Fairness

Alexandre Duret-Lutz

avril 2009

Fairness

- Hypotheses on the system to verify.
- You may consider fairness hypothesis anytime you have to deal with a repeated choice.
  - Messages sent over a lossy channel will be delivered after a finite number of retries (the repeated choice is whether the channel will lose the message)
  - Starvation-free resource allocator: anybody requesting the resource eventually gets it.
  - Two independent processes running on the same host get a "run slice" infinitely often (the repeated choice is done by the scheduler).
  - Finite wait: a process blocked until it is granted a resource will eventually get it.

Is there any difference between the last three examples?

Fairness hypotheses & the Emptiness Check

These hypotheses can be seen as constraints for the emptiness check.

If two independent processes running on the same host should get a "run slice" infinitely often:

- We want a counterexample where both processes are progressing infinitely often.
- We can ignore runs where one process is stuck.

Do we really need to modify the emptiness check algorithm?

Expressing Fairness with LTL

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

Expressing Fairness with LTL

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

$$\mathcal{A}_M \otimes \mathcal{A}_{\neg(\mathit{fairness} \rightarrow \mathit{prop})}$$

Expressing Fairness with LTL

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

$$\mathcal{A}_M \otimes \mathcal{A}_{\neg(\mathit{fairness} \rightarrow \mathit{prop})}$$
$$\mathcal{A}_M \otimes \mathcal{A}_{\mathit{fairness} \wedge \neg \mathit{prop}}$$
$$\mathcal{A}_M \otimes \mathcal{A}_{\mathit{fairness}} \otimes \mathcal{A}_{\neg \mathit{prop}}$$

Added complexity depends on A_{fairness}.

Kinds of Fairness Hypotheses

unconditional fairness Something will happen infinitely often.

weak fairness If something happens continuously, something else will happen infinitely often.

strong fairness If something happens infinitely often, something else will happen infinitely often.

Kinds of Fairness Hypotheses

unconditional fairness Something will happen infinitely often. $\mathbf{G}\,\mathbf{F}\,p$

weak fairness If something happens continuously, something else will happen infinitely often.

strong fairness If something happens infinitely often, something else will happen infinitely often.

Kinds of Fairness Hypotheses

unconditional fairness Something will happen infinitely often. $\mathbf{G}\,\mathbf{F}\,p$

weak fairness If something happens continuously, something else will happen infinitely often. $\mathbf{F}\,\mathbf{G}\,e \rightarrow \mathbf{G}\,\mathbf{F}\,e'$

strong fairness If something happens infinitely often, something else will happen infinitely often.

Kinds of Fairness Hypotheses

unconditional fairness Something will happen infinitely often. $\mathbf{G}\,\mathbf{F}\,p$

weak fairness If something happens continuously, something else will happen infinitely often. $\mathbf{F}\,\mathbf{G}\,e \rightarrow \mathbf{G}\,\mathbf{F}\,e'$

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

Kinds of Fairness Hypotheses

unconditional fairness Something will happen infinitely often. $\mathbf{G}\,\mathbf{F}\,p$

weak fairness If something happens continuously, something else will happen infinitely often. $\mathbf{F}\,\mathbf{G}\,e \rightarrow \mathbf{G}\,\mathbf{F}\,e'$

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

We have

$$\begin{aligned}\mathbf{F}\,\mathbf{G}\,e \rightarrow \mathbf{G}\,\mathbf{F}\,e' &= \neg\,\mathbf{F}\,\mathbf{G}\,e \vee \mathbf{G}\,\mathbf{F}\,e' \\ &= \mathbf{G}\,\mathbf{F}\,\neg e \vee \mathbf{G}\,\mathbf{F}\,e' \\ &= \mathbf{G}\,\mathbf{F}(\neg e \vee e')\end{aligned}$$

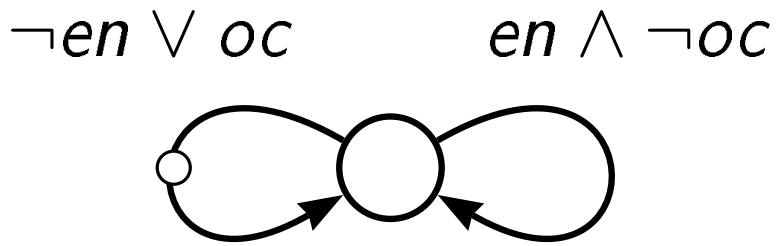
therefore *weak fairness* can be expressed as *unconditional fairness*.

Size of $A_{fairness}$ (Generalized Büchi)

We may want to apply several fairness hypotheses.

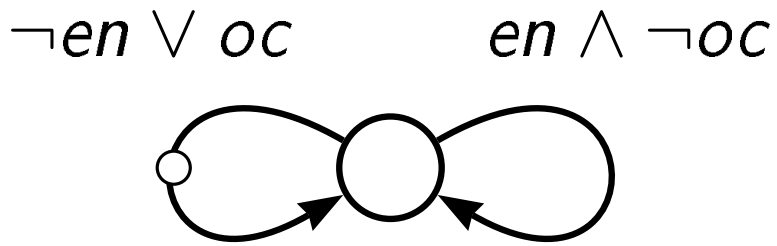
	weak fairness	strong fairness
n	$\bigwedge_{i=1}^{n} \mathbf{G}\,\mathbf{F}(\neg en_i \vee oc_i)$	$\bigwedge_{i=1}^{n} \mathbf{G}\,\mathbf{F}\,en_i \rightarrow \mathbf{G}\,\mathbf{F}\,oc_i$
1	1 state	4 states
2	1 state	10 states
3	1 state	28 states
4	1 state	82 states
⋮	⋮	⋮
n	1 state	$3^n + 1$ states
	deterministic	non-deterministic

Look of $A_{\mathbf{G}\,\mathbf{F}(\neg en \vee oc)}$



Büchi acceptance conditions correspond to formulæ such as $\mathbf{G}\,\mathbf{F}\,a$.

Look of $A_{\mathbf{G}\,\mathbf{F}(\neg en \vee oc)}$

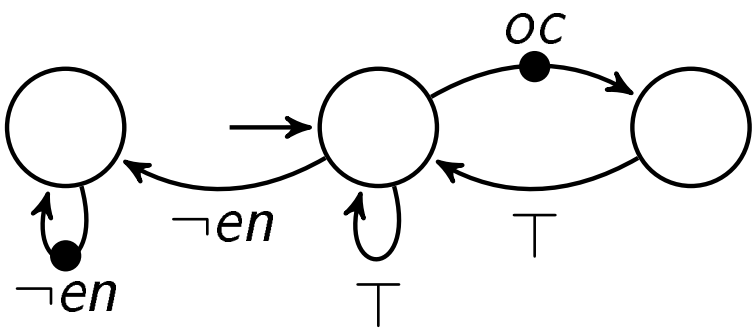


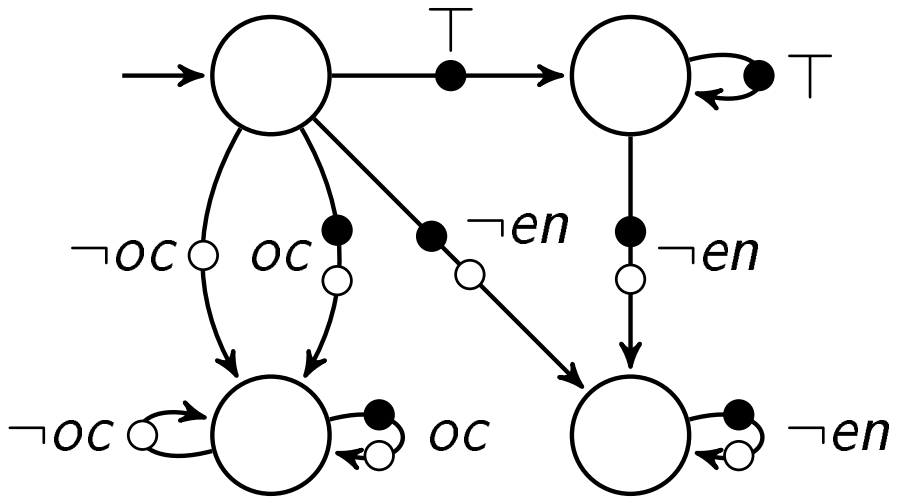
Büchi acceptance conditions correspond to formulæ such as $\mathbf{G}\,\mathbf{F}\,a$.

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

The system can be labeled with the appropriate acceptance conditions as the state space is explored.
Therefore weak fairness is *trivial* if you have a generalized emptiness check.

Look of $A_{\mathbf{G}\,\mathbf{F}\,en \rightarrow \mathbf{G}\,\mathbf{F}\,oc}$





Note: I don't know of any LTL translator who is able to create the left automaton!

With n fairness hypotheses, the left automaton reaches 3^n states and the right automaton reaches $3^n + 1$ states.

Size of $A_{fairness}$

- Weak fairness: $\bigwedge_{i=1}^{n} \mathbf{G}\,\mathbf{F}(\neg en_i \vee oc_i)$

- Strong fairness: $\bigwedge_{i=1}^{n} \mathbf{G}\,\mathbf{F}\,en_i \rightarrow \mathbf{G}\,\mathbf{F}\,oc_i$

Size of $A_{fairness}$

- Weak fairness: $\bigwedge_{i=1}^{n} \mathbf{G}\,\mathbf{F}(\neg en_i \vee oc_i)$
  - Generalized Büchi automata always 1 state, deterministic.
  - Weak fairness comes for free if you have a generalized emptiness check for generalized Büchi automata.
- Strong fairness: $\bigwedge_{i=1}^{n} \mathbf{G}\,\mathbf{F}\,en_i \rightarrow \mathbf{G}\,\mathbf{F}\,oc_i$

Size of $A_{fairness}$

- Weak fairness: $\bigwedge_{i=1}^{n} \mathbf{G}\,\mathbf{F}(\neg en_i \vee oc_i)$
  - Generalized Büchi automata always 1 state, deterministic.
  - Weak fairness comes for free if you have a generalized emptiness check for generalized Büchi automata.
- Strong fairness: $\bigwedge_{i=1}^{n} \mathbf{G}\,\mathbf{F}\,en_i \rightarrow \mathbf{G}\,\mathbf{F}\,oc_i$
  - Generalized Büchi automata with $3^n + 1$ states, non-deterministic.

Size of $A_{fairness}$

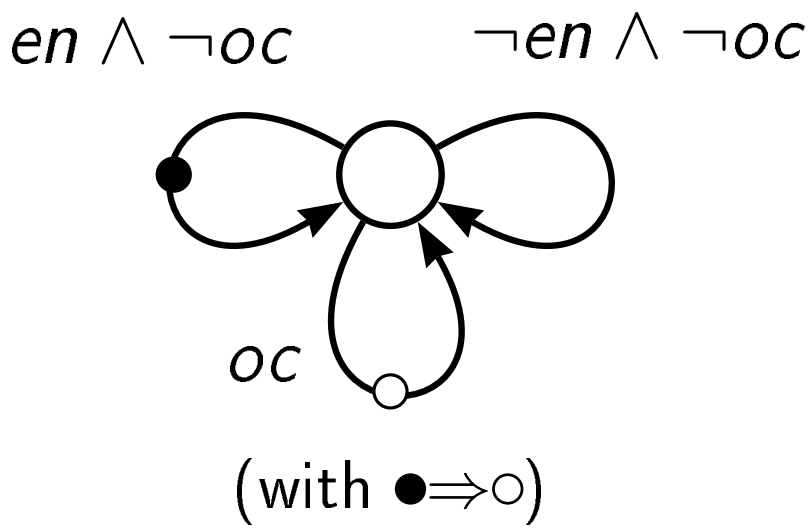
- Weak fairness: $\bigwedge_{i=1}^{n} \mathbf{G}\,\mathbf{F}(\neg en_i \vee oc_i)$
  - Generalized Büchi automata always 1 state, deterministic.
  - Weak fairness comes for free if you have a generalized emptiness check for generalized Büchi automata.
- Strong fairness: $\bigwedge_{i=1}^{n} \mathbf{G}\,\mathbf{F}\,en_i \rightarrow \mathbf{G}\,\mathbf{F}\,oc_i$
  - Generalized Büchi automata with $3^n + 1$ states, non-deterministic.
  - Streett automata always 1 state, deterministic.

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)

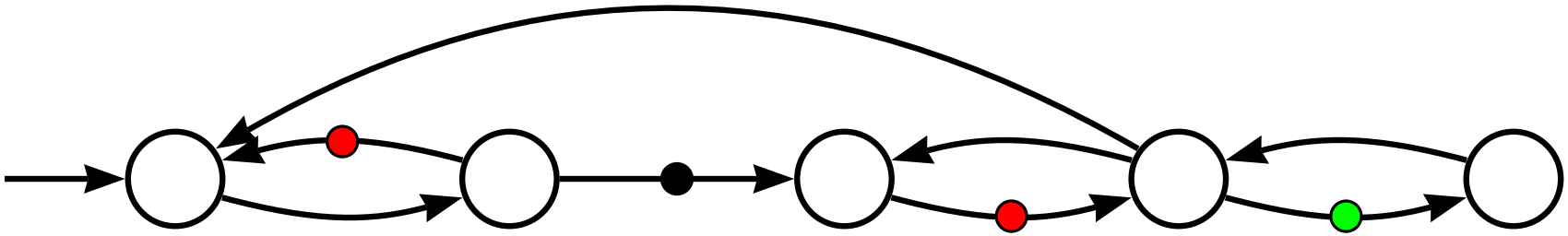
Streett Automata

- 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 $\mathbf{G}\,\mathbf{F}\,en \rightarrow \mathbf{G}\,\mathbf{F}\,oc$:



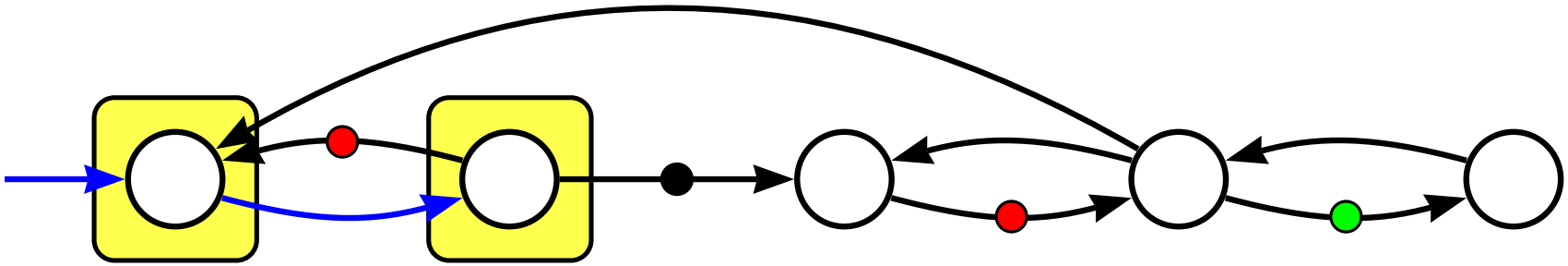
Emptiness Check for Streett Automata

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



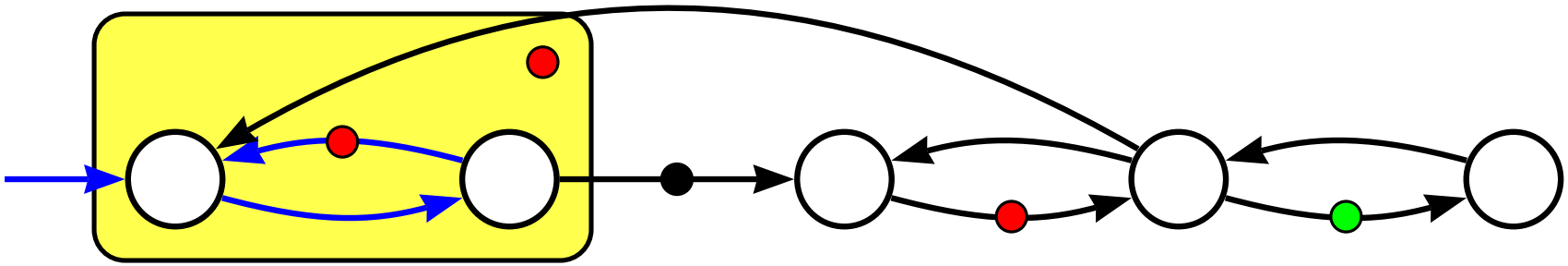
Emptiness Check for Streett Automata

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



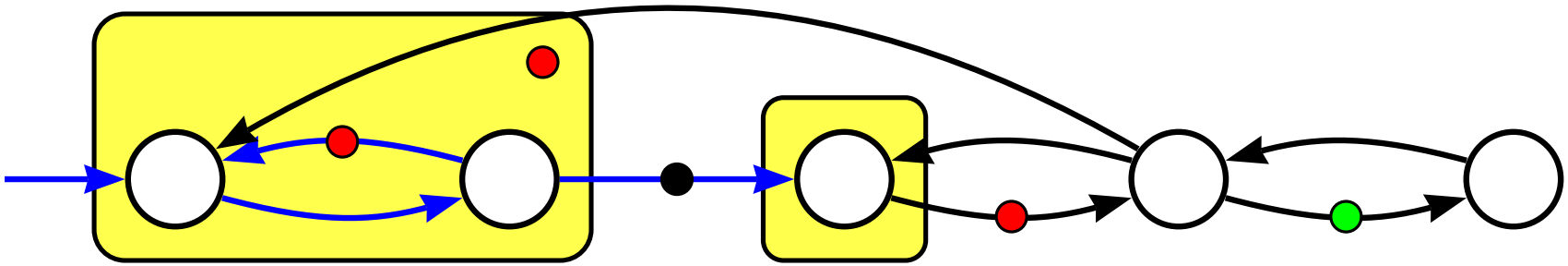
Emptiness Check for Streett Automata

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



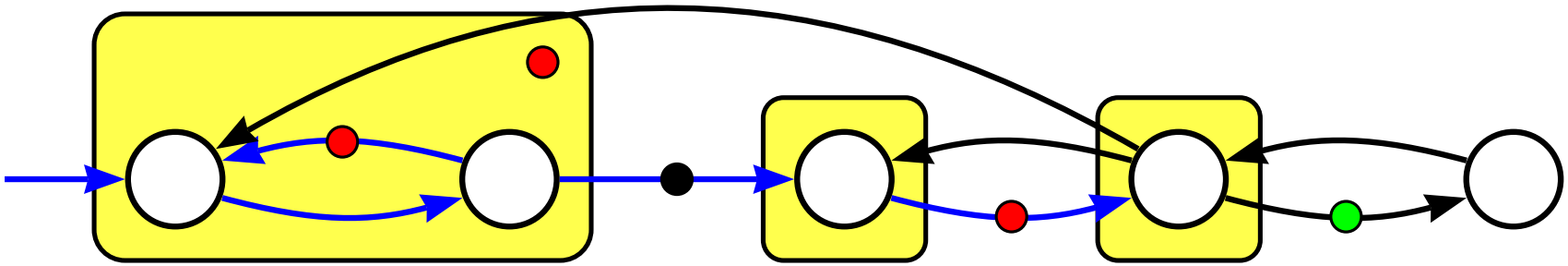
Emptiness Check for Streett Automata

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



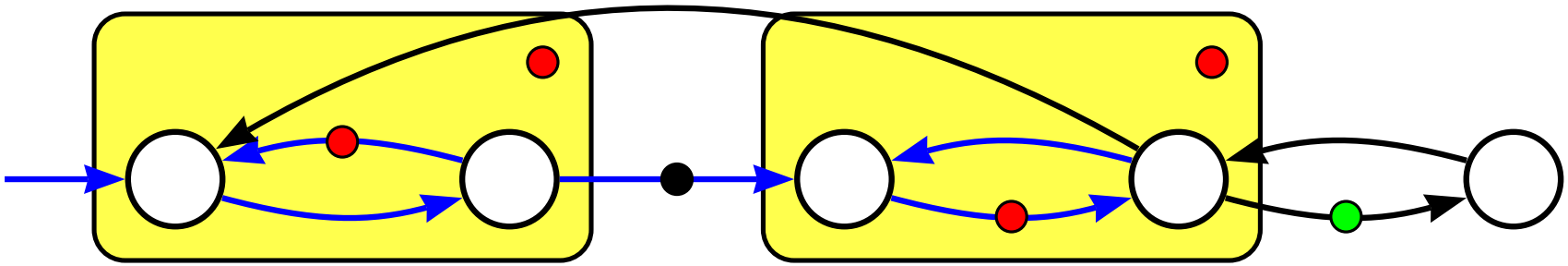
Emptiness Check for Streett Automata

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



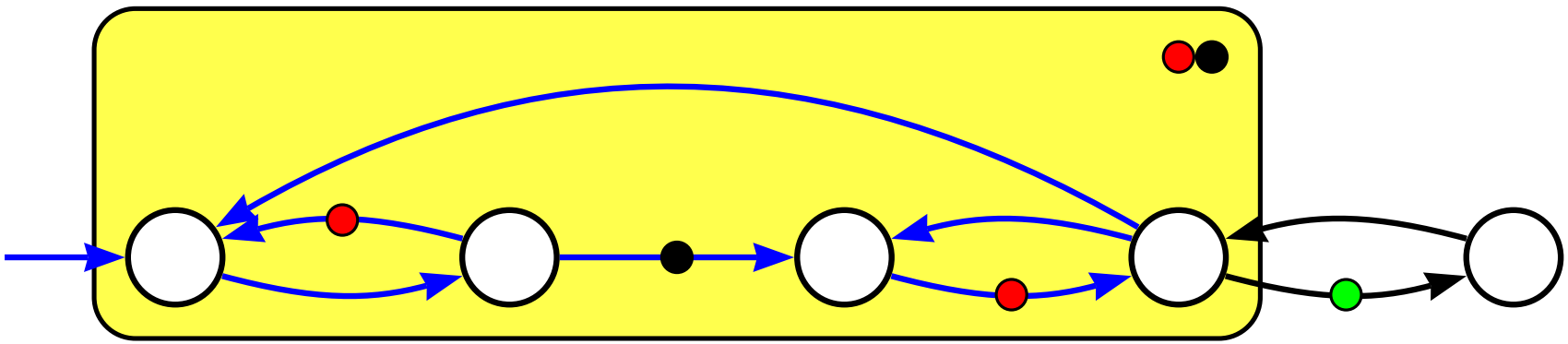
Emptiness Check for Streett Automata

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



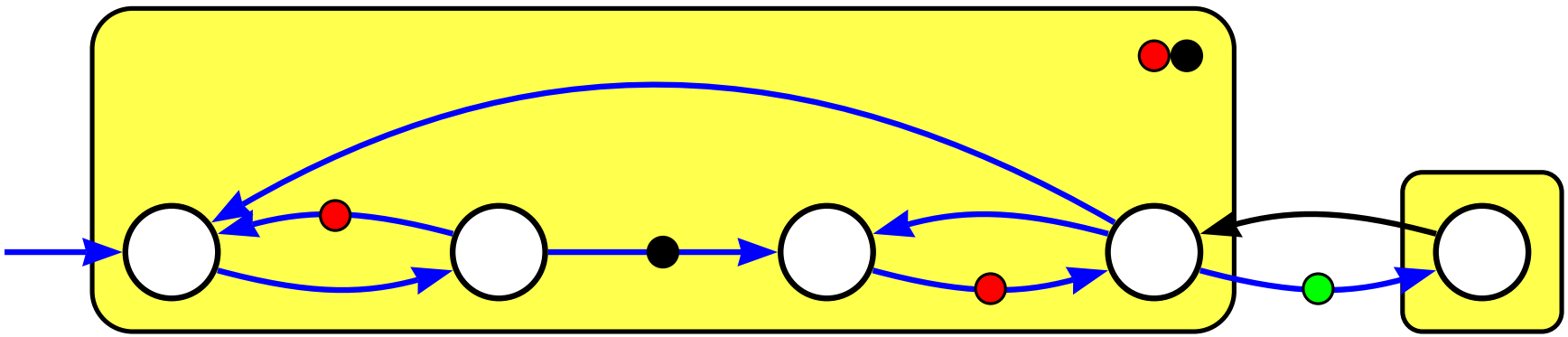
Emptiness Check for Streett Automata

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



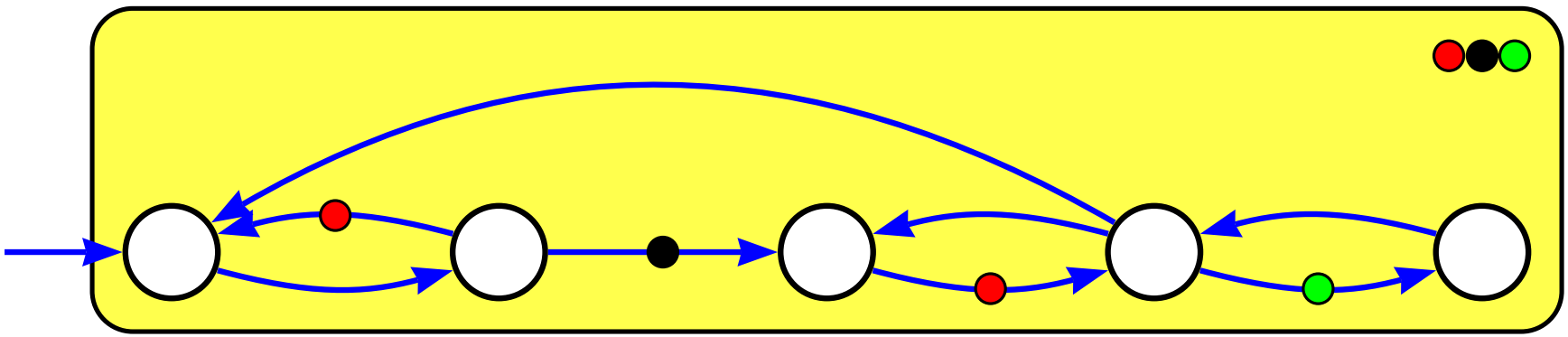
Emptiness Check for Streett Automata

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



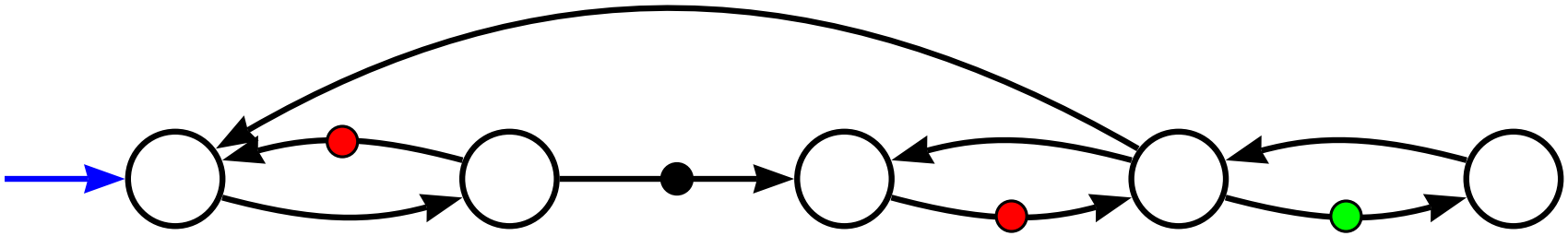
Emptiness Check for Streett Automata

- 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 Check for Streett Automata

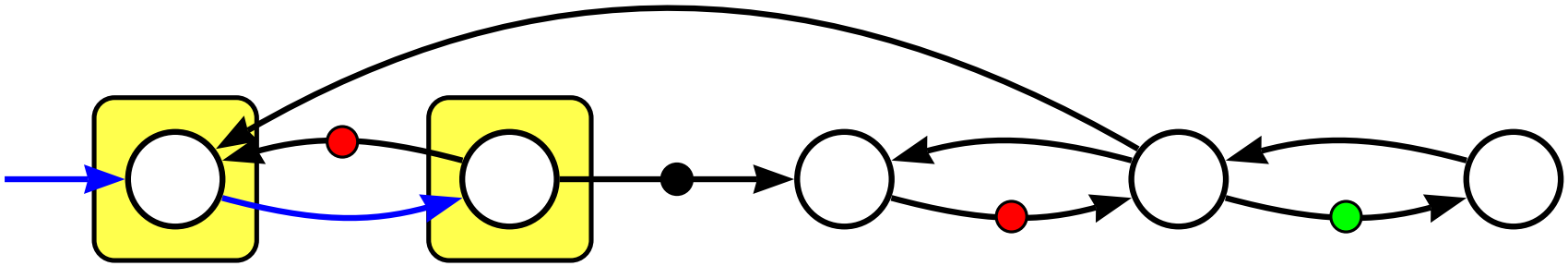
- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



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 Check for Streett Automata

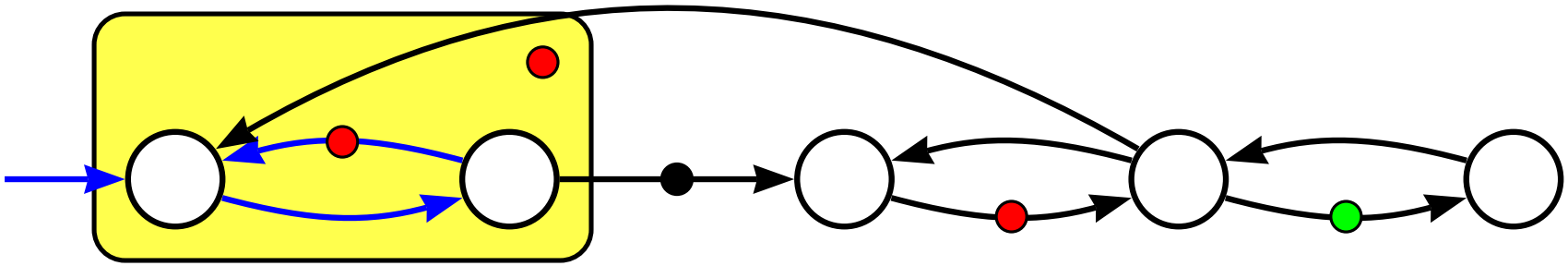
- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



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 Check for Streett Automata

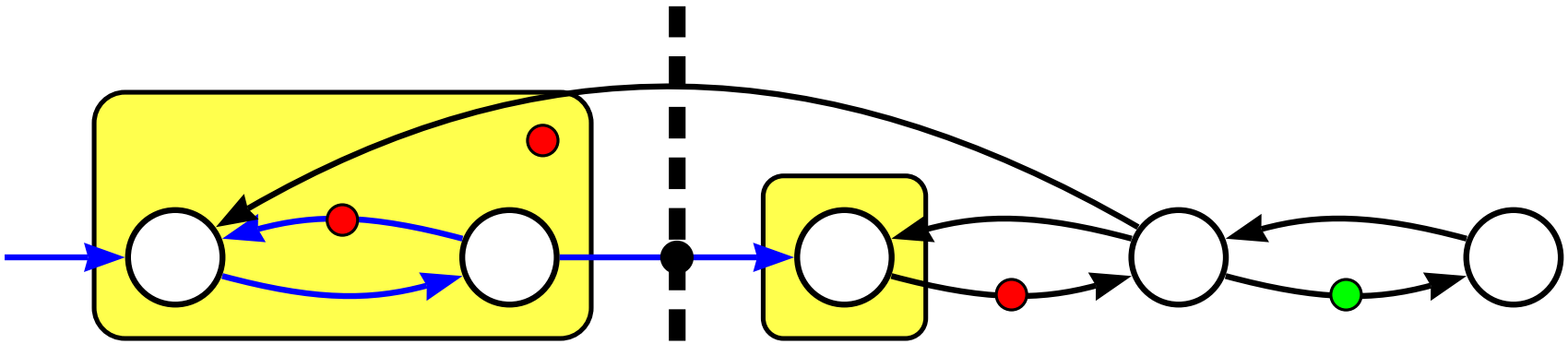
- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



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 Check for Streett Automata

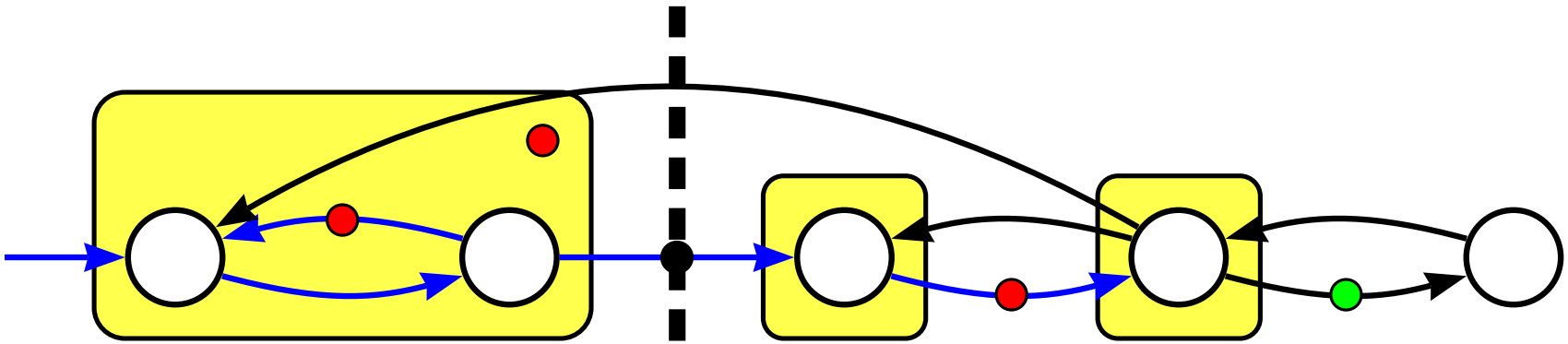
- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



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 Check for Streett Automata

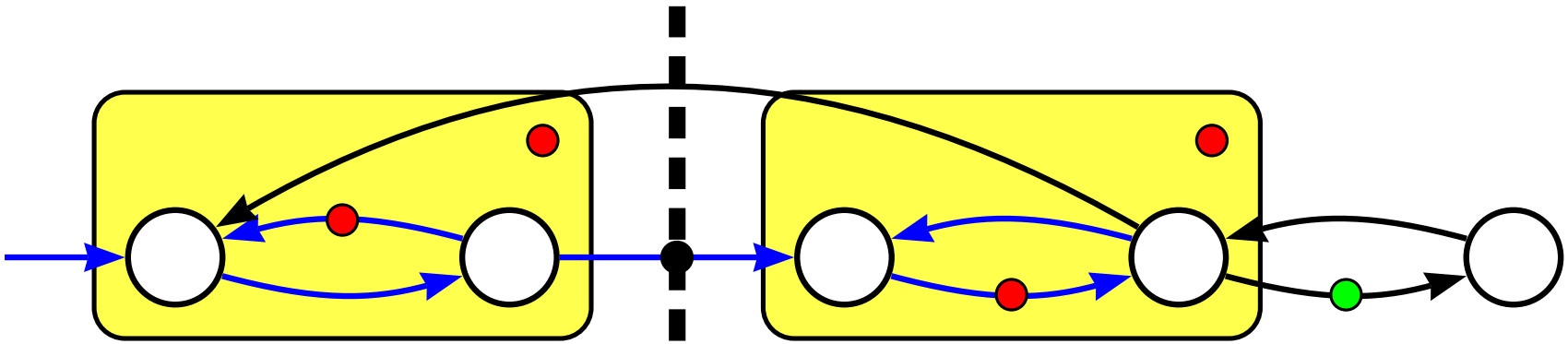
- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



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 Check for Streett Automata

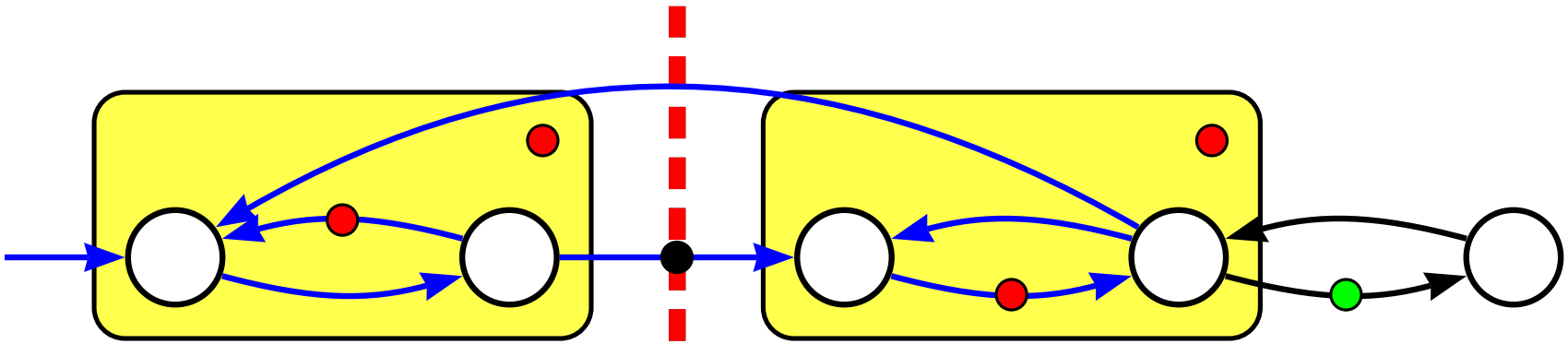
- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



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 Check for Streett Automata

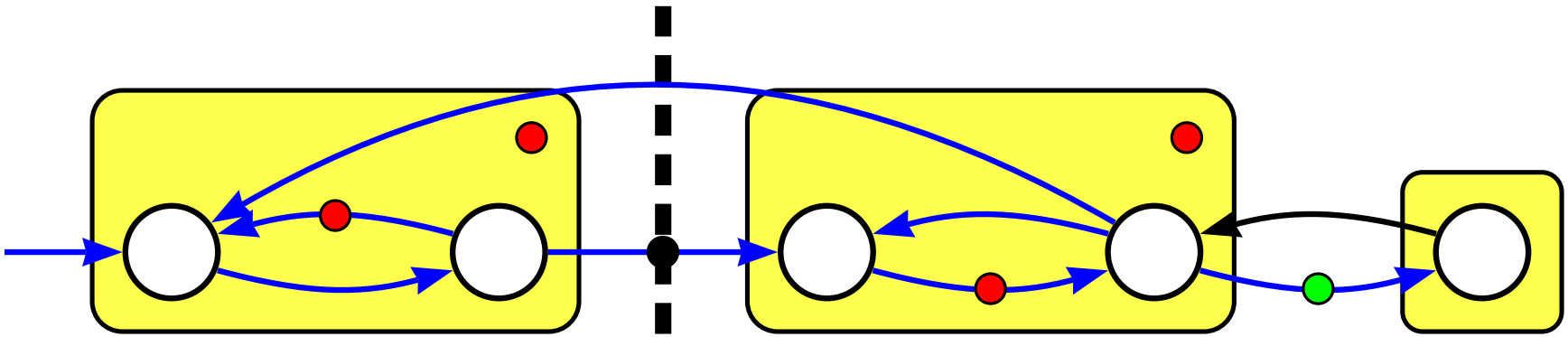
- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



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 Check for Streett Automata

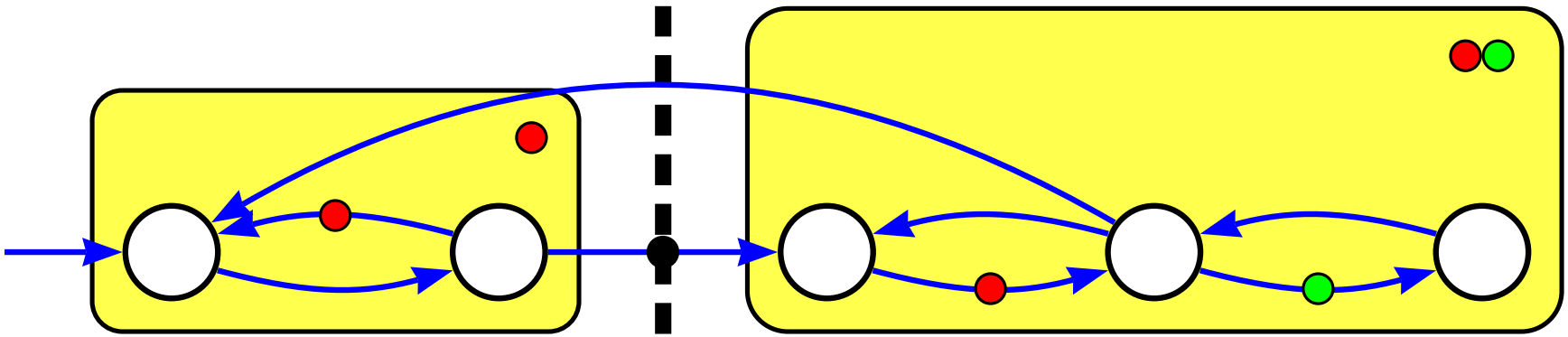
- For Büchi, we look for SCCs whose acceptance conditions verify $\bullet \wedge \circ \wedge \bullet \wedge \bullet$.
- For Streett they must verify something like $\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$.



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 Check for 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 $3^m + 1$ states of the Büchi automaton for the corresponding LTL formula...

This algorithm makes it possible to handle strong fairness with a linear slowdown instead of an exponential slowdown.

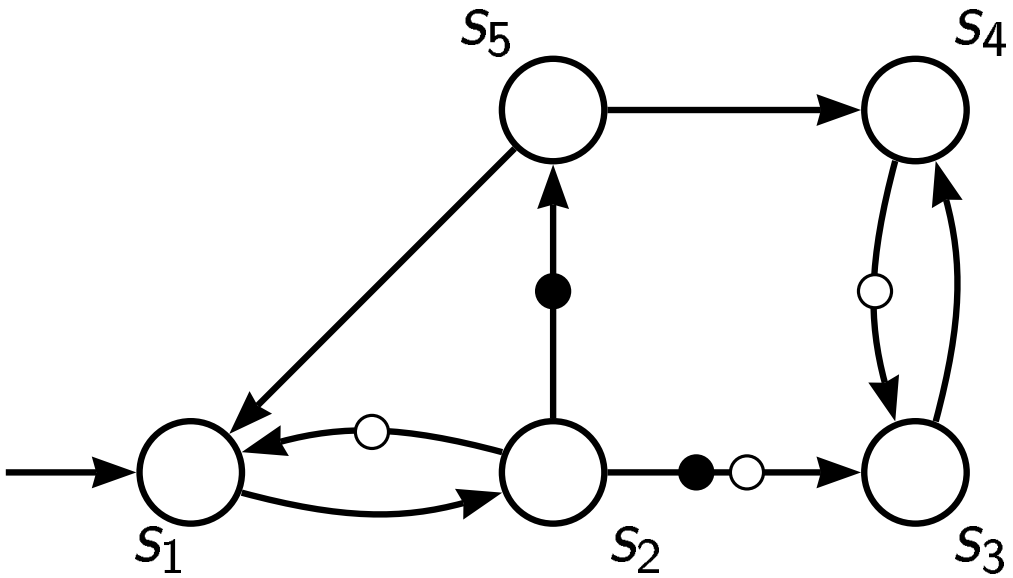
Büchi and Streett automata

- Converting (generalized) Büchi automata into Streett automata?

- Converting Streett automata into (generalized) Büchi automata?

Büchi and Streett automata

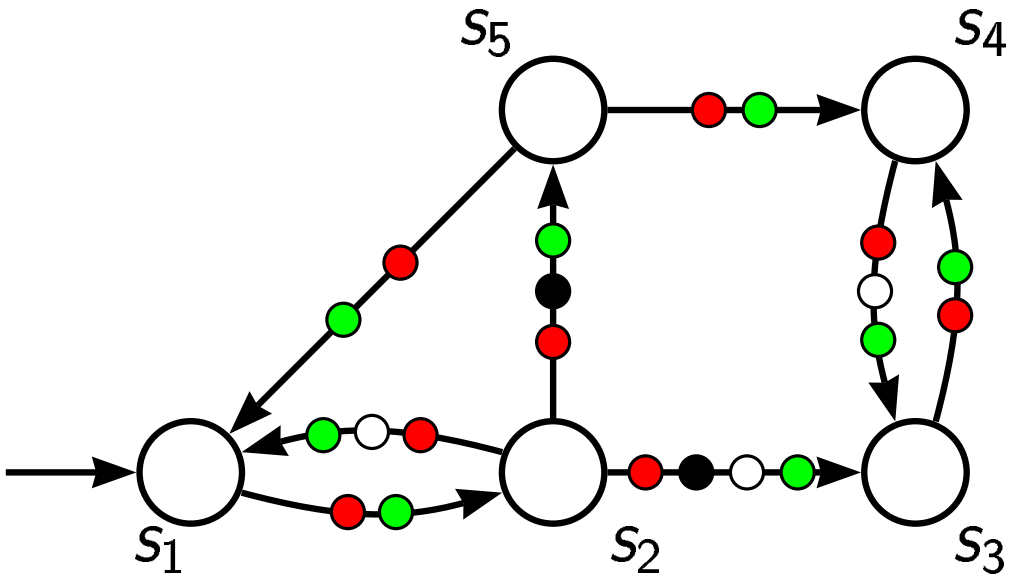
- Converting (generalized) Büchi automata into Streett automata? Easy!



- Converting Streett automata into (generalized) Büchi automata?

Büchi and Streett automata

- Converting (generalized) Büchi automata into Streett automata? Easy!

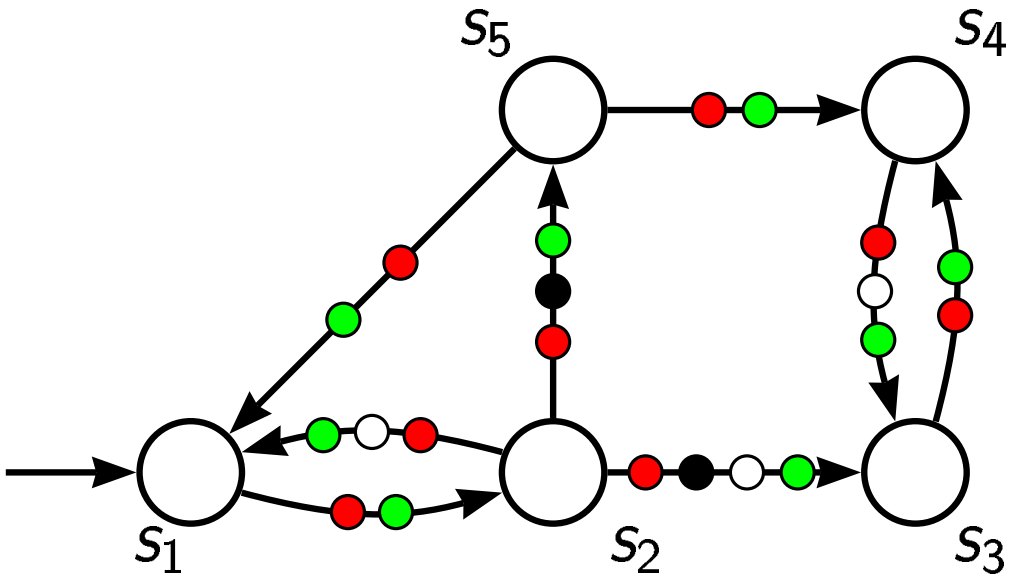


with ● $\Rightarrow$ ● $\wedge$ ● $\Rightarrow$ ○.

- Converting Streett automata into (generalized) Büchi automata?

Büchi and Streett automata

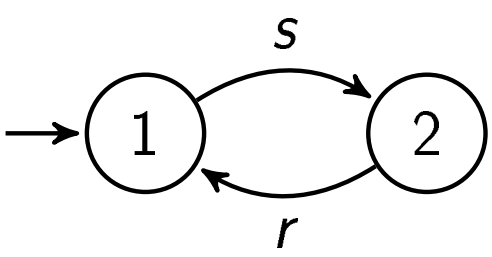
- Converting (generalized) Büchi automata into Streett automata? Easy!



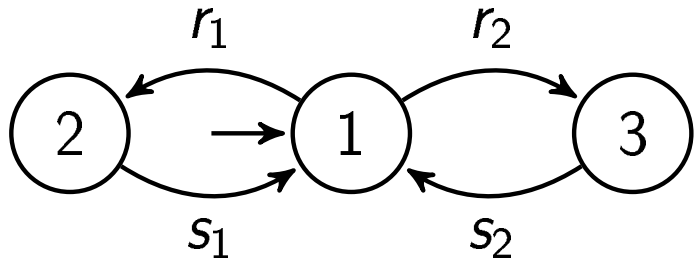
with ● $\Rightarrow$ ● $\wedge$ ● $\Rightarrow$ ○.

- Converting Streett automata into (generalized) Büchi automata? Exponential construction. No polynomial algorithm exist.

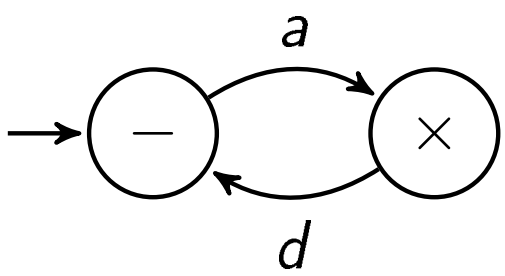
Back to the client/server example



Client C



Server S



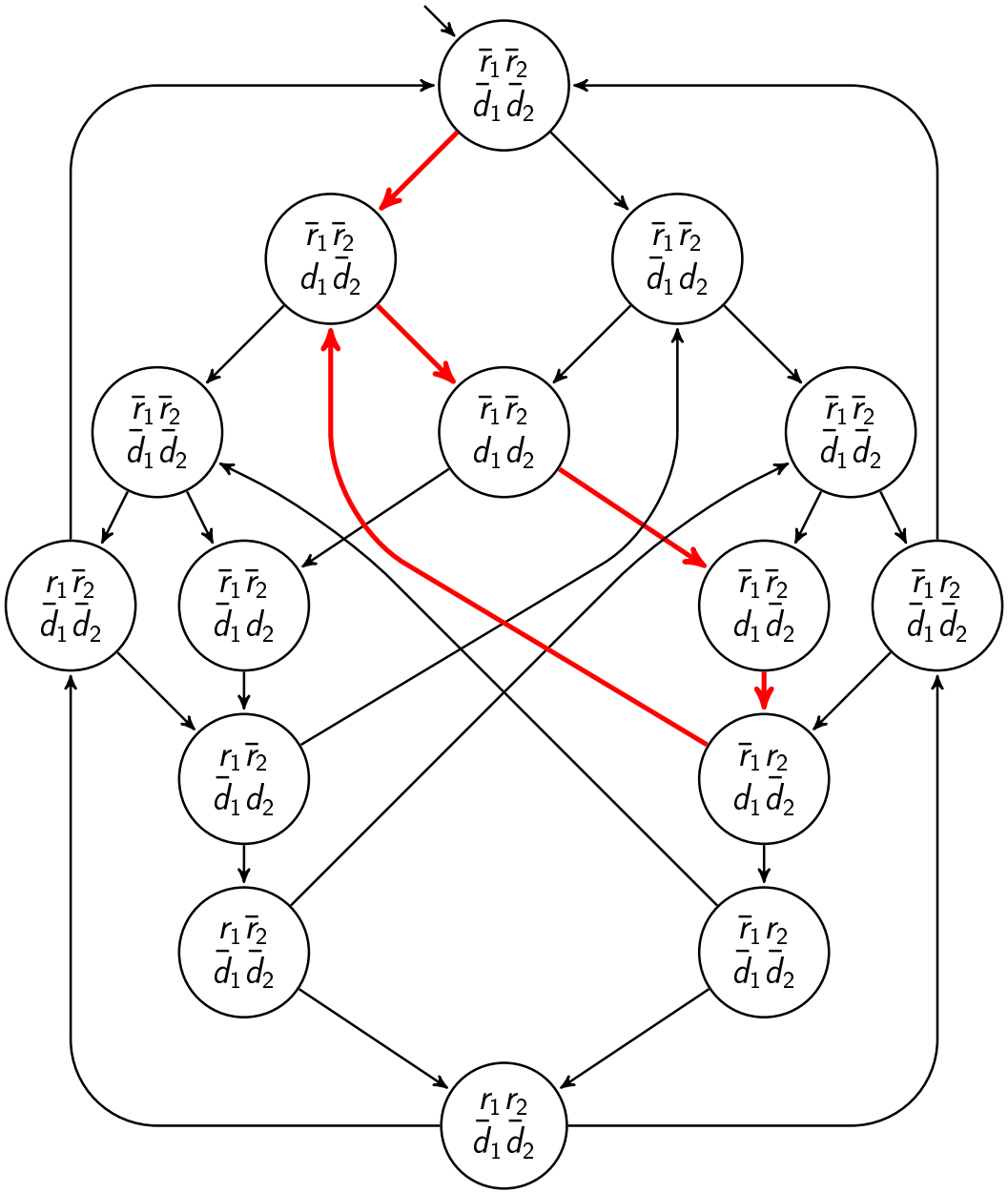
Channel B

Synchronization rules for the system $\langle C, C, S, B, B, B, B \rangle$:

$$
\begin{array}{ll}
(1) & \langle\ s\ ,\ .\ ,\ .\ ,\ .\ ,\ .\ ,\ .\ ,\ a\ ,\ .\ \rangle \\
(2) & \langle\ .\ ,\ s\ ,\ .\ ,\ .\ ,\ .\ ,\ .\ ,\ .\ ,\ a\ \rangle \\
(3) & \langle\ r\ ,\ .\ ,\ .\ ,\ .\ ,\ d\ ,\ .\ ,\ .\ ,\ .\ \rangle \\
(4) & \langle\ .\ ,\ r\ ,\ .\ ,\ .\ ,\ .\ ,\ d\ ,\ .\ ,\ .\ \rangle \\
(5) & \langle\ .\ ,\ .\ ,\ .\ ,\ r_1\ ,\ .\ ,\ .\ ,\ .\ ,\ d\ ,\ .\ \rangle \\
(6) & \langle\ .\ ,\ .\ ,\ .\ ,\ s_1\ ,\ a\ ,\ .\ ,\ .\ ,\ .\ ,\ .\ \rangle \\
(7) & \langle\ .\ ,\ .\ ,\ .\ ,\ r_2\ ,\ .\ ,\ .\ ,\ .\ ,\ .\ ,\ d\ \rangle \\
(8) & \langle\ .\ ,\ .\ ,\ .\ ,\ s_2\ ,\ .\ ,\ a\ ,\ .\ ,\ .\ ,\ .\ \rangle
\end{array}
$$

If a client sends a request, will it eventually get an answer?

(unfair) Kripke structure



How to we make it fair?

$$
\begin{array}{ll}
(1) & \langle\, s, ., ., ., ., ., a, . \,\rangle \\
(2) & \langle\, ., s, ., ., ., ., ., a \,\rangle \\
(3) & \langle\, r, ., ., ., d, ., ., ., . \,\rangle \\
(4) & \langle\, ., r, ., ., ., d, ., ., . \,\rangle \\
(5) & \langle\, ., ., ., r_1, ., ., ., d, . \,\rangle \\
(6) & \langle\, ., ., ., s_1, a, ., ., ., . \,\rangle \\
(7) & \langle\, ., ., ., r_2, ., ., ., ., d \,\rangle \\
(8) & \langle\, ., ., ., s_2, ., ., a, ., ., . \,\rangle
\end{array}
$$

How to we make it fair?

$$
\begin{array}{ll}
(1) & \langle\, s\,,\,.\,,\,.\,,\,.\,,\,.\,,\,a\,,\,.\,\rangle \\
(2) & \langle\,.\,,\,s\,,\,.\,,\,.\,,\,.\,,\,.\,,\,a\,\rangle \\
(3) & \langle\, r\,,\,.\,,\,.\,,\,d\,,\,.\,,\,.\,,\,.\,\rangle \\
(4) & \langle\,.\,,\,r\,,\,.\,,\,.\,,\,d\,,\,.\,,\,.\,\rangle \\
(5) & \langle\,.\,,\,.\,,\,r_1\,,\,.\,,\,.\,,\,d\,,\,.\,\rangle \\
(6) & \langle\,.\,,\,.\,,\,s_1\,,\,a\,,\,.\,,\,.\,,\,.\,\rangle \\
(7) & \langle\,.\,,\,.\,,\,r_2\,,\,.\,,\,.\,,\,.\,,\,d\,\rangle \\
(8) & \langle\,.\,,\,.\,,\,s_2\,,\,.\,,\,a\,,\,.\,,\,.\,\rangle
\end{array}
$$

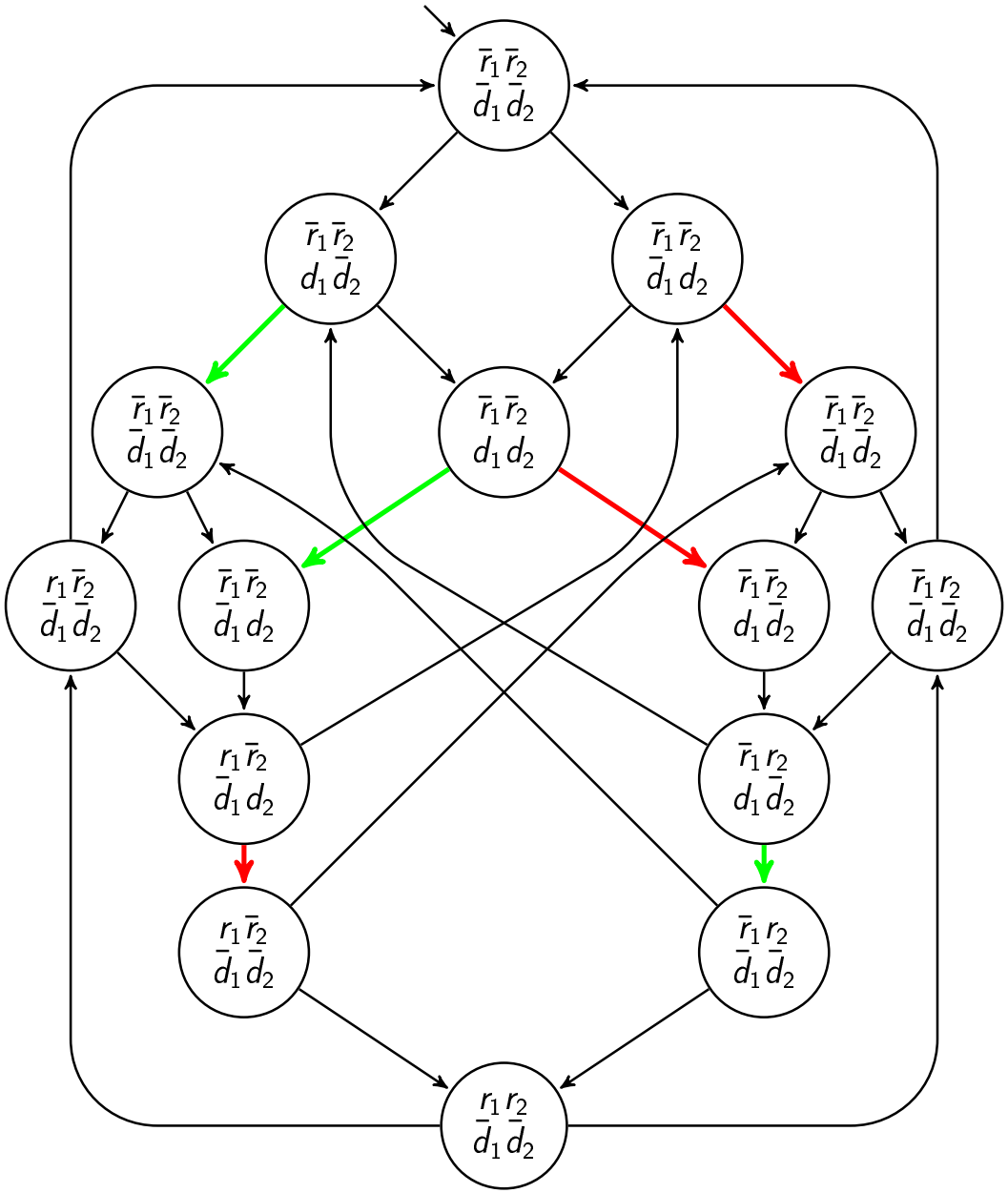
We want the choice between transition (5) and (6) to be fair.
Do we want strong or weak fairness?

How to we make it fair?

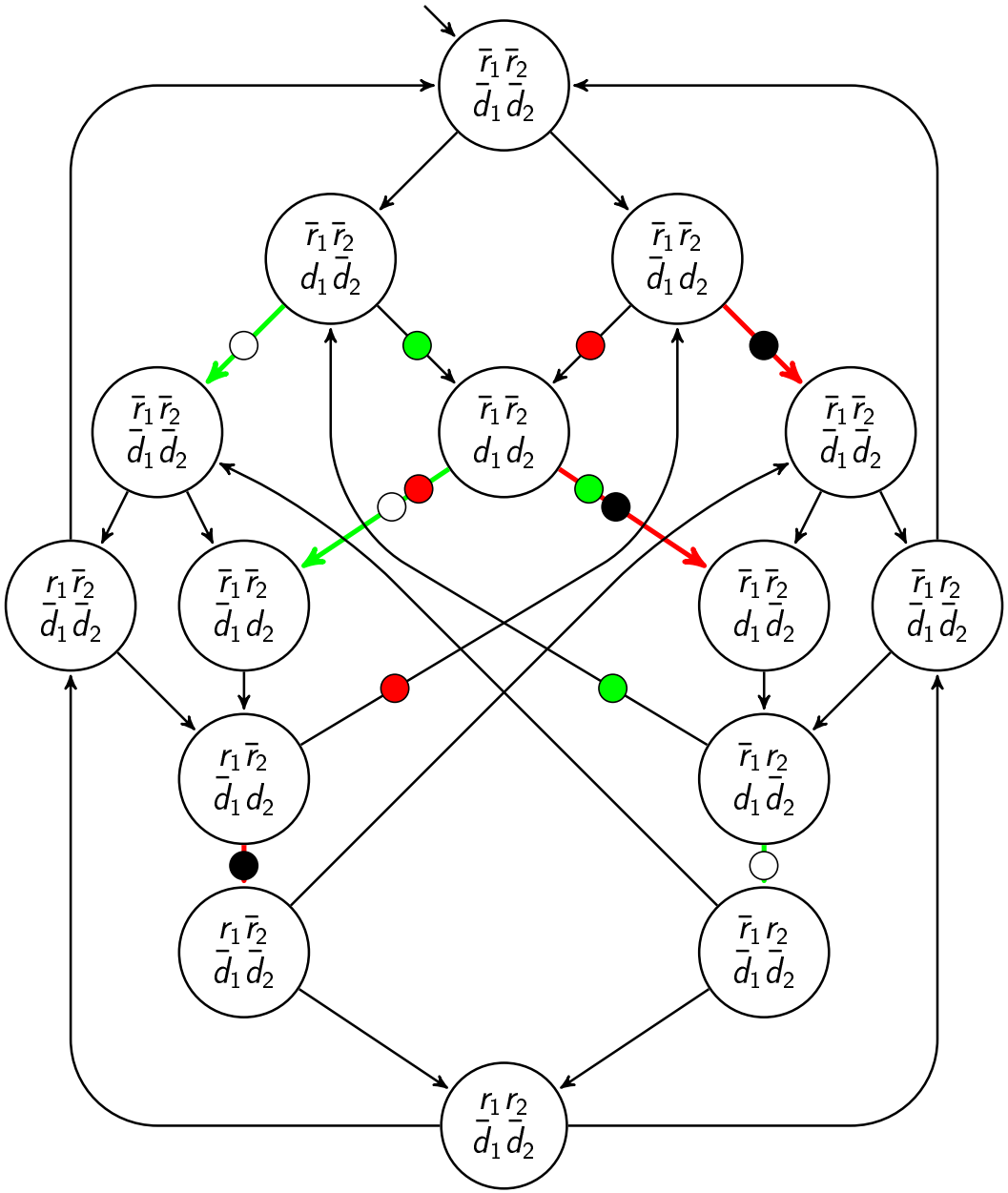
$$
\begin{array}{ll}
(1) & \langle\, s\,,\,.\,,\,.\,,\,.\,,\,.\,,\,a\,,\,.\,\rangle \\
(2) & \langle\, .\,,\,s\,,\,.\,,\,.\,,\,.\,,\,.\,,\,a\,\rangle \\
(3) & \langle\, r\,,\,.\,,\,.\,,\,d\,,\,.\,,\,.\,,\,.\,\rangle \\
(4) & \langle\, .\,,\,r\,,\,.\,,\,.\,,\,d\,,\,.\,,\,.\,\rangle \\
(5) & \langle\, .\,,\,.\,,\,r_1\,,\,.\,,\,.\,,\,d\,,\,.\,\rangle \\
(6) & \langle\, .\,,\,.\,,\,s_1\,,\,a\,,\,.\,,\,.\,,\,.\,\rangle \\
(7) & \langle\, .\,,\,.\,,\,r_2\,,\,.\,,\,.\,,\,.\,,\,d\,\rangle \\
(8) & \langle\, .\,,\,.\,,\,s_2\,,\,.\,,\,a\,,\,.\,,\,.\,\rangle
\end{array}
$$

We want the choice between transition (5) and (6) to be fair.
Do we want strong or weak fairness?
If both (5) and (7) are enabled (= can occur) and we pick (5), then transition (7) will not be enabled until (6) occurs. Therefore if we always pick (5), (7) will not be enabled continuously: it will only be enabled infinitely often.
We need strong fairness!

Making the Kripke Structure fair



Making the Kripke Structure fair ($\bullet \Rightarrow \circ \wedge \bullet \Rightarrow \bullet$)



Another kind of fairness

The previous example still allows scenarios where one client never work. What if we want to disallow this?

$$
\begin{array}{ll}
(1) & \langle\, s\,,\,.\,,\,.\,,\,.\,,\,.\,,\,a\,,\,.\,\rangle \\
(2) & \langle\,.\,,\,s\,,\,.\,,\,.\,,\,.\,,\,.\,,\,a\,\rangle \\
(3) & \langle\, r\,,\,.\,,\,.\,,\,d\,,\,.\,,\,.\,,\,.\,\rangle \\
(4) & \langle\,.\,,\,r\,,\,.\,,\,.\,,\,d\,,\,.\,,\,.\,\rangle \\
(5) & \langle\,.\,,\,.\,,\,r_1\,,\,.\,,\,.\,,\,d\,,\,.\,\rangle \\
(6) & \langle\,.\,,\,.\,,\,s_1\,,\,a\,,\,.\,,\,.\,,\,.\,\rangle \\
(7) & \langle\,.\,,\,.\,,\,r_2\,,\,.\,,\,.\,,\,.\,,\,d\,\rangle \\
(8) & \langle\,.\,,\,.\,,\,s_2\,,\,.\,,\,a\,,\,.\,,\,.\,\rangle
\end{array}
$$

Another kind of fairness

The previous example still allows scenarios where one client never work. What if we want to disallow this?

$$
\begin{array}{ll}
(1) & \langle\, s, ., ., ., ., ., a, . \,\rangle \\
(2) & \langle\, ., s, ., ., ., ., ., a \,\rangle \\
(3) & \langle\, r, ., ., ., d, ., ., ., . \,\rangle \\
(4) & \langle\, ., r, ., ., ., d, ., ., . \,\rangle \\
(5) & \langle\, ., ., ., r_1, ., ., ., d, . \,\rangle \\
(6) & \langle\, ., ., ., s_1, a, ., ., ., . \,\rangle \\
(7) & \langle\, ., ., ., r_2, ., ., ., ., d \,\rangle \\
(8) & \langle\, ., ., ., s_2, ., ., a, ., ., . \,\rangle
\end{array}
$$

We want (1) or (3) to occur infinitely often (i.e., client 1 progresses), and (2) or (3) to occur infinitely often (i.e., client 2 progresses too).

Strong or weak?

Another kind of fairness

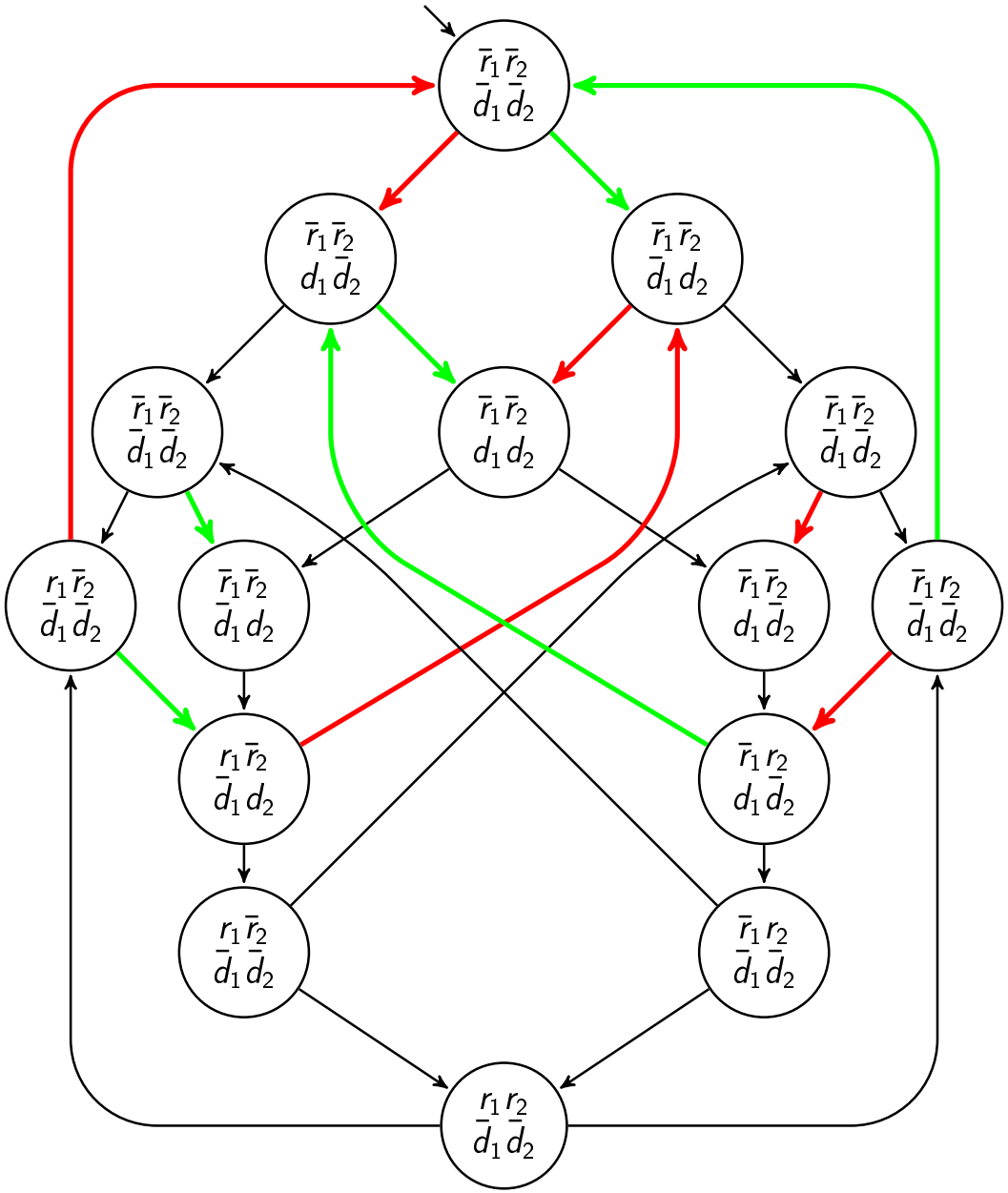
The previous example still allows scenarios where one client never work. What if we want to disallow this?

$$
\begin{array}{ll}
(1) & \langle\, s, ., ., ., ., ., a, . \,\rangle \\
(2) & \langle\, ., s, ., ., ., ., ., a \,\rangle \\
(3) & \langle\, r, ., ., ., d, ., ., ., . \,\rangle \\
(4) & \langle\, ., r, ., ., ., d, ., ., . \,\rangle \\
(5) & \langle\, ., ., ., r_1, ., ., ., d, . \,\rangle \\
(6) & \langle\, ., ., ., s_1, a, ., ., ., . \,\rangle \\
(7) & \langle\, ., ., ., r_2, ., ., ., ., d \,\rangle \\
(8) & \langle\, ., ., ., s_2, ., ., a, ., ., . \,\rangle
\end{array}
$$

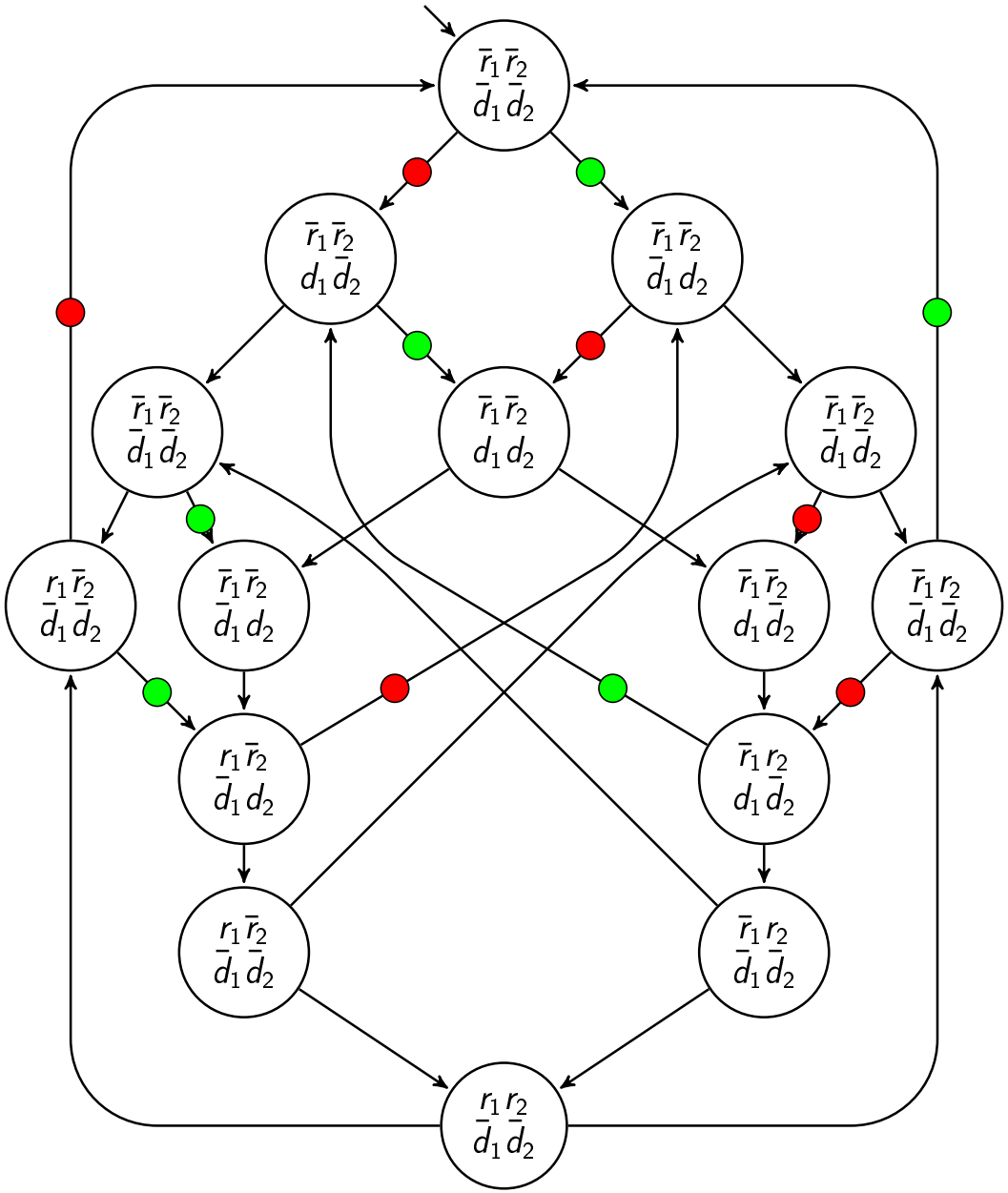
We want (1) or (3) to occur infinitely often (i.e., client 1 progresses), and (2) or (3) to occur infinitely often (i.e., client 2 progresses too).

Strong or weak? Weak. In fact, unconditional fairness.

Making the Kripke Structure fair



Making the Kripke Structure fair (Büchi acc.)



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. 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 in 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 in 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 (w/o acc.cond.)
Therefore $\mathcal{A}_{\neg(\psi\to\varphi)}$ can be represented as a 2-state Streett automaton with 8 pairs of acc.cond.