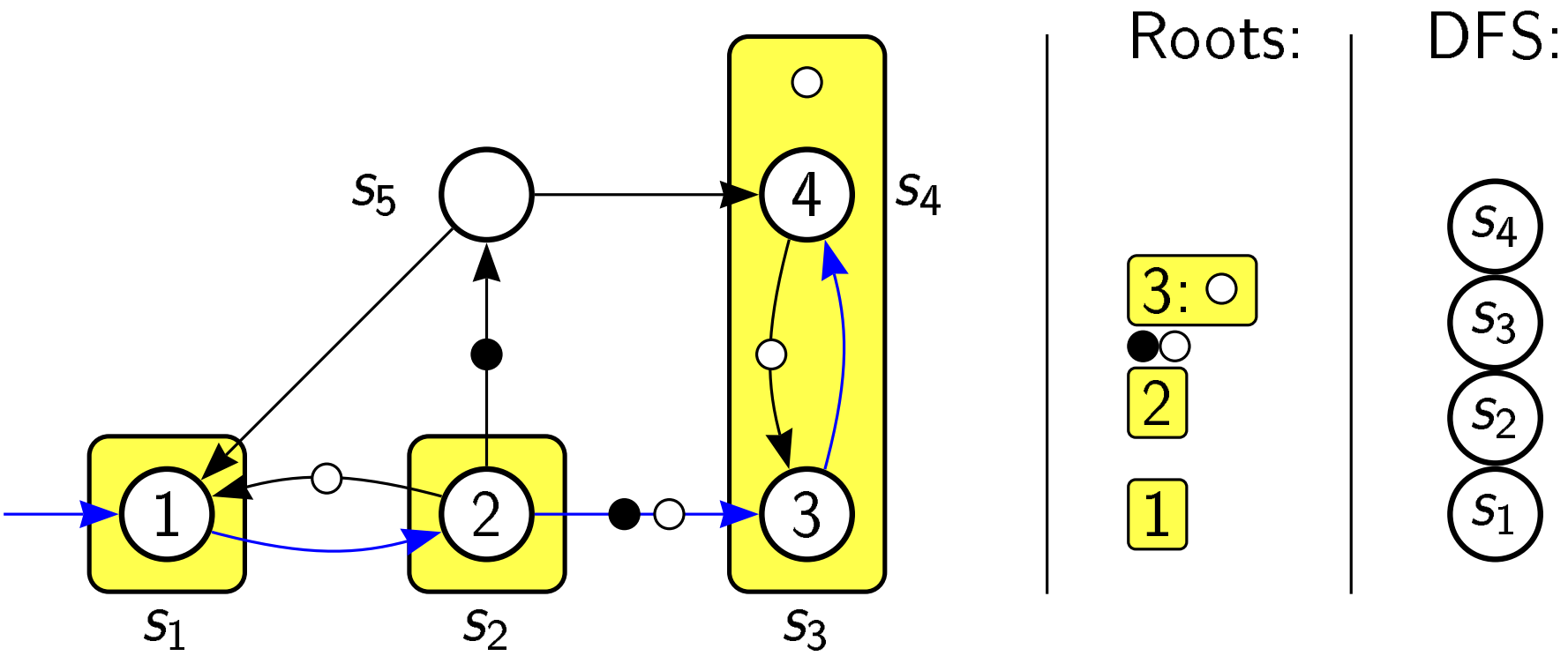
(From: *On-the-fly Verification of Temporal Logic*. Jean-Michel Couvreur. FM'99. LNCS 1708.)

SCC-Based Emptiness Check for TGBA



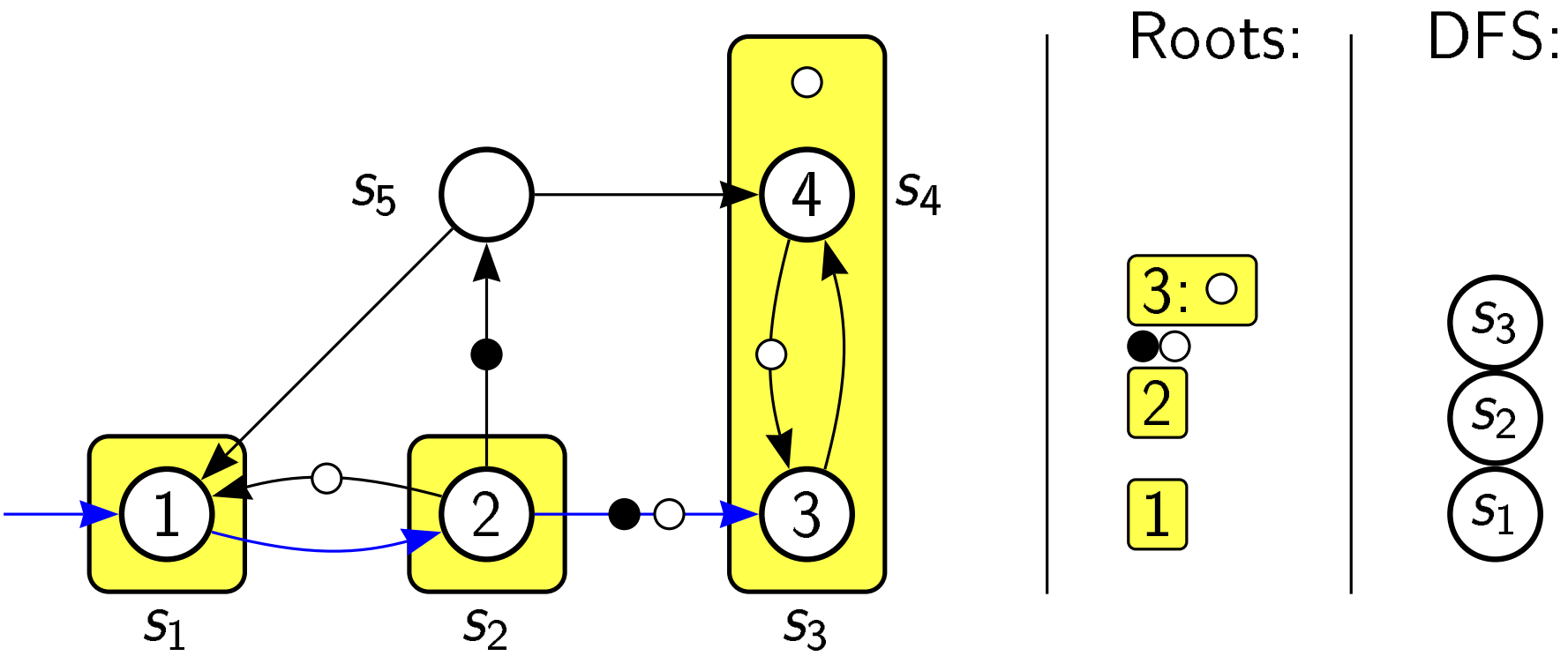
(From: *On-the-fly Verification of Temporal Logic*. Jean-Michel Couvreur. FM'99. LNCS 1708.)

SCC-Based Emptiness Check for TGBA



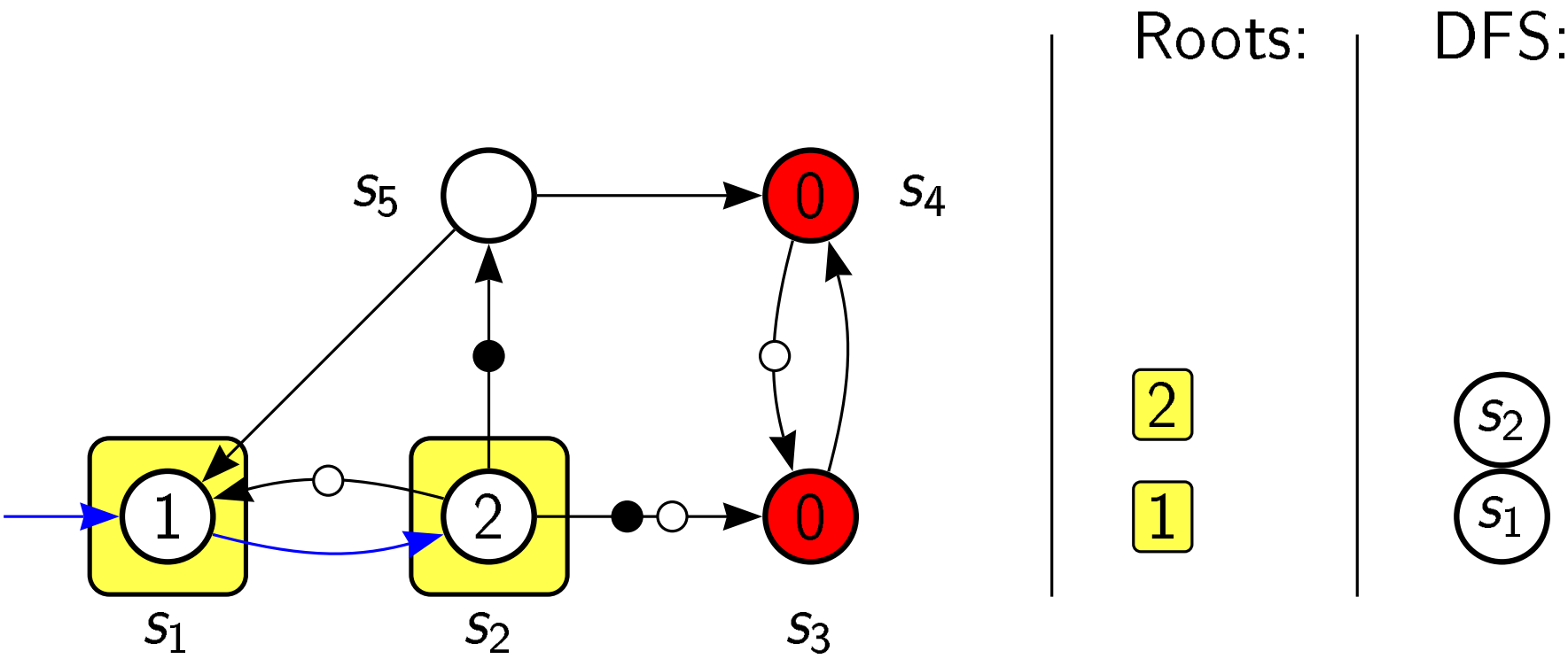
(From: *On-the-fly Verification of Temporal Logic*. Jean-Michel Couvreur. FM'99. LNCS 1708.)

SCC-Based Emptiness Check for TGBA



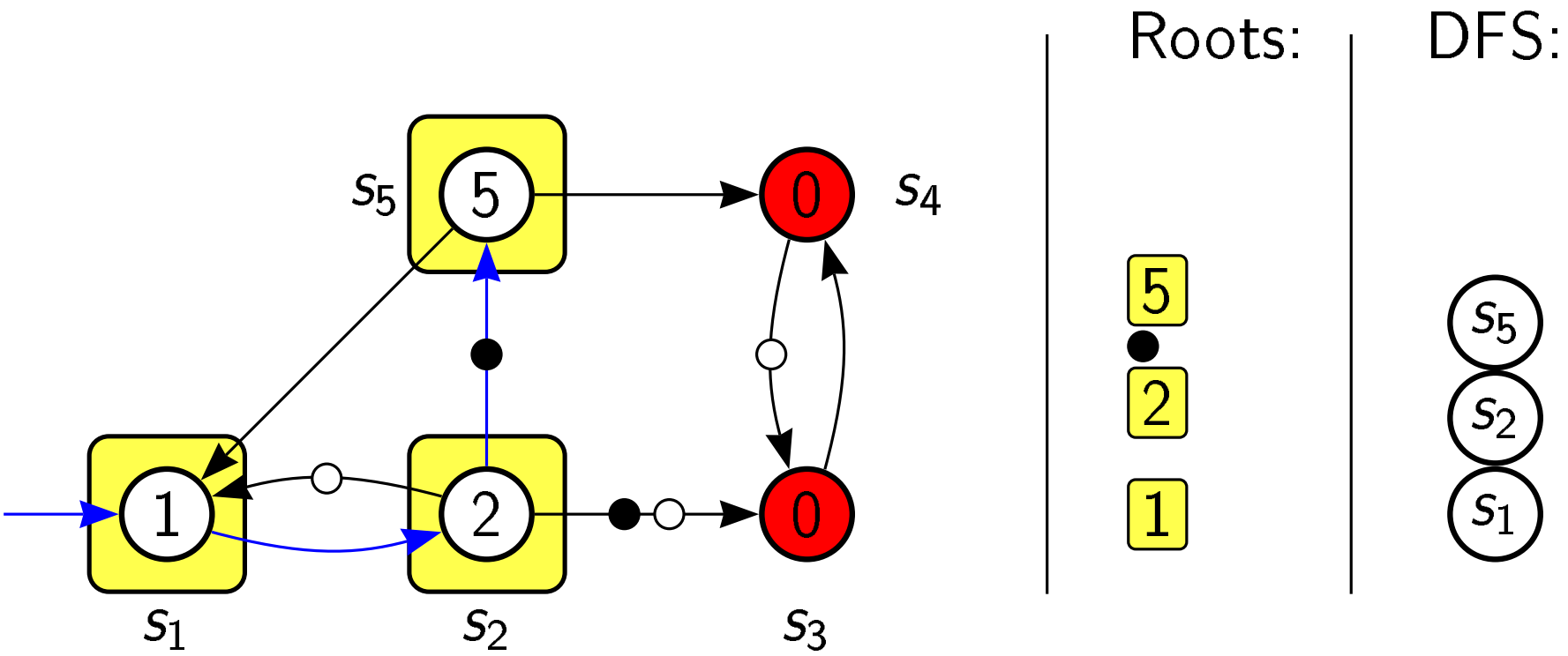
(From: *On-the-fly Verification of Temporal Logic*. Jean-Michel Couvreur. FM'99. LNCS 1708.)

SCC-Based Emptiness Check for TGBA



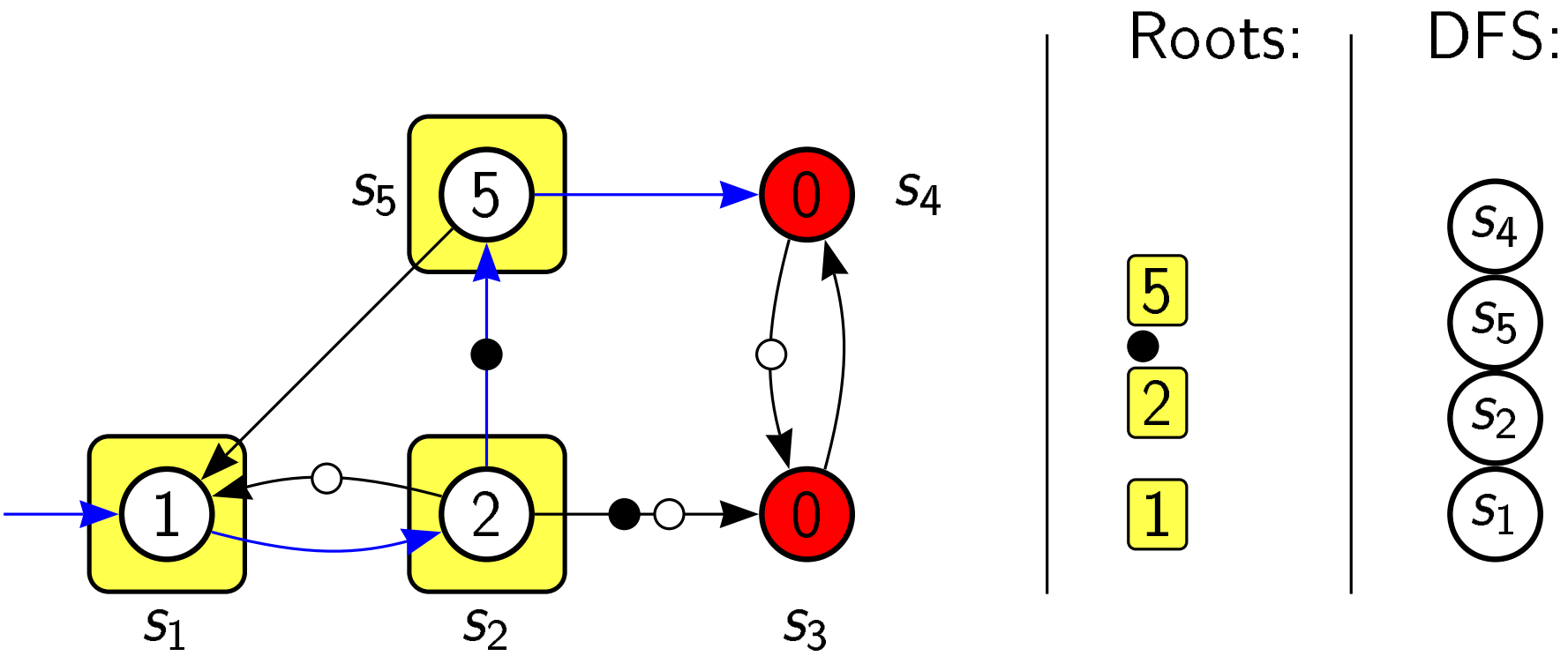
(From: *On-the-fly Verification of Temporal Logic*. Jean-Michel Couvreur. FM'99. LNCS 1708.)

SCC-Based Emptiness Check for TGBA



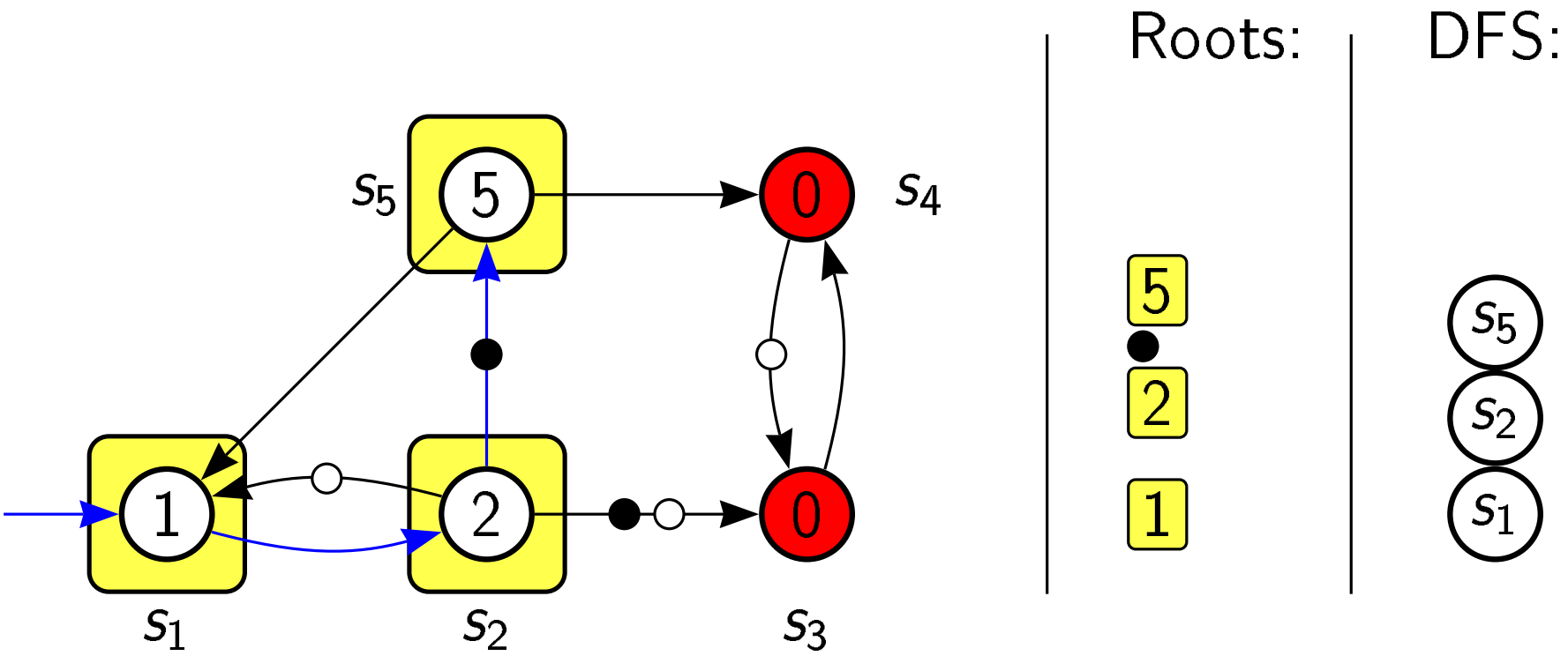
(From: *On-the-fly Verification of Temporal Logic*. Jean-Michel Couvreur. FM'99. LNCS 1708.)

SCC-Based Emptiness Check for TGBA



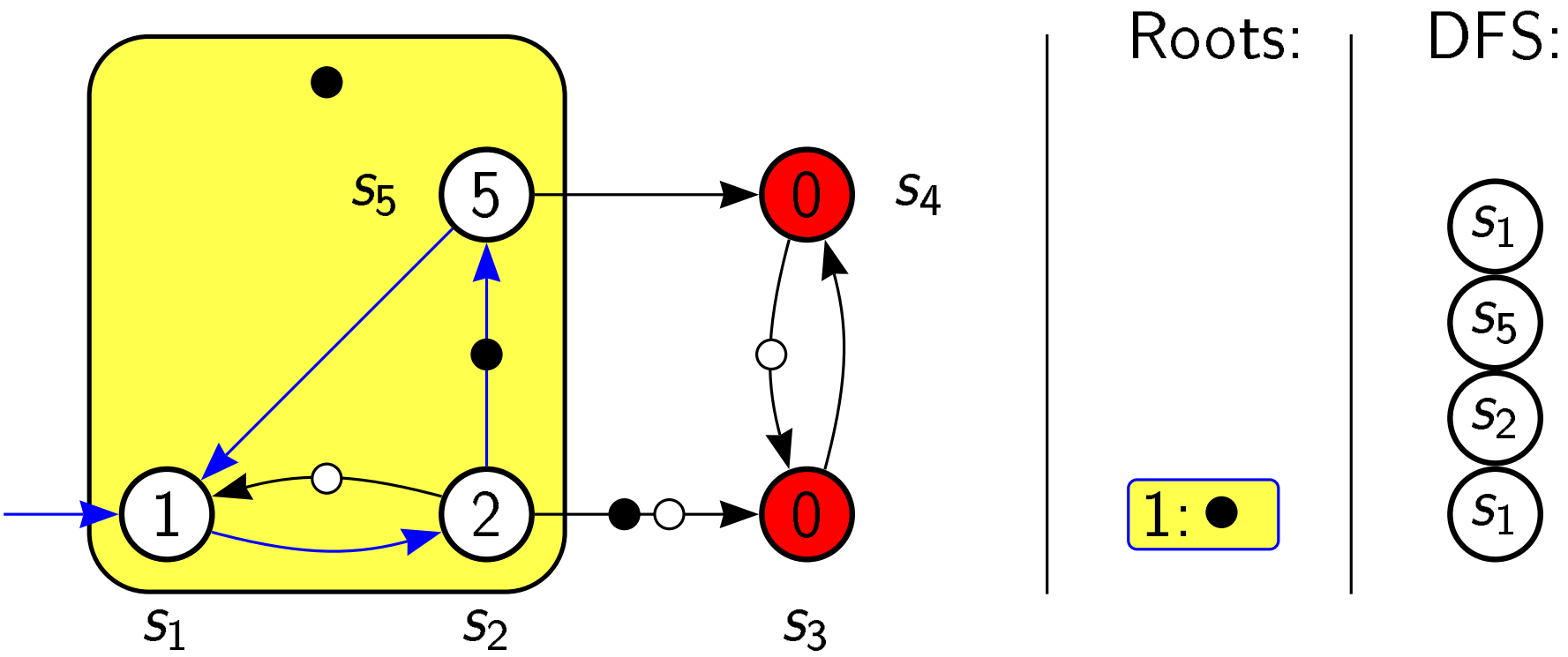
(From: *On-the-fly Verification of Temporal Logic*. Jean-Michel Couvreur. FM'99. LNCS 1708.)

SCC-Based Emptiness Check for TGBA



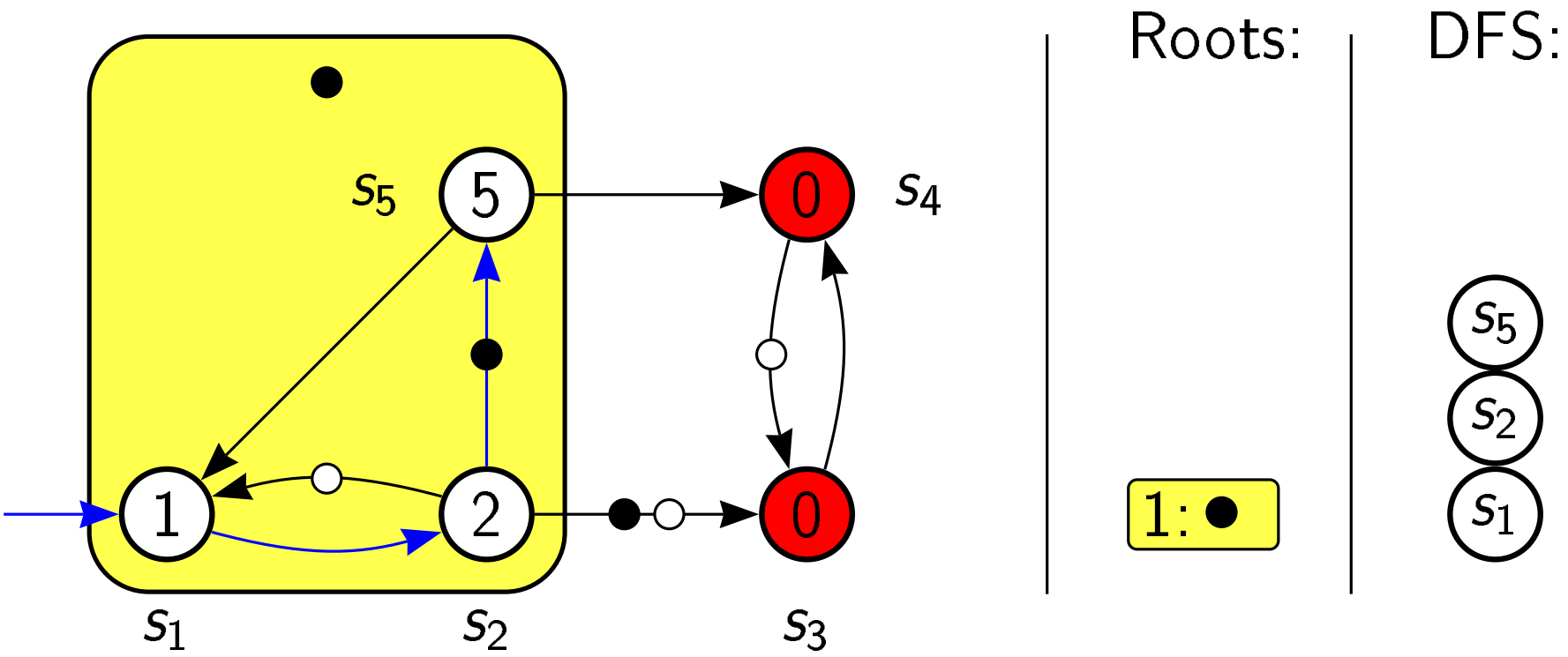
(From: *On-the-fly Verification of Temporal Logic*. Jean-Michel Couvreur. FM'99. LNCS 1708.)

SCC-Based Emptiness Check for TGBA



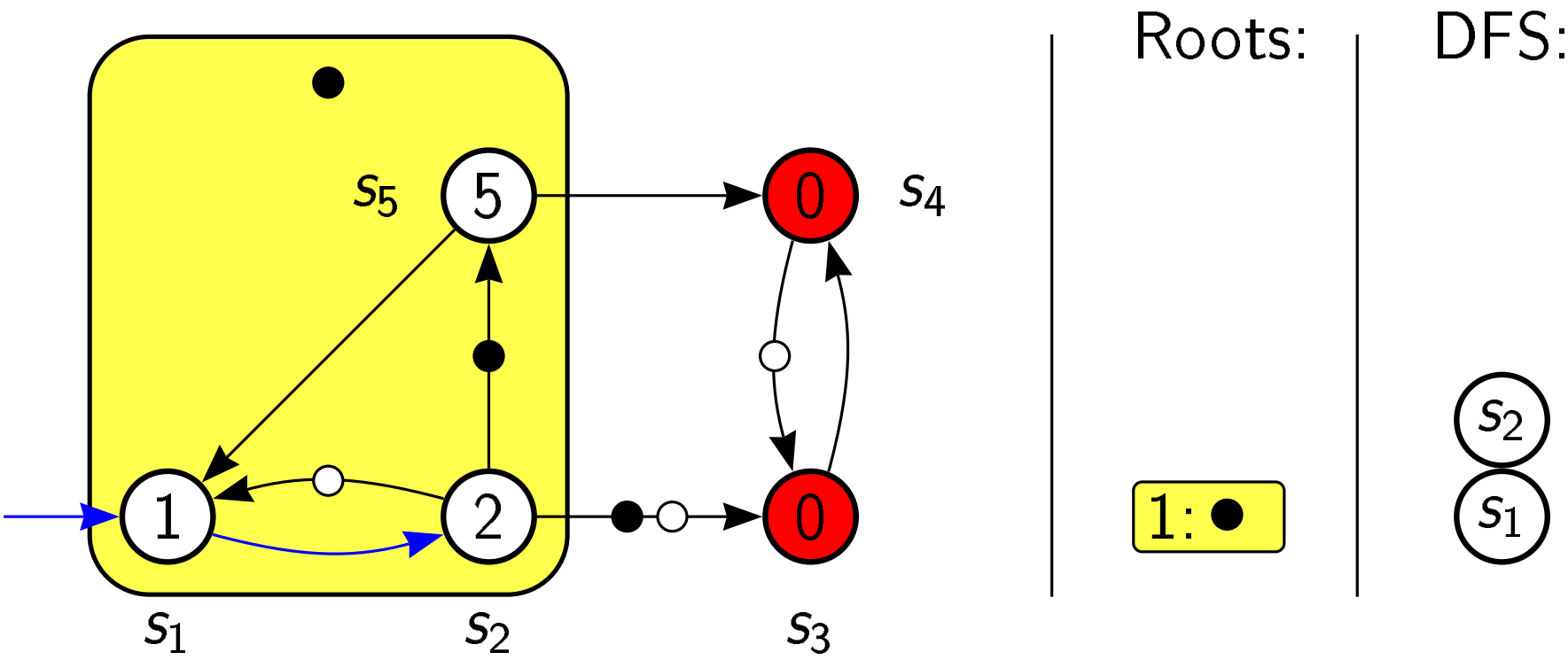
(From: *On-the-fly Verification of Temporal Logic*. Jean-Michel Couvreur. FM'99. LNCS 1708.)

SCC-Based Emptiness Check for TGBA



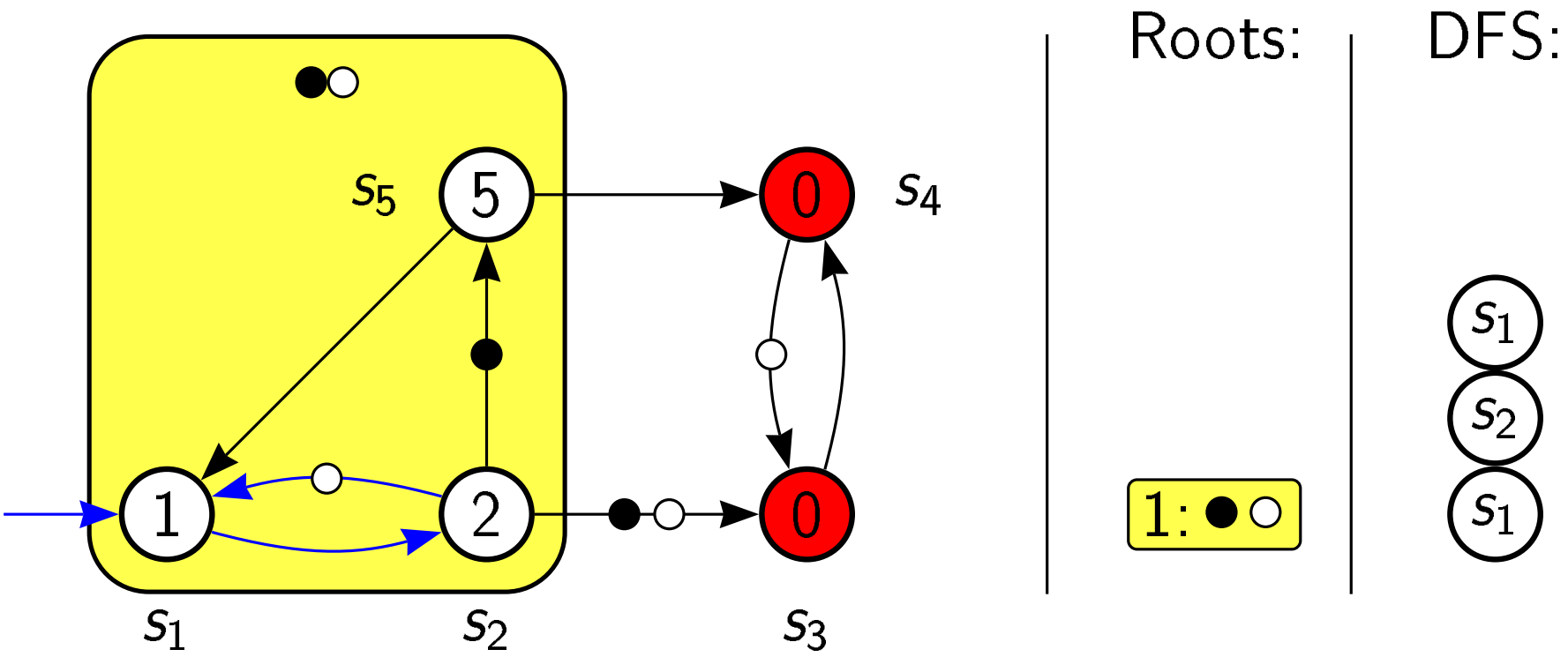
(From: *On-the-fly Verification of Temporal Logic*. Jean-Michel Couvreur. FM'99. LNCS 1708.)

SCC-Based Emptiness Check for TGBA



(From: *On-the-fly Verification of Temporal Logic*. Jean-Michel Couvreur. FM'99. LNCS 1708.)

SCC-Based Emptiness Check for TGBA

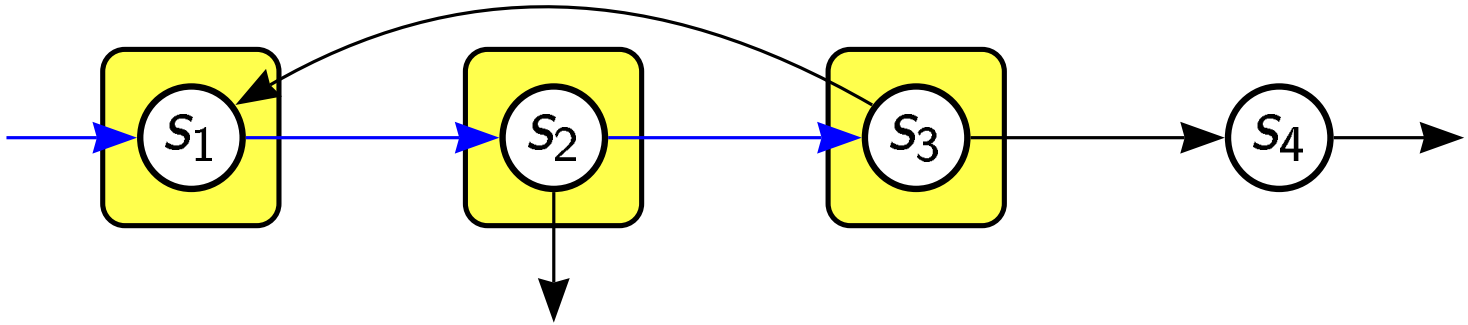


Found!

(From: *On-the-fly Verification of Temporal Logic*. Jean-Michel Couvreur. FM'99. LNCS 1708.)

Two Heuristics for SCCs

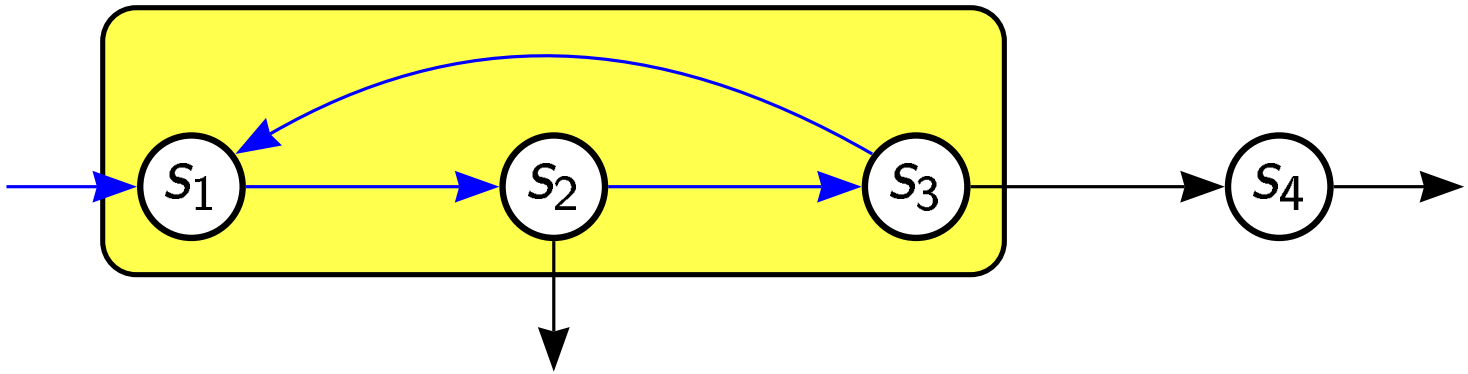
- H1: visit transitions that go to visited states first.



(From: *On-the-Fly Emptiness Checks for Generalized Büchi Automata*. J.-M. Couvreur, A. Duret-Lutz, and D. Poitrenaud. SPIN'05. LNCS 3639.)

Two Heuristics for SCCs

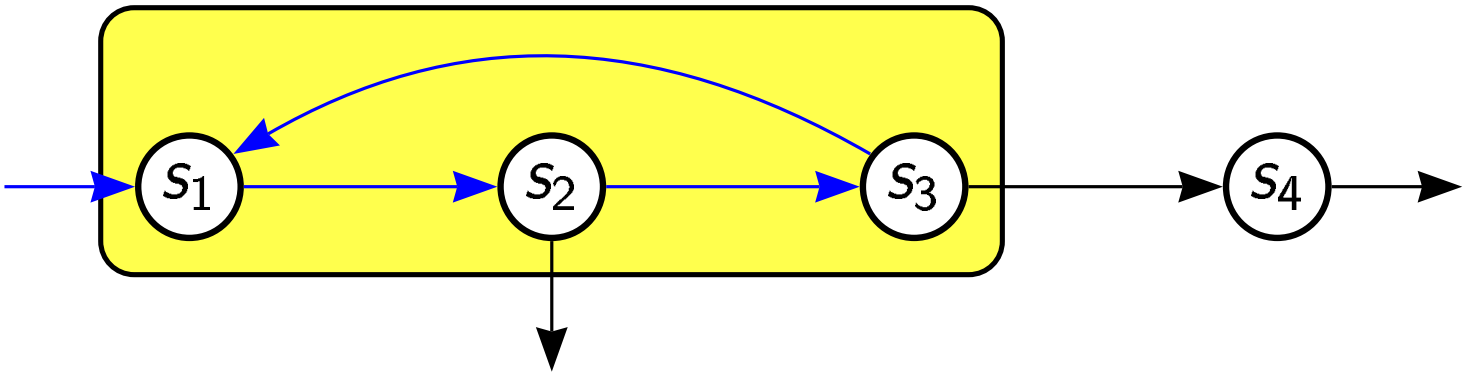
- H1: visit transitions that go to visited states first.



(From: *On-the-Fly Emptiness Checks for Generalized Büchi Automata*. J.-M. Couvreur, A. Duret-Lutz, and D. Poitrenaud. SPIN'05. LNCS 3639.)

Two Heuristics for SCCs

- H1: visit transitions that go to visited states first.

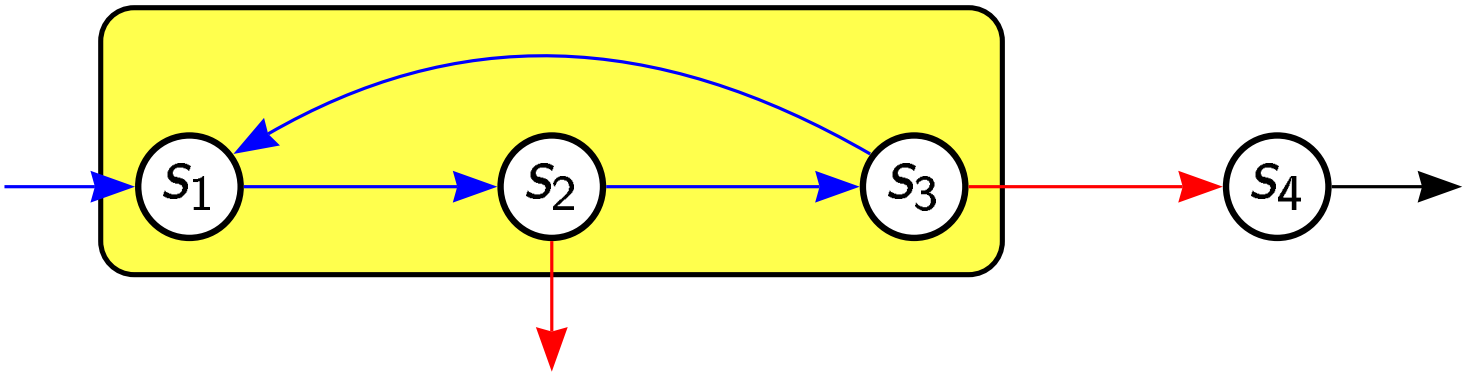


- H2: H1 + consider the DFS in term of SCCs when choosing a successor.

(From: *On-the-Fly Emptiness Checks for Generalized Büchi Automata*. J.-M. Couvreur, A. Duret-Lutz, and D. Poitrenaud. SPIN'05. LNCS 3639.)

Two Heuristics for SCCs

- H1: visit transitions that go to visited states first.



- H2: H1 + consider the DFS in term of SCCs when choosing a successor.

(From: *On-the-Fly Emptiness Checks for Generalized Büchi Automata*. J.-M. Couvreur, A. Duret-Lutz, and D. Poitrenaud. SPIN'05. LNCS 3639.)

Fairness Hypotheses

- Hypotheses on the system to verify.
  - E.g.: two independent processes running on the same host get a "run slice" infinitely often (alternative: model the host's scheduler too)

Fairness Hypotheses

- Hypotheses on the system to verify.
  - E.g.: two independent processes running on the same host get a "run slice" infinitely often (alternative: model the host's scheduler too)
- These hypotheses are constraints for the emptiness check.
  - We want a counterexample where both processes are progressing infinitely often.
  - We can ignore runs where one process is stuck.

Expressing Weak and Strong Fairness with LTL

weak fairness If something happens continuously, something else will happen infinitely often. $\mathbf{F}\,\mathbf{G}\,en \rightarrow \mathbf{G}\,\mathbf{F}\,oc = \mathbf{G}\,\mathbf{F}(\neg en \vee oc)$

strong fairness If something happens infinitely often, something else will happen infinitely often. $\mathbf{G}\,\mathbf{F}\,en \rightarrow \mathbf{G}\,\mathbf{F}\,oc$

Expressing Weak and Strong Fairness with LTL

weak fairness If something happens continuously, something else will happen infinitely often. $\mathbf{F}\,\mathbf{G}\,en \rightarrow \mathbf{G}\,\mathbf{F}\,oc = \mathbf{G}\,\mathbf{F}(\neg en \vee oc)$

strong fairness If something happens infinitely often, something else will happen infinitely often. $\mathbf{G}\,\mathbf{F}\,en \rightarrow \mathbf{G}\,\mathbf{F}\,oc$

To check proposition *prop* under hypothesis *fairness*
we check $fairness \rightarrow prop$.

Expressing Weak and Strong Fairness with LTL

weak fairness If something happens continuously, something else will happen infinitely often. $\mathbf{F}\,\mathbf{G}\,en \rightarrow \mathbf{G}\,\mathbf{F}\,oc = \mathbf{G}\,\mathbf{F}(\neg en \vee oc)$

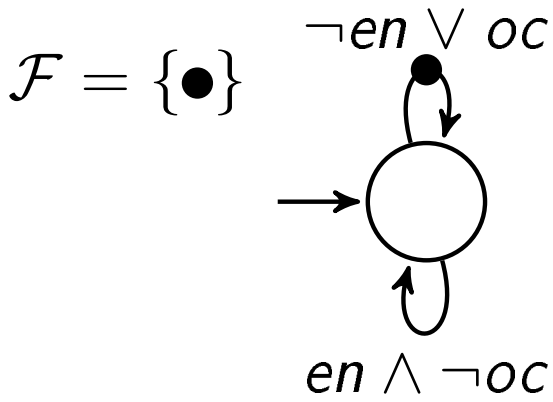
strong fairness If something happens infinitely often, something else will happen infinitely often. $\mathbf{G}\,\mathbf{F}\,en \rightarrow \mathbf{G}\,\mathbf{F}\,oc$

To check proposition *prop* under hypothesis *fairness*
we check $fairness \rightarrow prop$.

$$\begin{aligned}\mathcal{A}_M \otimes \mathcal{A}_{\neg(fairness \rightarrow prop)} &= \mathcal{A}_M \otimes \mathcal{A}_{fairness \wedge \neg prop} \\ &= \mathcal{A}_M \otimes \mathcal{A}_{fairness} \otimes \mathcal{A}_{\neg prop}\end{aligned}$$

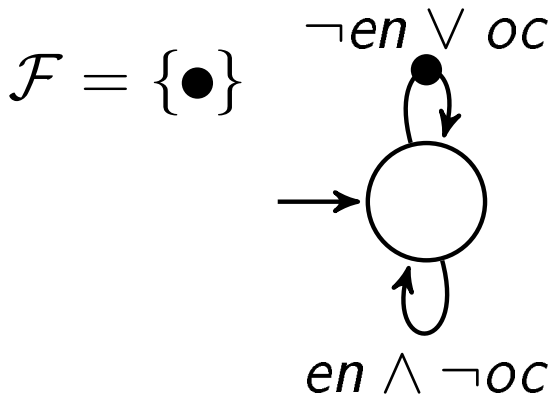
Added complexity depends on $A_{fairness}$.

TGBA for $\mathbf{G}\,\mathbf{F}(\neg en \vee oc)$ (Weak Fairness)



Büchi acceptance conditions correspond to formulæ such as $\mathbf{G}\,\mathbf{F}\,a$. In fact, any formula of the form $\bigwedge_{i=1}^{n} \mathbf{G}\,\mathbf{F}(\neg en_i \vee oc_i)$ can be translated into a 1-state deterministic TGBA (with 2^n transitions, and n acceptance conditions).

TGBA for $\mathbf{G}\,\mathbf{F}(\neg en \vee oc)$ (Weak Fairness)



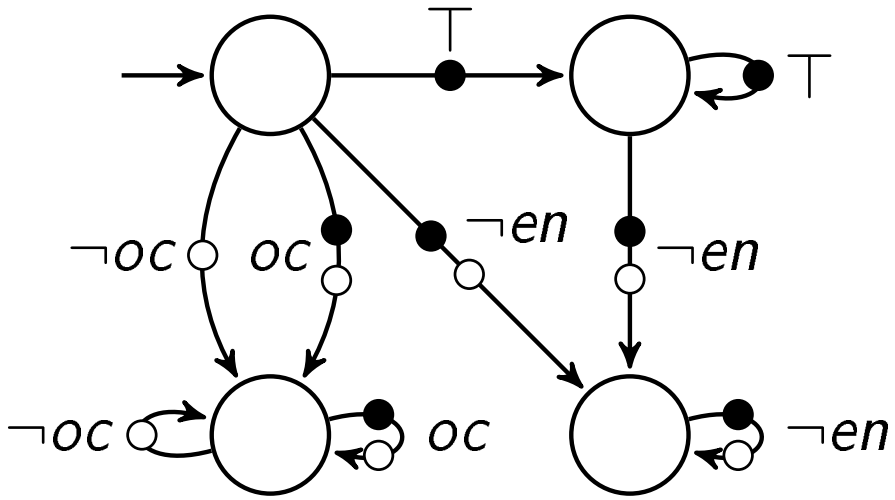
Büchi acceptance conditions correspond to formulæ such as $\mathbf{G}\,\mathbf{F}\,a$.
In fact, any formula of the form $\bigwedge_{i=1}^{n} \mathbf{G}\,\mathbf{F}(\neg en_i \vee oc_i)$ can be translated into a 1-state deterministic TGBA (with 2^n transitions, and n acceptance conditions).

$$A_M \otimes A_{fairness} \otimes A_{\neg prop}$$

In practice the system can be labeled with the appropriate acceptance conditions as the state space is explored.
Weak fairness is free if you have a generalized emptiness check.

TGBA for $\mathbf{G}\,\mathbf{F}\,en \rightarrow \mathbf{G}\,\mathbf{F}\,oc$ (Strong Fairness)

$\mathcal{F} = \{\bullet, \circ\}$



Above is automatically generated. It gets worse for more hypotheses: $\bigwedge_{i=1}^{n}(\mathbf{G}\,\mathbf{F}\,en_i \rightarrow \mathbf{G}\,\mathbf{F}\,oc_i)$ leads to $3^n + 1$ states. We don't know of an LTL translator that does better.

Using an *ad hoc* construction we can build a TGBA with $2^n + 1$ states. We have $2^{O(n)}$ transitions in both cases.

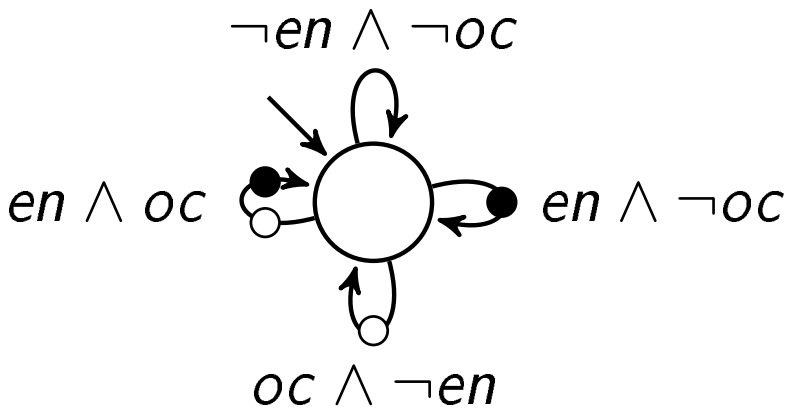
Strong fairness is costly: $A_{fairness}$ adds an exponential blowup in $A_M \otimes A_{fairness} \otimes A_{\neg prop}$.

Streett Automata

- Differ from Büchi automata only in acceptance conditions.
- Acceptance conditions look like "if a run sees $\bullet$ infinitely often, then it will see $\circ$ infinitely often" (can be generalized to more pairs of colors)

Streett Automata

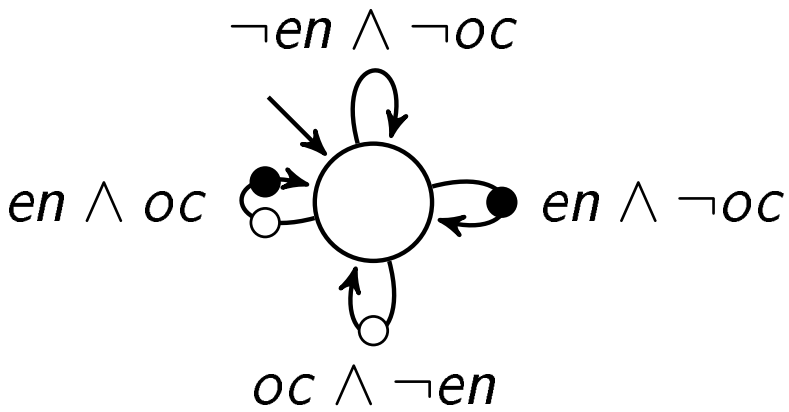
- Differ from Büchi automata only in acceptance conditions.
- Acceptance conditions look like "if a run sees ● infinitely often, then it will see ○ infinitely often" (can be generalized to more pairs of colors)
- Exactly what is needed to recognize $\mathbf{G}\,\mathbf{F}\,en \rightarrow \mathbf{G}\,\mathbf{F}\,oc$:



$$\mathcal{F} = \{(\bullet, \circ)\}$$

Streett Automata

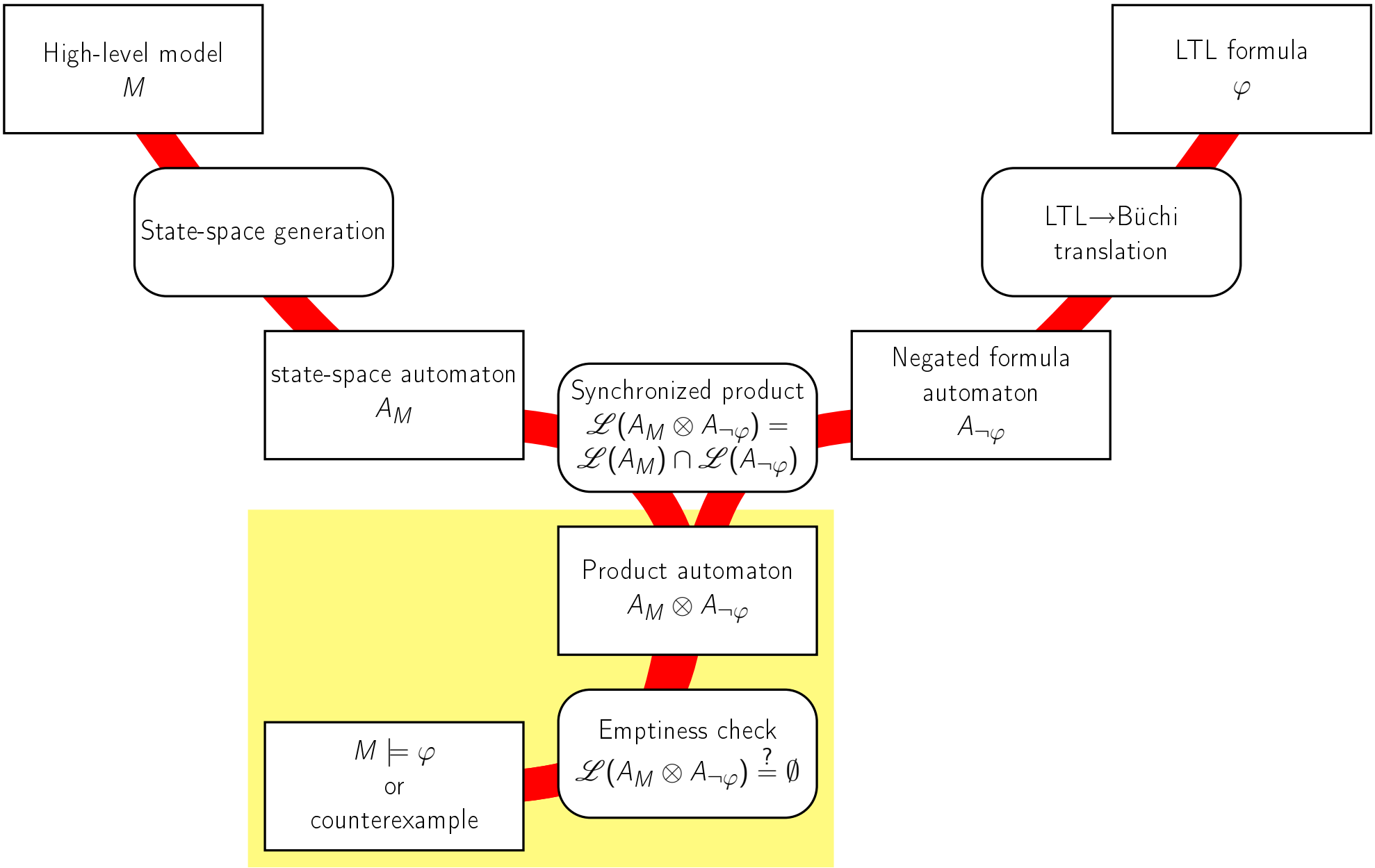
- Differ from Büchi automata only in acceptance conditions.
- Acceptance conditions look like "if a run sees ● infinitely often, then it will see ○ infinitely often" (can be generalized to more pairs of colors)
- Exactly what is needed to recognize $\mathbf{G}\,\mathbf{F}\,en \rightarrow \mathbf{G}\,\mathbf{F}\,oc$:



$$\mathcal{F} = \{(\bullet, \circ)\}$$

- $\bigwedge_{i=1}^{n}(\mathbf{G}\,\mathbf{F}\,en_i \rightarrow \mathbf{G}\,\mathbf{F}\,oc_i)$ can be converted into a 1-state deterministic Streett automaton.
  In practice the system can be labeled with the appropriate acceptance conditions as the state space is explored.

Automata Theoretic Approach to Model Checking



Emptiness Check for Streett Automata

- The basic idea (using SCCs) has been known for a long time (Lichtenstein and Pnueli in 1985, Emerson and Lei in 1987).
- Our algorithm is close to that of Latvala and Heljanko (2000). Their algorithm work for state-based Streett automata.

E.g. using $\mathcal{F} = \{(\bullet, \circ), (\color{red}\bullet, \color{green}\bullet)\}$

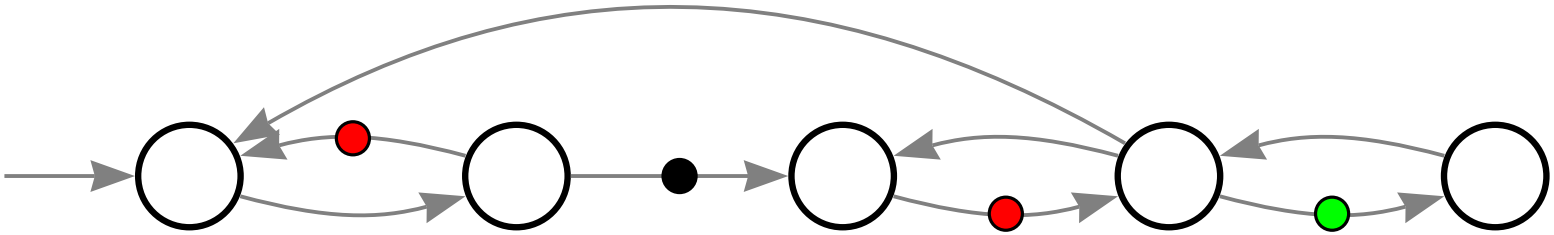
1. Build a list of all the reachable SCCs of the automata.
2. While the list isn't empty, pick an SCC off the list and do:
   - If it's a trivial SCC (single state without self-loop), drop it.
   - If the acceptance conditions of all the states in the SCC verifies $\bullet \Rightarrow \circ \wedge \color{red}\bullet \Rightarrow \color{green}\bullet$, then report non-emptiness.
   - Otherwise, there exists some bad states in the SCC: states labeled by a left-color with no right-color match in the SCC.
     - Erase these bad states
     - Recompute the SCCs from the remaining states, and add them to the list of SCCs.
3. Once the list is empty, report emptiness.

What Do We Want?

- An algorithm that will work with on-the-fly constructions: let's not construct **all** SCCs unless necessary. This part is rather easy.
- A transition-based algorithm. Because it is harder to erase a transition (especially since we work on the fly) we will use *barriers*.

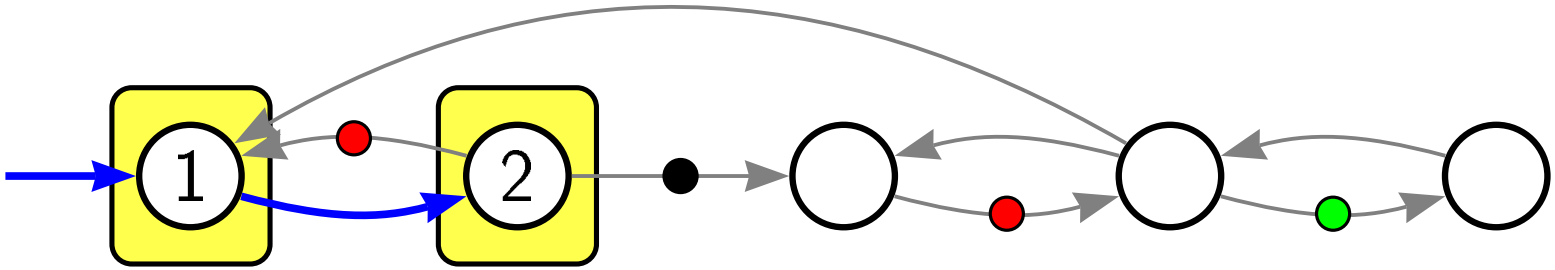
Emptiness of Transition-based Streett Automata

- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



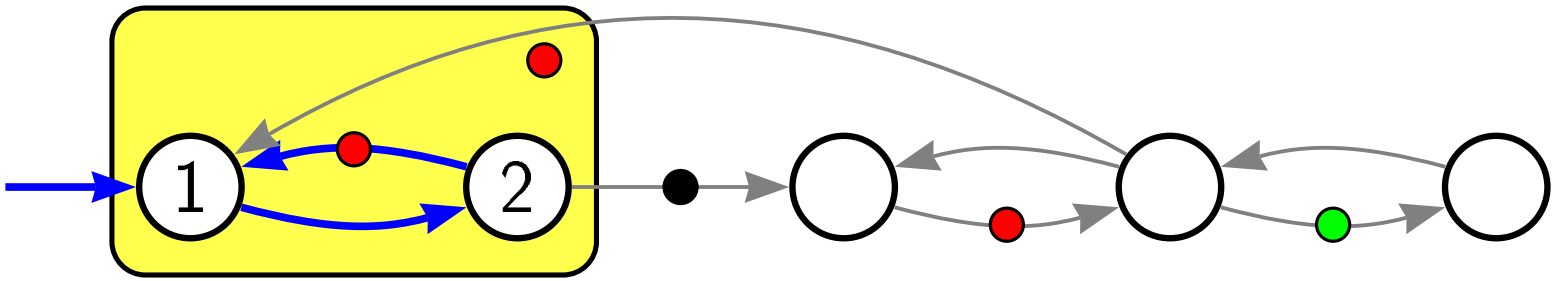
Emptiness of Transition-based Streett Automata

- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



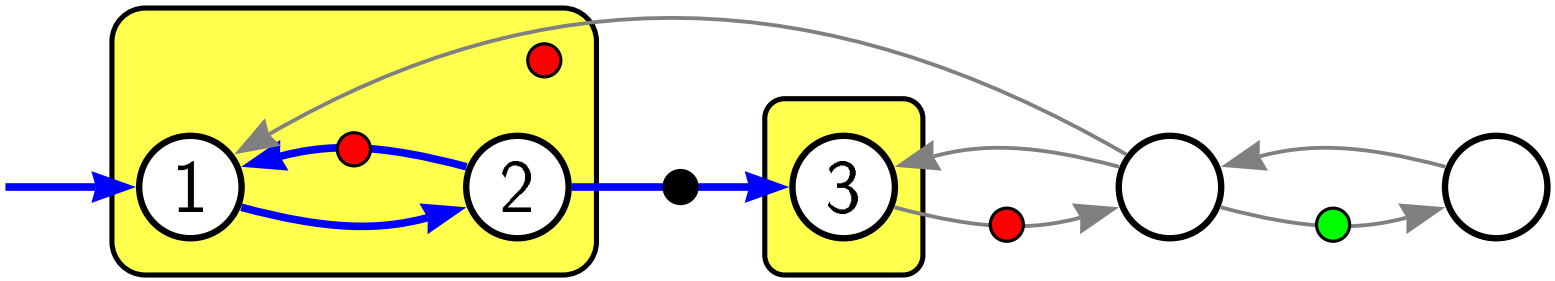
Emptiness of Transition-based Streett Automata

- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



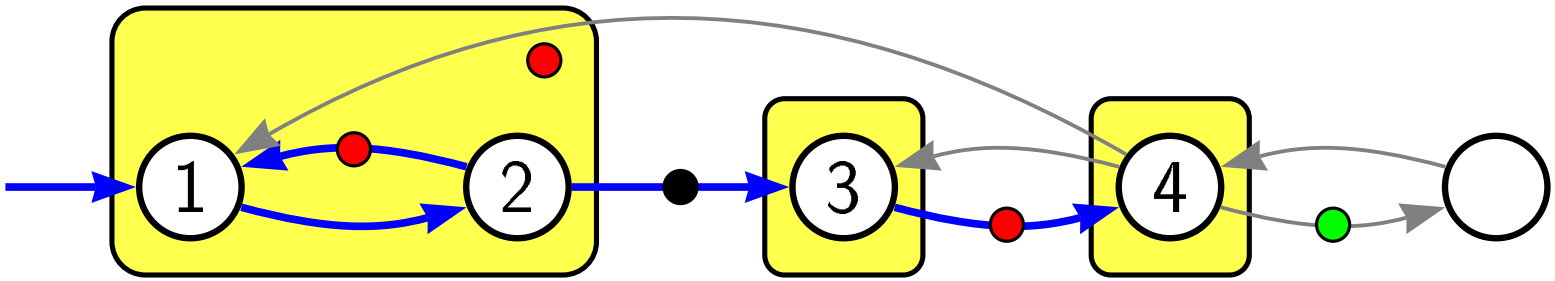
Emptiness of Transition-based Streett Automata

- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



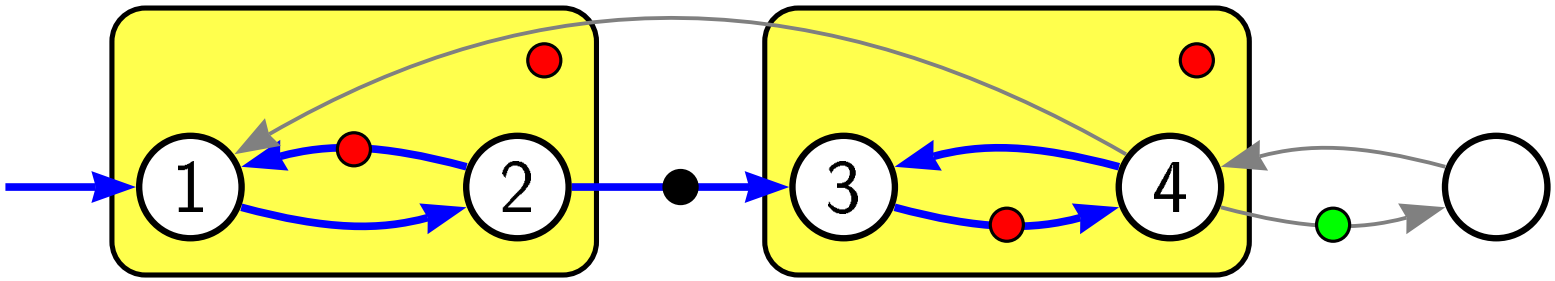
Emptiness of Transition-based Streett Automata

- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



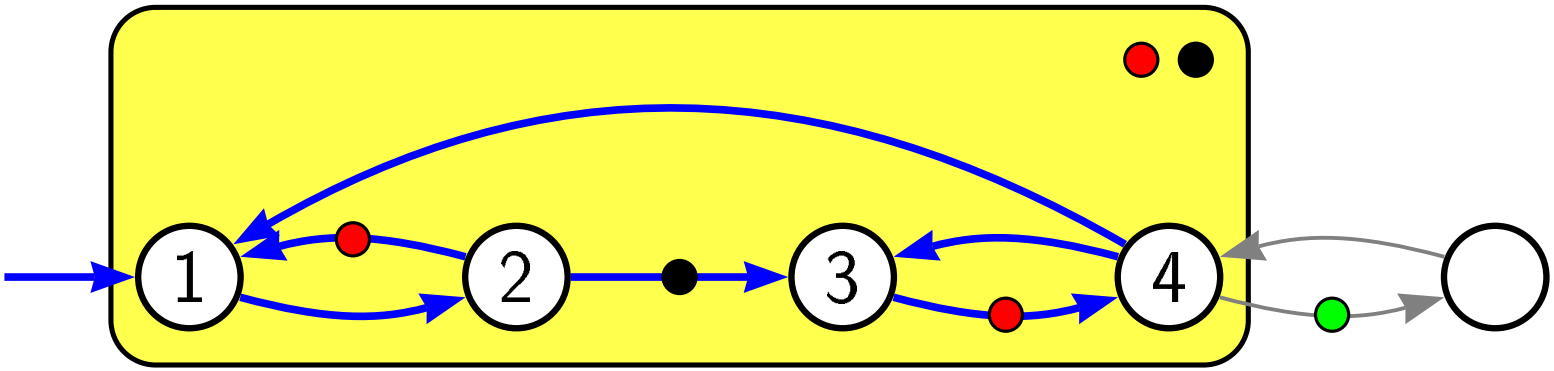
Emptiness of Transition-based Streett Automata

- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



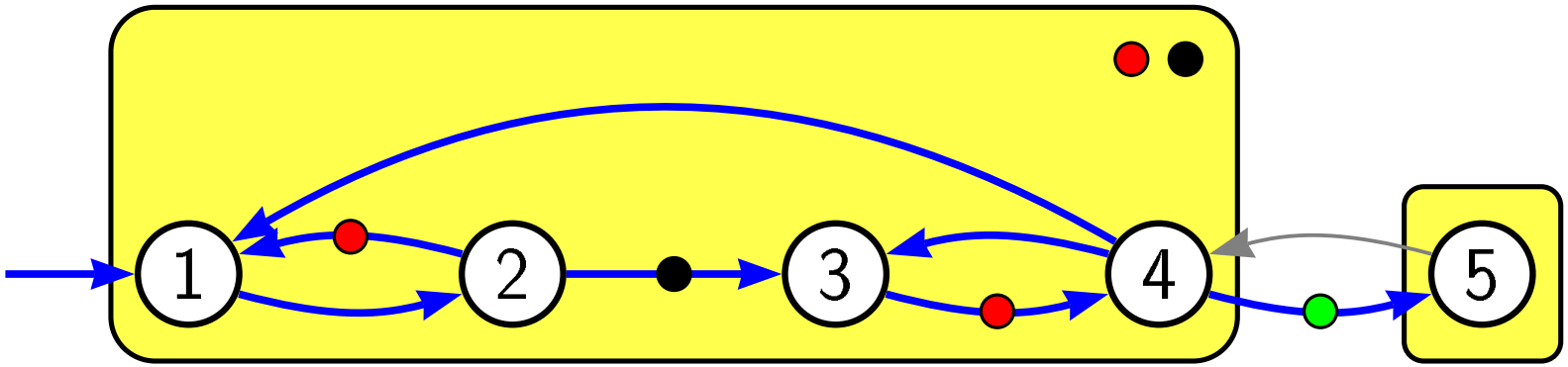
Emptiness of Transition-based Streett Automata

- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



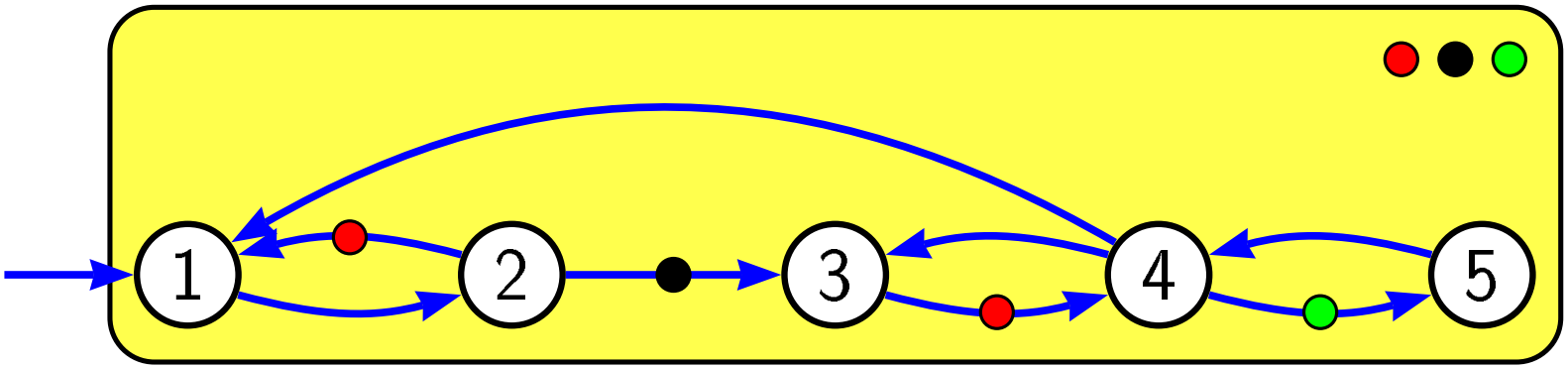
Emptiness of Transition-based Streett Automata

- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



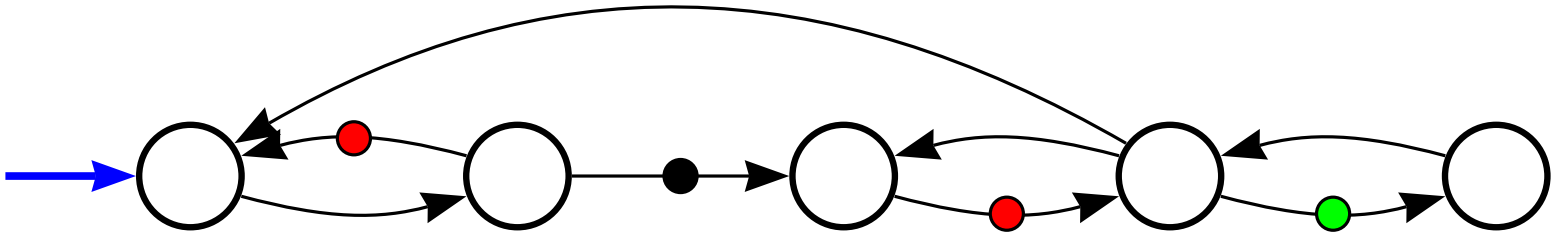
Emptiness of Transition-based Streett Automata

- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



Emptiness of Transition-based Streett Automata

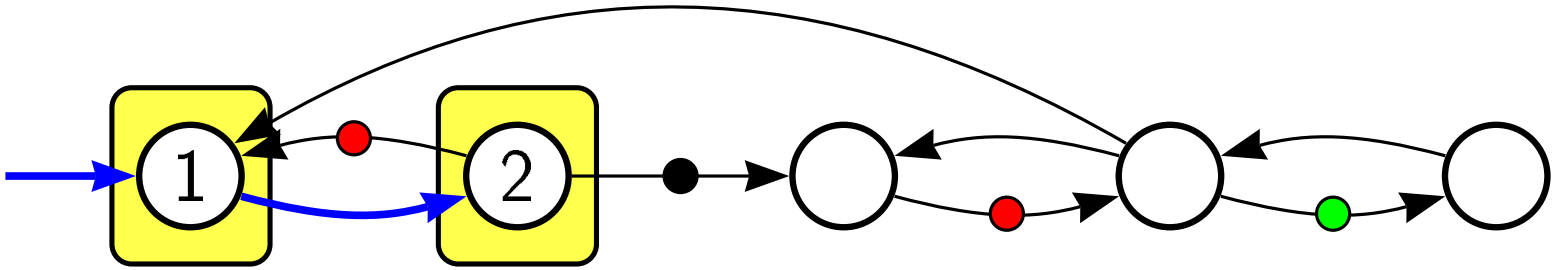
- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



Since there is no $\circ$ in this SCC, we start again, but read $\bullet$ as a one-way "barrier" forbidding attempts to come back.

Emptiness of Transition-based Streett Automata

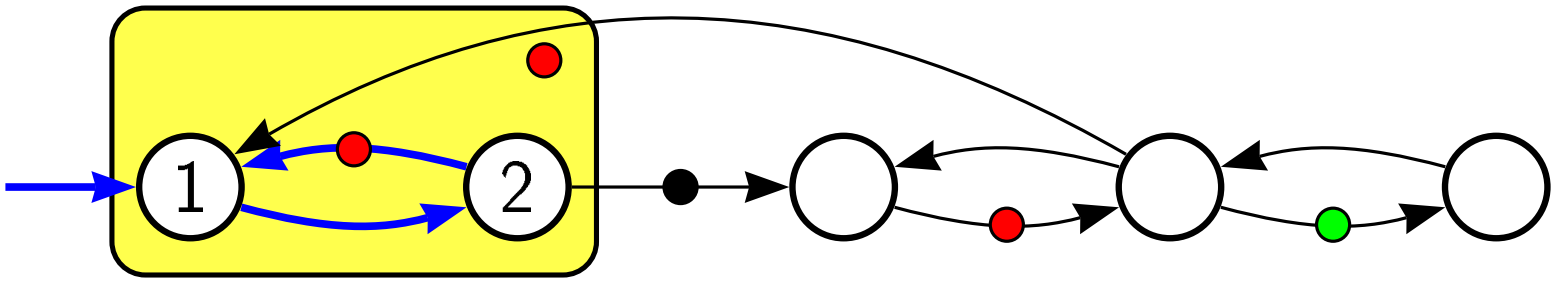
- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



Since there is no $\circ$ in this SCC, we start again, but read $\bullet$ as a one-way "barrier" forbidding attempts to come back.

Emptiness of Transition-based Streett Automata

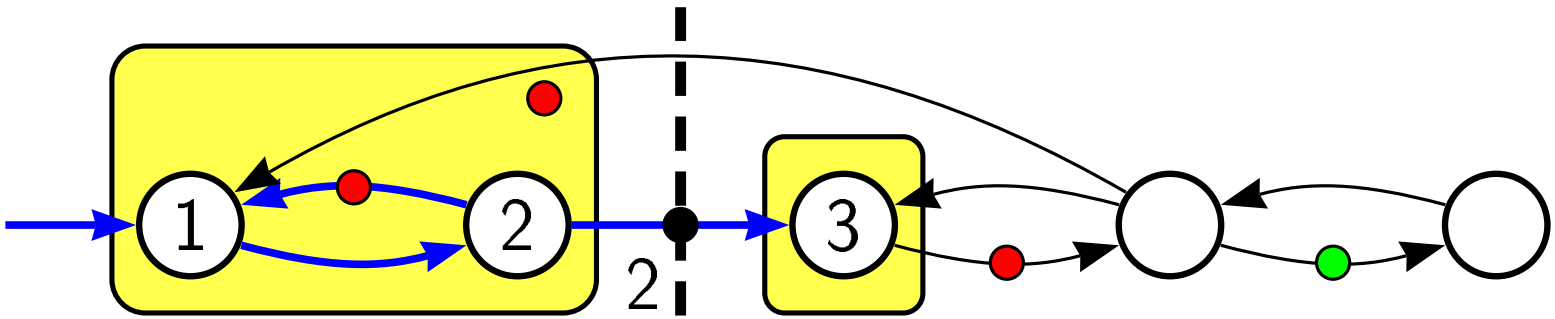
- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



Since there is no $\circ$ in this SCC, we start again, but read $\bullet$ as a one-way "barrier" forbidding attempts to come back.

Emptiness of Transition-based Streett Automata

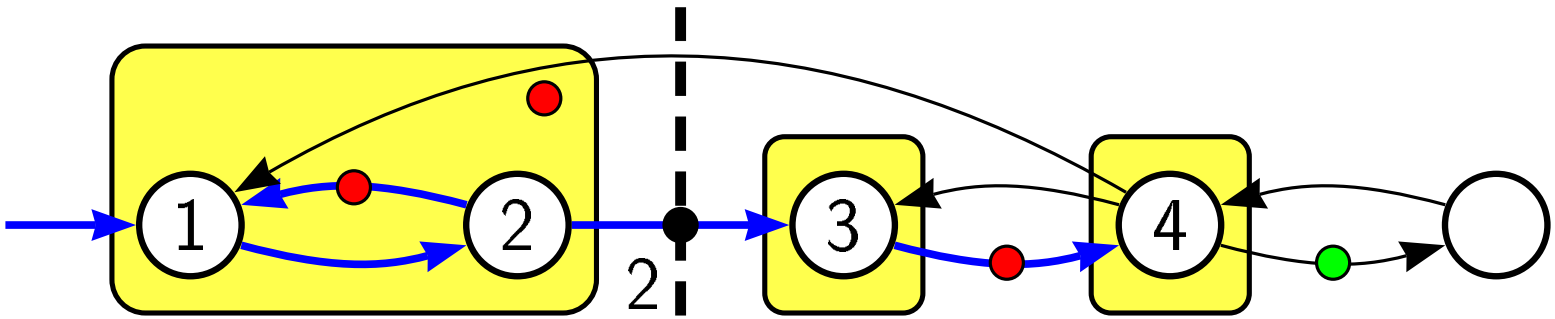
- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



Since there is no $\circ$ in this SCC, we start again, but read $\bullet$ as a one-way "barrier" forbidding attempts to come back.

Emptiness of Transition-based Streett Automata

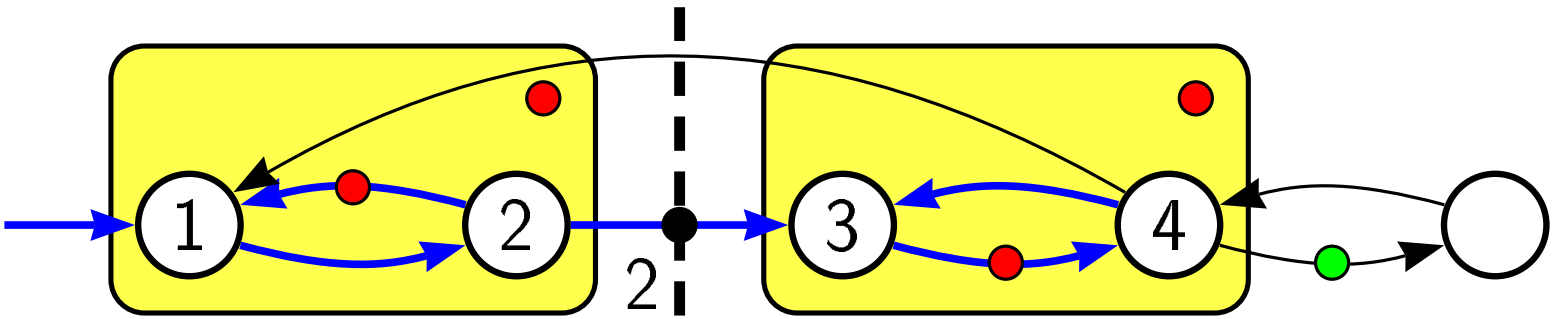
- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



Since there is no $\circ$ in this SCC, we start again, but read $\bullet$ as a one-way "barrier" forbidding attempts to come back.

Emptiness of Transition-based Streett Automata

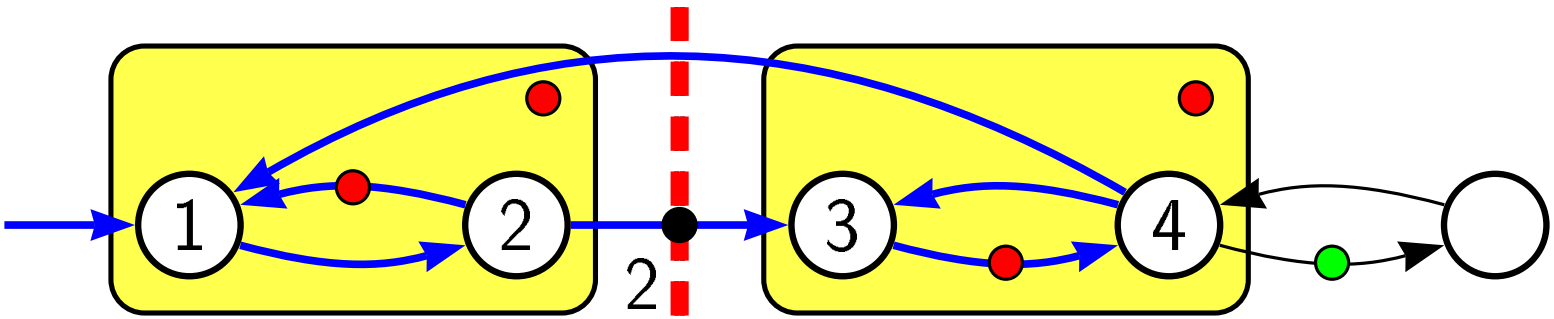
- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



Since there is no $\circ$ in this SCC, we start again, but read $\bullet$ as a one-way "barrier" forbidding attempts to come back.

Emptiness of Transition-based Streett Automata

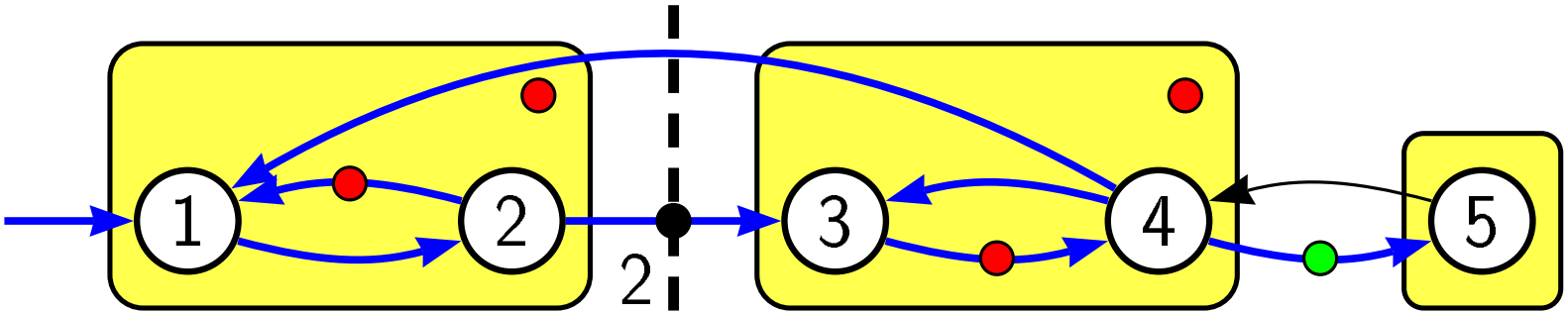
- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



Since there is no $\circ$ in this SCC, we start again, but read $\bullet$ as a one-way "barrier" forbidding attempts to come back.

Emptiness of Transition-based Streett Automata

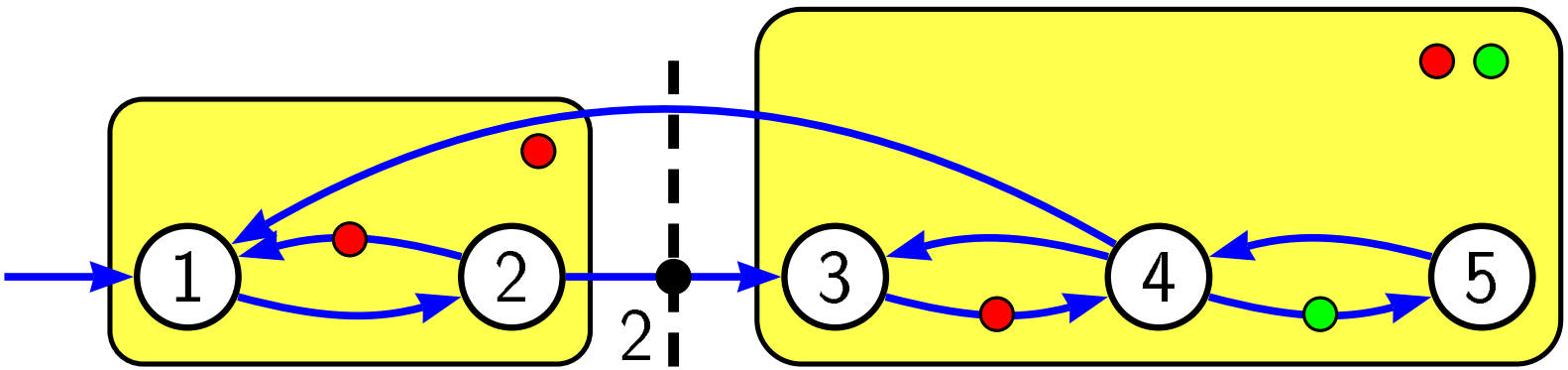
- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



Since there is no $\circ$ in this SCC, we start again, but read $\bullet$ as a one-way "barrier" forbidding attempts to come back.

Emptiness of Transition-based Streett Automata

- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



Since there is no $\circ$ in this SCC, we start again, but read $\bullet$ as a one-way "barrier" forbidding attempts to come back.

Finally

- To be correct, has to be combined with a variant of heuristic H2 (ordering successors SCC-wise)
- "Barriers" must be crossed only after all normal successors have been visited.

Slowdown (w.r.t. generalized Büchi emptiness check):

- SCC are revisited at worst m times if m pairs of acceptance conditions.
- Compare with the $2^m + 1$ states of the Büchi automaton for the corresponding LTL formula...

Our motivation was fairness hypotheses, but what about running the whole approach using Streett automata? Streett automata are exponentially more succinct than Büchi automata, but can we translate LTL to Streett efficiently?

A small LTL formula...

$$\Big((\mathbf{G}\,\mathbf{F}\,p_0 \rightarrow \mathbf{G}\,\mathbf{F}\,p_1) \wedge (\mathbf{G}\,\mathbf{F}\,p_2 \rightarrow \mathbf{G}\,\mathbf{F}\,p_0) \wedge$$
$$(\mathbf{G}\,\mathbf{F}\,p_3 \rightarrow \mathbf{G}\,\mathbf{F}\,p_2) \wedge (\mathbf{G}\,\mathbf{F}\,p_4 \rightarrow \mathbf{G}\,\mathbf{F}\,p_2) \wedge$$
$$(\mathbf{G}\,\mathbf{F}\,p_5 \rightarrow \mathbf{G}\,\mathbf{F}\,p_3) \wedge (\mathbf{G}\,\mathbf{F}\,p_6 \rightarrow \mathbf{G}\,\mathbf{F}(p_5 \vee p_4)) \wedge$$
$$(\mathbf{G}\,\mathbf{F}\,p_7 \rightarrow \mathbf{G}\,\mathbf{F}\,p_6) \wedge (\mathbf{G}\,\mathbf{F}\,p_1 \rightarrow \mathbf{G}\,\mathbf{F}\,p_7)\Big) \rightarrow \mathbf{G}\,\mathbf{F}\,p_8$$

How many states to encode the negation in a Büchi automaton?

A small LTL formula...

$$\Big((\mathbf{G}\,\mathbf{F}\,p_0 \rightarrow \mathbf{G}\,\mathbf{F}\,p_1) \wedge (\mathbf{G}\,\mathbf{F}\,p_2 \rightarrow \mathbf{G}\,\mathbf{F}\,p_0) \wedge$$
$$(\mathbf{G}\,\mathbf{F}\,p_3 \rightarrow \mathbf{G}\,\mathbf{F}\,p_2) \wedge (\mathbf{G}\,\mathbf{F}\,p_4 \rightarrow \mathbf{G}\,\mathbf{F}\,p_2) \wedge$$
$$(\mathbf{G}\,\mathbf{F}\,p_5 \rightarrow \mathbf{G}\,\mathbf{F}\,p_3) \wedge (\mathbf{G}\,\mathbf{F}\,p_6 \rightarrow \mathbf{G}\,\mathbf{F}(p_5 \vee p_4)) \wedge$$
$$(\mathbf{G}\,\mathbf{F}\,p_7 \rightarrow \mathbf{G}\,\mathbf{F}\,p_6) \wedge (\mathbf{G}\,\mathbf{F}\,p_1 \rightarrow \mathbf{G}\,\mathbf{F}\,p_7)\Big) \rightarrow \mathbf{G}\,\mathbf{F}\,p_8$$

How many states to encode the negation in a Büchi automaton?

Spot's LTL to Büchi translation without optimizations : 7291 states.
With optimizations : 1731 states.
Sebastiani et al. (CAV'05) dedicated translation : 1281 states.

A small LTL formula...

$$\Big((\mathbf{G}\,\mathbf{F}\,p_0 \rightarrow \mathbf{G}\,\mathbf{F}\,p_1) \wedge (\mathbf{G}\,\mathbf{F}\,p_2 \rightarrow \mathbf{G}\,\mathbf{F}\,p_0) \wedge$$
$$(\mathbf{G}\,\mathbf{F}\,p_3 \rightarrow \mathbf{G}\,\mathbf{F}\,p_2) \wedge (\mathbf{G}\,\mathbf{F}\,p_4 \rightarrow \mathbf{G}\,\mathbf{F}\,p_2) \wedge$$
$$(\mathbf{G}\,\mathbf{F}\,p_5 \rightarrow \mathbf{G}\,\mathbf{F}\,p_3) \wedge (\mathbf{G}\,\mathbf{F}\,p_6 \rightarrow \mathbf{G}\,\mathbf{F}(p_5 \vee p_4)) \wedge$$
$$(\mathbf{G}\,\mathbf{F}\,p_7 \rightarrow \mathbf{G}\,\mathbf{F}\,p_6) \wedge (\mathbf{G}\,\mathbf{F}\,p_1 \rightarrow \mathbf{G}\,\mathbf{F}\,p_7)\Big) \rightarrow \mathbf{G}\,\mathbf{F}\,p_8$$

How many states to encode the negation as a Streett automaton?

A small LTL formula...

$$\Big((\mathbf{G}\,\mathbf{F}\,p_0 \to \mathbf{G}\,\mathbf{F}\,p_1) \wedge (\mathbf{G}\,\mathbf{F}\,p_2 \to \mathbf{G}\,\mathbf{F}\,p_0) \wedge$$
$$(\mathbf{G}\,\mathbf{F}\,p_3 \to \mathbf{G}\,\mathbf{F}\,p_2) \wedge (\mathbf{G}\,\mathbf{F}\,p_4 \to \mathbf{G}\,\mathbf{F}\,p_2) \wedge$$
$$(\mathbf{G}\,\mathbf{F}\,p_5 \to \mathbf{G}\,\mathbf{F}\,p_3) \wedge (\mathbf{G}\,\mathbf{F}\,p_6 \to \mathbf{G}\,\mathbf{F}(p_5 \vee p_4)) \wedge$$
$$(\mathbf{G}\,\mathbf{F}\,p_7 \to \mathbf{G}\,\mathbf{F}\,p_6) \wedge (\mathbf{G}\,\mathbf{F}\,p_1 \to \mathbf{G}\,\mathbf{F}\,p_7)\Big) \to \mathbf{G}\,\mathbf{F}\,p_8$$

How many states to encode the negation as a Streett automaton?
Formula of the form $\psi \to \varphi$ where ψ is a strong fairness hypothesis.

$$\mathcal{A}_{\neg(\psi\to\varphi)} = \mathcal{A}_{\psi} \otimes \mathcal{A}_{\neg\varphi}$$

$\mathcal{A}_{\psi}$: 1-state deterministic Streett automaton with 8 pairs of acc.cond.
$\mathcal{A}_{\neg\varphi} = \mathcal{A}_{\mathbf{F}\,\mathbf{G}\,\neg p_8}$: 2-state Büchi automaton with 1 acc.cond.
Therefore $\mathcal{A}_{\neg(\psi\to\varphi)}$ can be represented as a 2-state Streett automaton with 9 pairs of acc.cond.

A small LTL formula...

$$\Big((\mathbf{G}\,\mathbf{F}\,p_0 \rightarrow \mathbf{G}\,\mathbf{F}\,p_1) \wedge (\mathbf{G}\,\mathbf{F}\,p_2 \rightarrow \mathbf{G}\,\mathbf{F}\,p_0) \wedge$$
$$(\mathbf{G}\,\mathbf{F}\,p_3 \rightarrow \mathbf{G}\,\mathbf{F}\,p_2) \wedge (\mathbf{G}\,\mathbf{F}\,p_4 \rightarrow \mathbf{G}\,\mathbf{F}\,p_2) \wedge$$
$$(\mathbf{G}\,\mathbf{F}\,p_5 \rightarrow \mathbf{G}\,\mathbf{F}\,p_3) \wedge (\mathbf{G}\,\mathbf{F}\,p_6 \rightarrow \mathbf{G}\,\mathbf{F}(p_5 \vee p_4)) \wedge$$
$$(\mathbf{G}\,\mathbf{F}\,p_7 \rightarrow \mathbf{G}\,\mathbf{F}\,p_6) \wedge (\mathbf{G}\,\mathbf{F}\,p_1 \rightarrow \mathbf{G}\,\mathbf{F}\,p_7)\Big) \rightarrow \mathbf{G}\,\mathbf{F}\,p_8$$

How many states to encode the negation in a Büchi automaton?

Spot's LTL to Büchi translation without optimizations : 7291 states.
With optimizations : 1731 states.
Sebastiani et al. (CAV'05) dedicated translation : 1281 states.
A n states Streett automaton with m acceptance pairs can be converted into a $n \times (2^m + 1)$ state TGBA. Therefore for this formula we can build a TGBA with $2 \times (2^9 + 1) = 1026$ states.