

# MEMORY REDUCTION FOR STRATEGIES IN INFINITE GAMES

CHRISTOF LÖDING  
RWTH AACHEN, GERMANY

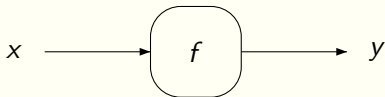
JOINT WORK WITH MICHAEL HOLTMANN

GDV  
SEATTLE, AUGUST 21, 2006

# INTRODUCTION

---

- Some synthesis problems (for circuits/controllers/programs) can be reduced to questions on infinite two player games (or tree automata). [Church'63], [Büchi, Landweber'69], [Rabin'69], [Pnueli, Rosner'89],...



- The object to synthesize corresponds to a winning strategy
- There are two aspects concerning the complexity:
  - How expensive is it to find a solution (strategy)?
  - How big is the solution?

# OUTLINE

---

Infinite games and strategy automata

First approach: Minimizing strategy automata

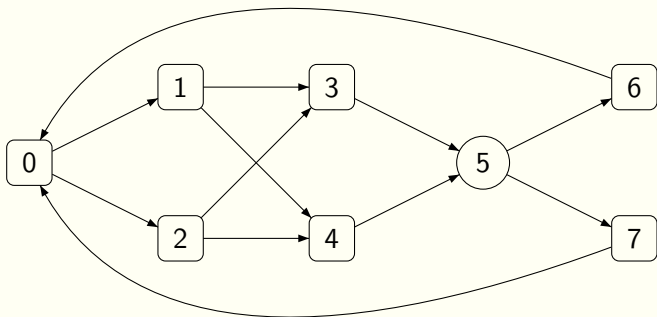
Second approach: Reducing the memory before computing the strategy

# INFINITE GAMES AND STRATEGY AUTOMATA

# INFINITE GAMES ON GRAPHS

- **Game graph:**  $G = (V, V_o, V_{\square}, E)$
- **Game:**  $\mathcal{G} = (G, \Omega)$ , winning condition  $\Omega \subseteq V^{\omega}$  for player  $\bigcirc$
- **Play:** infinite sequence of vertices consistent with edges

Example



Winning condition for player  $\bigcirc$ : avoid sequences 1356, 1457, 2357, 2456

# STRATEGIES AND AUTOMATA

- A **strategy** for player  $\bigcirc$  is a function  $f : V^*V_o \rightarrow V$  that is consistent with the edges. Winning strategy: as usual
- **Strategy automaton**: implementation of a strategy as finite automaton with output (Mealy machine)

$$\mathcal{A} = (S, s_0, \delta, f)$$

with finite state set  $S$ , initial state  $s_0$ ,

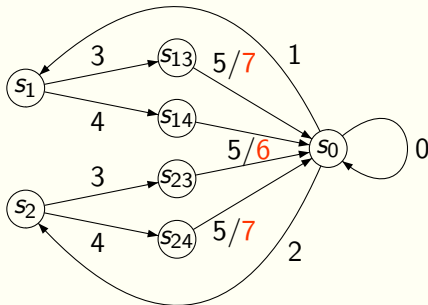
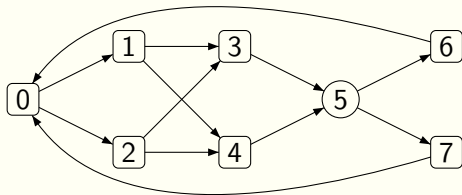
$$\delta : S \times V \rightarrow S \text{ and } f : S \times V_o \rightarrow V$$

**Variant**: automaton reads edges instead of vertices (more robust for our purposes)  $\delta : S \times E \rightarrow S$  and  $f : S \times E \rightarrow V$

- **Positional strategy**: strategy automaton with  $|S| = 1$ .

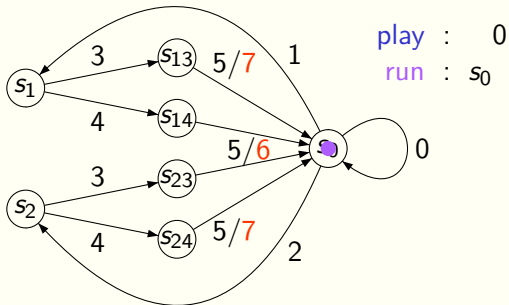
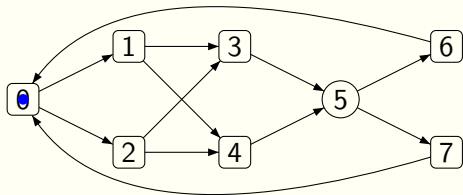
# EXAMPLE (STRATEGY AUTOMATON)

Winning condition for player  $\bigcirc$ : avoid sequences 1356, 1457, 2357, 2456



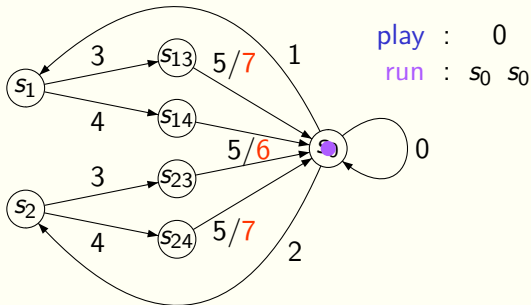
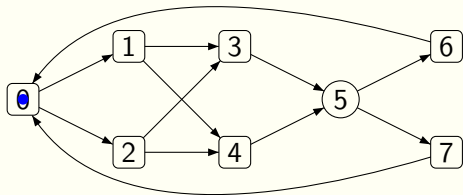
# EXAMPLE (STRATEGY AUTOMATON)

Winning condition for player  $\bigcirc$ : avoid sequences 1356, 1457, 2357, 2456



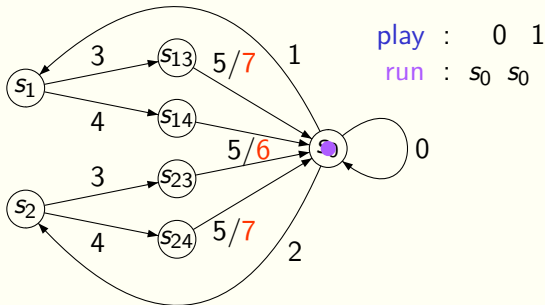
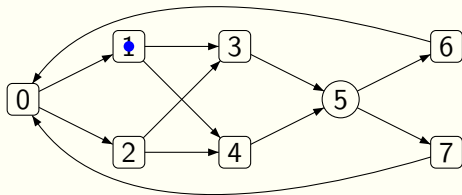
# EXAMPLE (STRATEGY AUTOMATON)

Winning condition for player  $\bigcirc$ : avoid sequences 1356, 1457, 2357, 2456



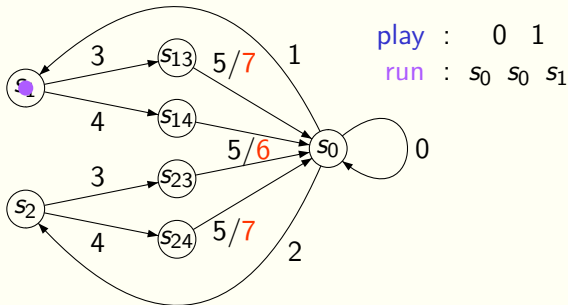
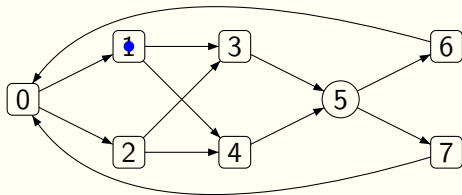
# EXAMPLE (STRATEGY AUTOMATON)

Winning condition for player  $\bigcirc$ : avoid sequences 1356, 1457, 2357, 2456



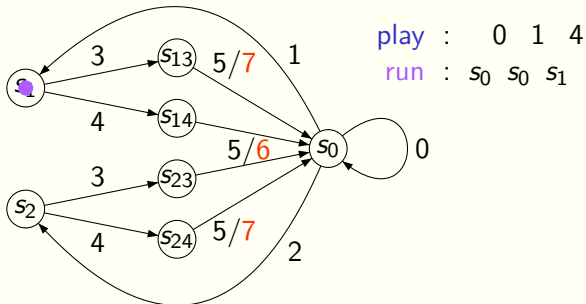
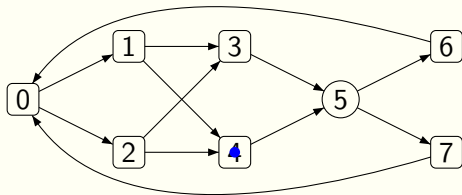
# EXAMPLE (STRATEGY AUTOMATON)

Winning condition for player  $\bigcirc$ : avoid sequences 1356, 1457, 2357, 2456



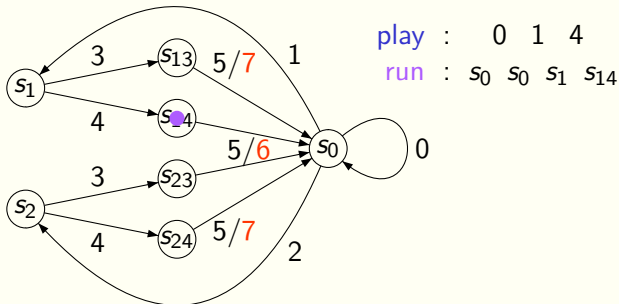
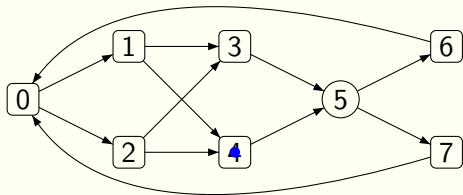
# EXAMPLE (STRATEGY AUTOMATON)

Winning condition for player  $\bigcirc$ : avoid sequences 1356, 1457, 2357, 2456



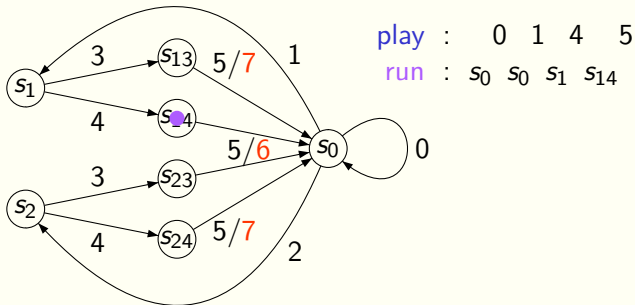
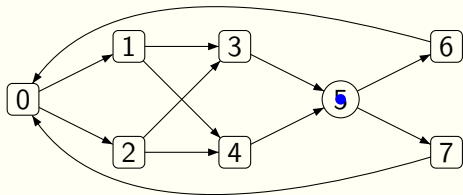
# EXAMPLE (STRATEGY AUTOMATON)

Winning condition for player  $\bigcirc$ : avoid sequences 1356, 1457, 2357, 2456



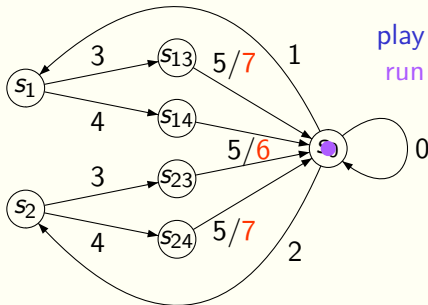
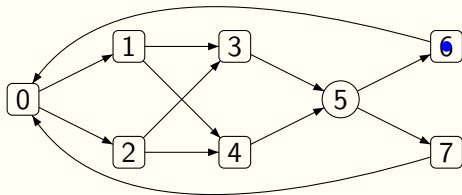
# EXAMPLE (STRATEGY AUTOMATON)

Winning condition for player  $\bigcirc$ : avoid sequences 1356, 1457, 2357, 2456



# EXAMPLE (STRATEGY AUTOMATON)

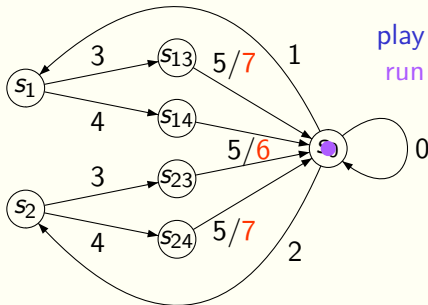
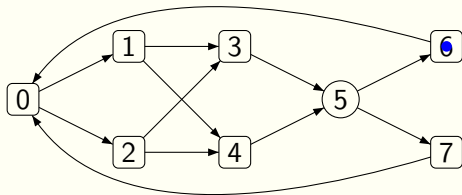
Winning condition for player  $\bigcirc$ : avoid sequences 1356, 1457, 2357, 2456



play : 0 1 4 5 6  
run : s<sub>0</sub> s<sub>0</sub> s<sub>1</sub> s<sub>14</sub> s<sub>0</sub>

# EXAMPLE (STRATEGY AUTOMATON)

Winning condition for player  $\bigcirc$ : avoid sequences 1356, 1457, 2357, 2456



play : 0 1 4 5 6 ...  
run :  $s_0 s_0 s_1 s_{14} s_0 s_0 \dots$

# STRATEGY SYNTHESIS

---

**Given:** Game  $(G, \Omega)$  with (regular) winning condition  $\Omega$

**Problem:** Compute a strategy automaton that implements a winning strategy for Player  $\bigcirc$  (if exists)

## Theorem (Büchi, Landweber'69)

For regular winning conditions one of the players has a finite state winning strategy.

# STRATEGY SYNTHESIS

---

**Given:** Game  $(G, \Omega)$  with (regular) winning condition  $\Omega$

**Problem:** Compute a strategy automaton that implements a winning strategy for Player  $\bigcirc$  (if exists)

## Theorem (Büchi, Landweber'69)

For regular winning conditions one of the players has a finite state winning strategy.

## This talk:

How can we reduce the size of the strategy?

# FIRST APPROACH: MINIMIZING STRATEGY AUTOMATA

# MINIMIZING STRATEGY AUTOMATA

---

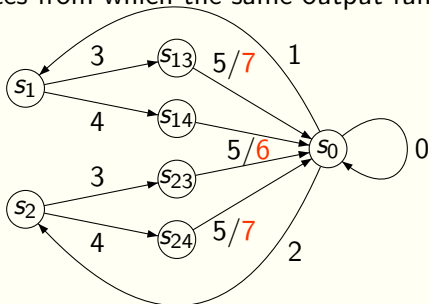
**First approach:** Compute the strategy automaton and then minimize it.

- Strategy automata can be minimized as standard Mealy machines: merge states from which the same output function is computed

# MINIMIZING STRATEGY AUTOMATA

First approach: Compute the strategy automaton and then minimize it.

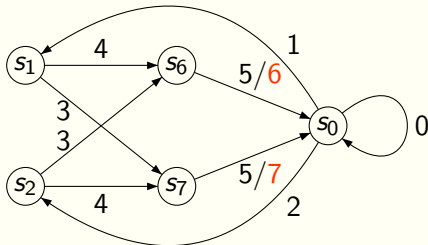
- Strategy automata can be minimized as standard Mealy machines: merge states from which the same output function is computed



# MINIMIZING STRATEGY AUTOMATA

First approach: Compute the strategy automaton and then minimize it.

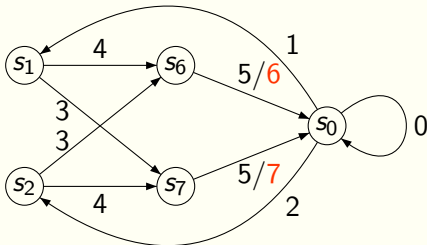
- Strategy automata can be minimized as standard Mealy machines: merge states from which the same output function is computed



# MINIMIZING STRATEGY AUTOMATA

First approach: Compute the strategy automaton and then minimize it.

- Strategy automata can be minimized as standard Mealy machines: merge states from which the same output function is computed



- Advantages:
  - efficient
  - independent of the winning condition

# PROBLEMS WITH FIRST APPROACH

---

Quality of result depends on

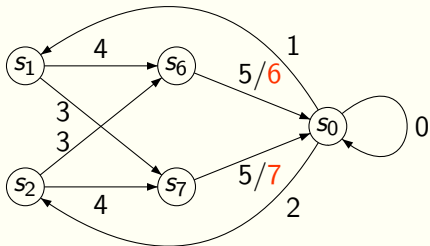
- (a) the definition of the irrelevant transitions
- (b) the strategy

# PROBLEMS WITH FIRST APPROACH

Quality of result depends on

- (a) the definition of the irrelevant transitions
- (b) the strategy

Example for (a)

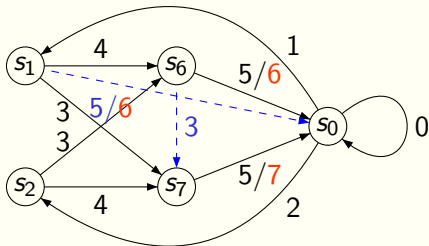


# PROBLEMS WITH FIRST APPROACH

Quality of result depends on

- (a) the definition of the irrelevant transitions
- (b) the strategy

Example for (a)

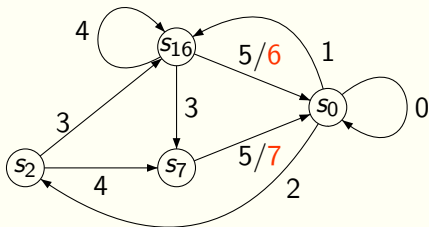


# PROBLEMS WITH FIRST APPROACH

Quality of result depends on

- (a) the definition of the irrelevant transitions
- (b) the strategy

Example for (a)

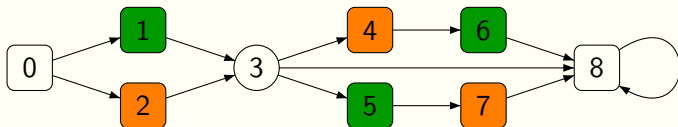


# PROBLEMS WITH FIRST APPROACH

Quality of result depends on

- (a) the definition of the irrelevant transitions
- (b) the strategy

Example for (b):



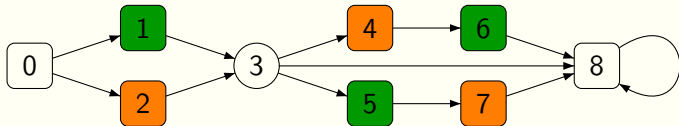
**Winning condition:** visit a green and an orange state

# PROBLEMS WITH FIRST APPROACH

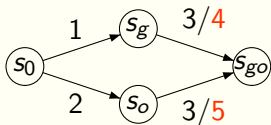
Quality of result depends on

- (a) the definition of the irrelevant transitions
- (b) the strategy

Example for (b):



**Winning condition:** visit a green and an orange state

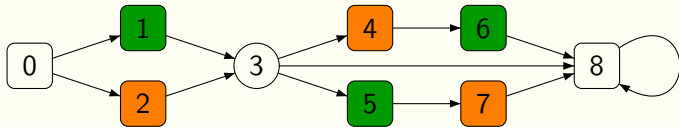


# PROBLEMS WITH FIRST APPROACH

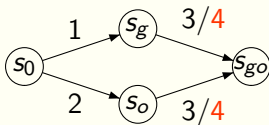
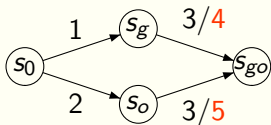
Quality of result depends on

- (a) the definition of the irrelevant transitions
- (b) the strategy

Example for (b):

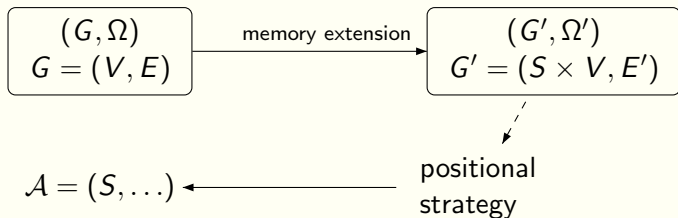


**Winning condition:** visit a green and an orange state

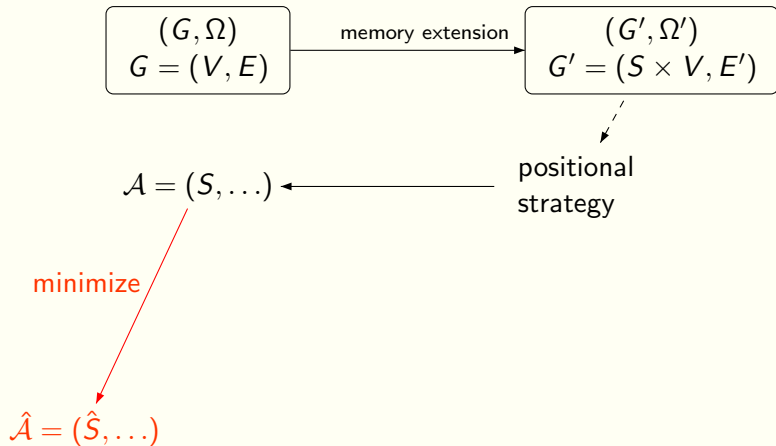


# SECOND APPROACH: REDUCING THE MEMORY BEFORE COMPUTING THE STRATEGY

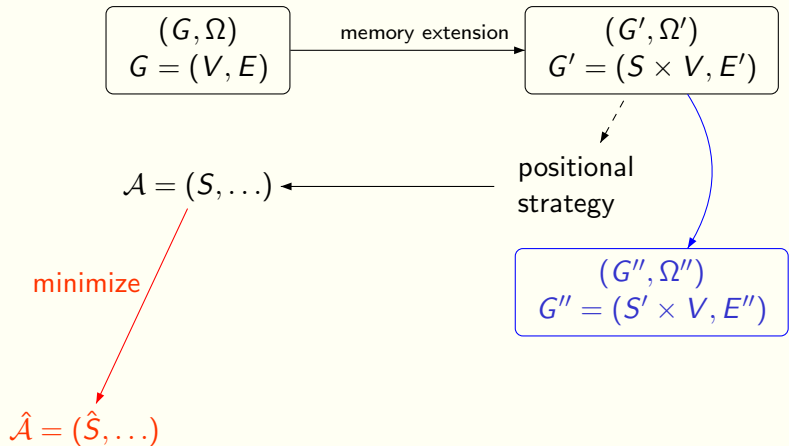
# COMPUTING STRATEGY AUTOMATA



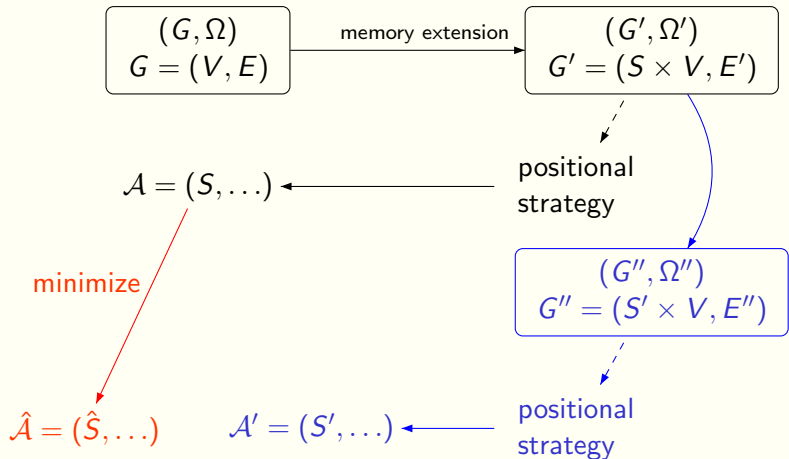
# COMPUTING STRATEGY AUTOMATA



# COMPUTING STRATEGY AUTOMATA



# COMPUTING STRATEGY AUTOMATA



# MEMORY EXTENSIONS OF GAMES

---

$(G', \Omega')$  is a memory extension of  $(G, \Omega)$ , written  $(G, \Omega) \leq (G', \Omega')$  iff

(1)  $V' \subseteq S \times V$

$$V'_o = V' \cap (S \times V_o)$$

$$(s_1, v_1) \rightarrow (s_2, v_2) \in E' \text{ implies } v_1 \rightarrow v_2 \in E$$

# MEMORY EXTENSIONS OF GAMES

---

$(G', \Omega')$  is a memory extension of  $(G, \Omega)$ , written  $(G, \Omega) \leq (G', \Omega')$  iff

(1)  $V' \subseteq S \times V$

$$V'_o = V' \cap (S \times V_o)$$

$(s_1, v_1) \rightarrow (s_2, v_2) \in E'$  implies  $v_1 \rightarrow v_2 \in E$

(2) for all  $(s, v_1) \in V'$  and  $v_1 \rightarrow v_2 \in E$  there is exactly one  $s'$  such that

$$(s, v_1) \rightarrow (s', v_2) \in E'$$

# MEMORY EXTENSIONS OF GAMES

$(G', \Omega')$  is a memory extension of  $(G, \Omega)$ , written  $(G, \Omega) \leq (G', \Omega')$  iff

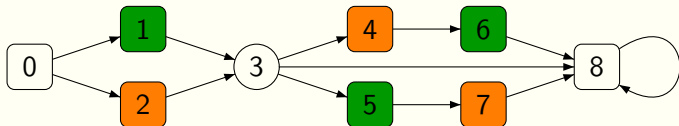
- (1)  $V' \subseteq S \times V$   
 $V'_o = V' \cap (S \times V_o)$   
 $(s_1, v_1) \rightarrow (s_2, v_2) \in E'$  implies  $v_1 \rightarrow v_2 \in E$
- (2) for all  $(s, v_1) \in V'$  and  $v_1 \rightarrow v_2 \in E$  there is exactly one  $s'$  such that  $(s, v_1) \rightarrow (s', v_2) \in E'$
- (3) there is  $s_0 \in S$  such that for each play  $\pi' = (s_0, v_0)(s_1, v_1) \cdots$  in  $G'$  the corresponding play  $\pi = v_0 v_1 \cdots$  is in  $\Omega$  iff  $\pi'$  is in  $\Omega'$

# MEMORY EXTENSIONS OF GAMES

$(G', \Omega')$  is a memory extension of  $(G, \Omega)$ , written  $(G, \Omega) \leq (G', \Omega')$  iff

- (1)  $V' \subseteq S \times V$   
 $V'_o = V' \cap (S \times V_o)$   
 $(s_1, v_1) \rightarrow (s_2, v_2) \in E'$  implies  $v_1 \rightarrow v_2 \in E$
- (2) for all  $(s, v_1) \in V'$  and  $v_1 \rightarrow v_2 \in E$  there is exactly one  $s'$  such that  $(s, v_1) \rightarrow (s', v_2) \in E'$
- (3) there is  $s_0 \in S$  such that for each play  $\pi' = (s_0, v_0)(s_1, v_1) \cdots$  in  $G'$  the corresponding play  $\pi = v_0 v_1 \cdots$  is in  $\Omega$  iff  $\pi'$  is in  $\Omega'$

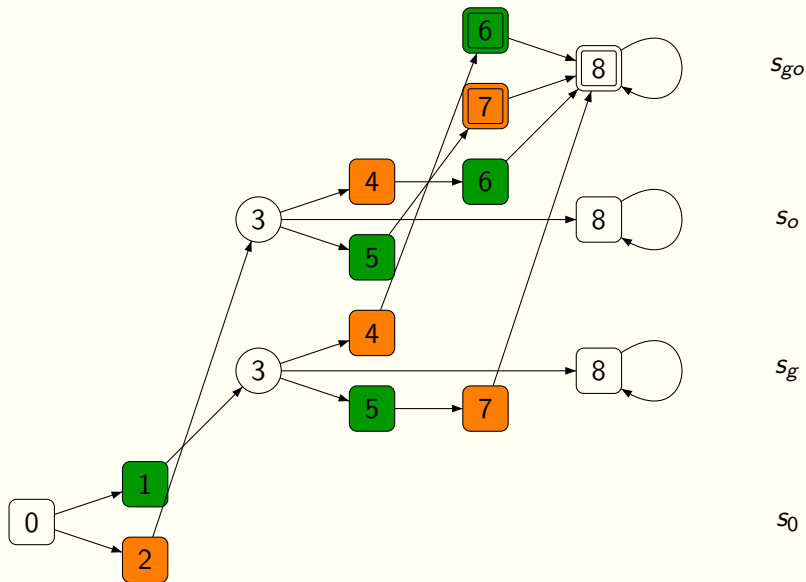
## Example



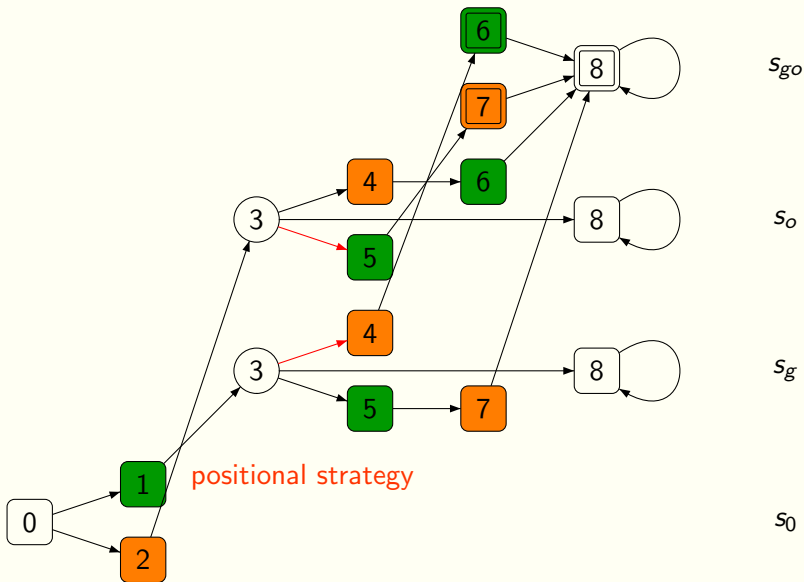
**Winning condition:** visit a green and an orange state

Use memory  $S = \{s_0, s_g, s_o, s_{go}\}$

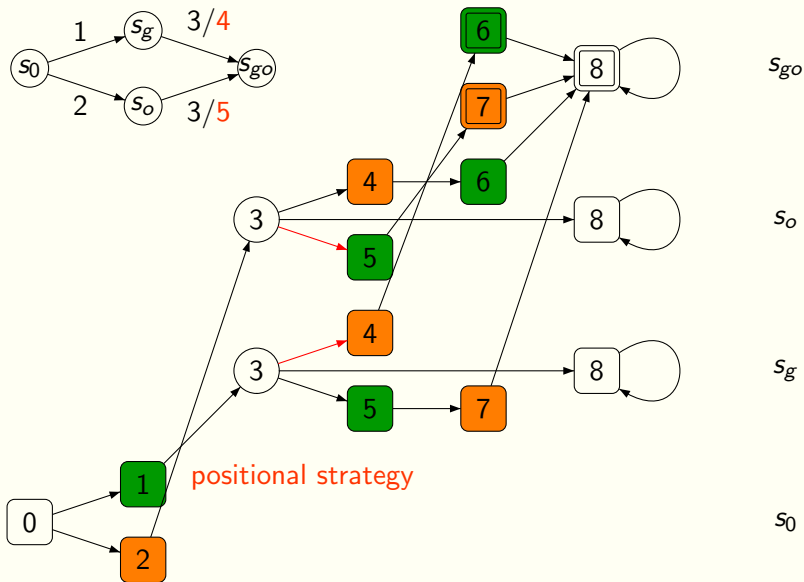
# EXAMPLE – MEMORY EXTENSION



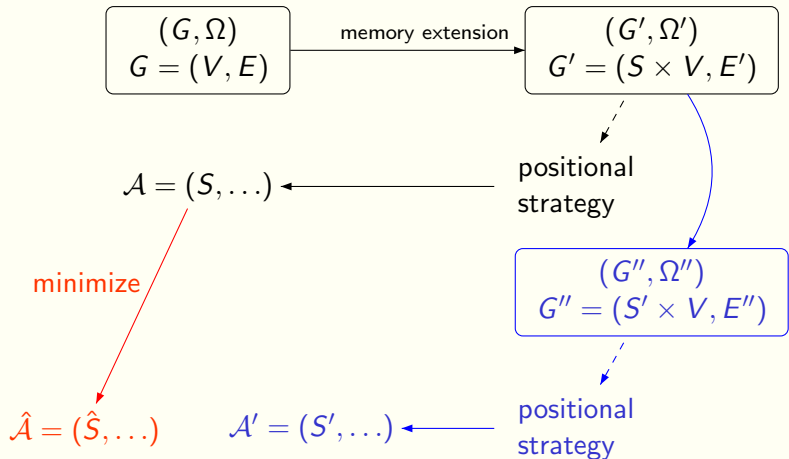
# EXAMPLE – MEMORY EXTENSION



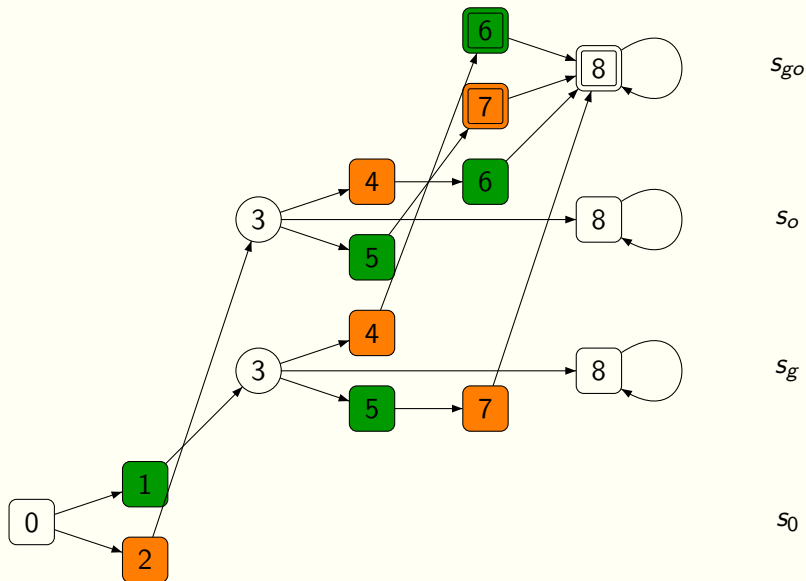
# EXAMPLE – MEMORY EXTENSION



# OVERVIEW

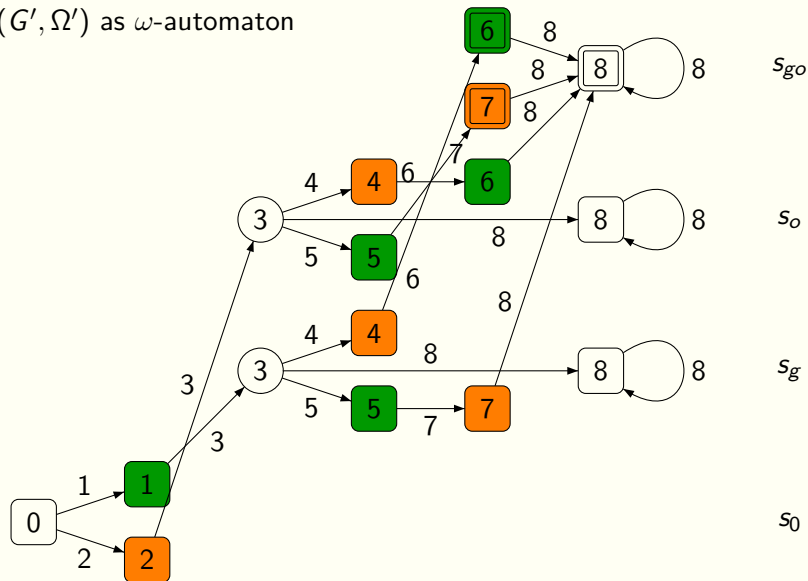


# EXAMPLE – MEMORY REDUCTION



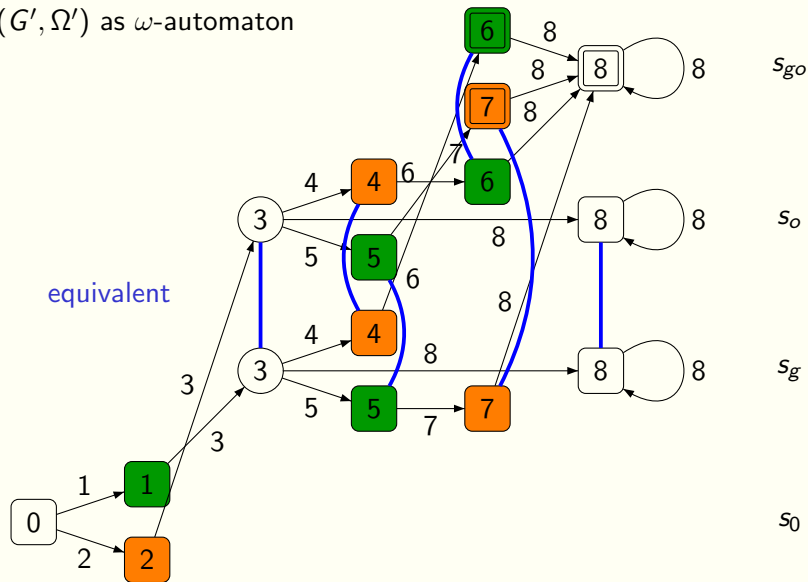
# EXAMPLE – MEMORY REDUCTION

$(G', \Omega')$  as  $\omega$ -automaton



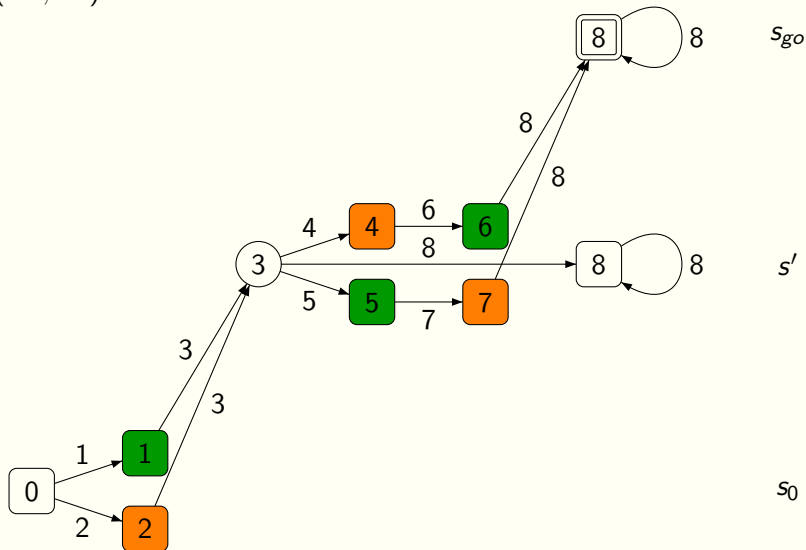
# EXAMPLE – MEMORY REDUCTION

$(G', \Omega')$  as  $\omega$ -automaton



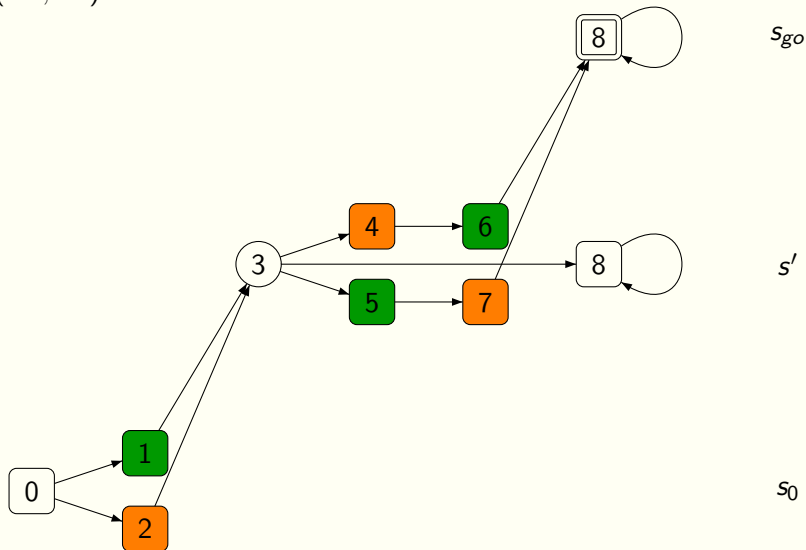
# EXAMPLE – MEMORY REDUCTION

$(G'', \Omega'')$  as  $\omega$ -automaton



# EXAMPLE – MEMORY REDUCTION

$(G'', \Omega'')$



# GENERAL METHOD

---

**Input:**  $(G, \Omega)$ ,  $G = (V, E)$

(1) Compute memory extension  $(G', \Omega')$ ,  $G' = (S \times V, E')$

(2) View  $(G', \Omega')$  as det.  $\omega$ -automaton reading plays of  $G$ :

$$(s_1, v_1) \xrightarrow{v_2} (s_2, v_2)$$

(3) Reduce  $(G', \Omega')$  as  $\omega$ -automaton, obtain  $(G'', \Omega'')$

(4) Compute strategy automaton from a positional strategy in  $(G'', \Omega'')$

**Input:**  $(G, \Omega)$ ,  $G = (V, E)$

- (1) Compute memory extension  $(G', \Omega')$ ,  $G' = (S \times V, E')$
- (2) View  $(G', \Omega')$  as det.  $\omega$ -automaton reading plays of  $G$ :

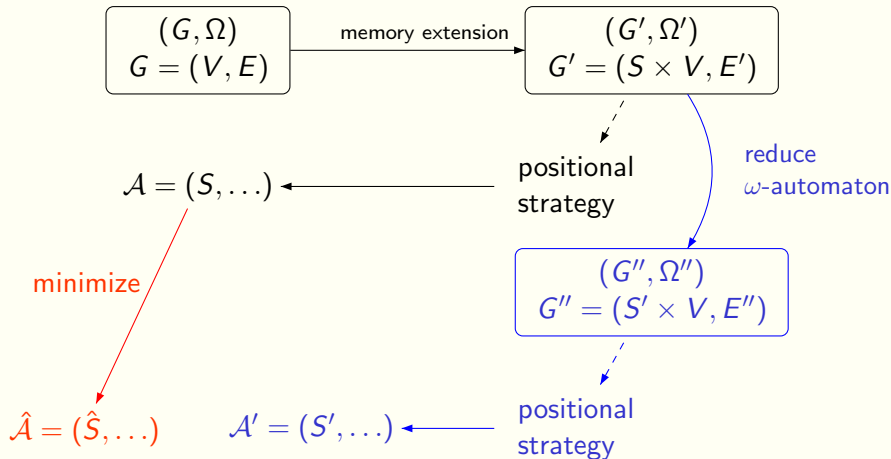
$$(s_1, v_1) \xrightarrow{v_2} (s_2, v_2)$$

- (3) Reduce  $(G', \Omega')$  as  $\omega$ -automaton, obtain  $(G'', \Omega'')$
- (4) Compute strategy automaton from a positional strategy in  $(G'', \Omega'')$

## Proposition

Let  $(G, \Omega)$ ,  $(G', \Omega')$  be games with  $(G, \Omega) \leq (G', \Omega')$ . If  $(G'', \Omega'')$  is equivalent to  $(G', \Omega')$  as  $\omega$ -automaton (and respects some simple structural properties), then  $(G, \Omega) \leq (G'', \Omega'')$ .

# OVERVIEW



# MINIMIZATION OF DET. $\omega$ -AUTOMATA

---

- Exact minimization of det.  $\omega$ -automata is a difficult problem, already for Büchi automata
- Possible for the class of weak Büchi automata
- Heuristics developed for nondet. Büchi automata

# WEAK BÜCHI CONDITION

---

- A deterministic Büchi automaton (DBA) is called weak if each of the strongly connected components of its transition graph completely consists of accepting states or completely consists of non-accepting states
- Capture Boolean combinations of safety and reachability conditions

## Theorem (Staiger'83)

For each weak DBA there is a minimal equivalent weak DBA with a unique transition structure.

# WEAK BÜCHI CONDITION

---

- A deterministic Büchi automaton (DBA) is called weak if each of the strongly connected components of its transition graph completely consists of accepting states or completely consists of non-accepting states
- Capture Boolean combinations of safety and reachability conditions

## Theorem (Staiger'83)

For each weak DBA there is a minimal equivalent weak DBA with a unique transition structure.

## Theorem (L.'01)

The problem of minimizing weak DBA can be reduced in linear time to the problem of minimizing standard finite automata.

# CONSEQUENCE FOR MEMORY REDUCTION

---

## Theorem

Let  $(G, \Omega) \leq (G', \Omega')$  for a weak Büchi game  $(G', \Omega')$ .

One can compute in time  $\mathcal{O}(|V|^2 \cdot |S| \log(|V| \cdot |S|))$  a minimal weak Büchi game  $(G'', \Omega'')$  with  $(G, \Omega) \leq (G'', \Omega'')$ .

(Minimal meaning that the size of the memory component of  $G''$  is minimal.)

# CONSEQUENCE FOR MEMORY REDUCTION

---

## Theorem

Let  $(G, \Omega) \leq (G', \Omega')$  for a weak Büchi game  $(G', \Omega')$ .

One can compute in time  $\mathcal{O}(|V|^2 \cdot |S| \log(|V| \cdot |S|))$  a minimal weak Büchi game  $(G'', \Omega'')$  with  $(G, \Omega) \leq (G'', \Omega'')$ .

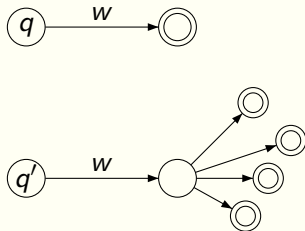
(Minimal meaning that the size of the memory component of  $G''$  is minimal.)

## Remark

This does **not** mean that  $G''$  provides the optimal amount of memory required for a winning strategy of player  $\bigcirc$ !

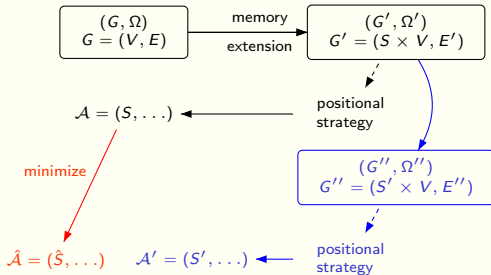
# BÜCHI CONDITION

- For Büchi automata we can use the “**delayed simulation quotient**” [Etessami, Wilke, Schuller'05]



- Successfully applied in translation of LTL into (nondet.) Büchi automata
- In our setting (det. automata) computable in time  $\mathcal{O}(|V|^2 \cdot |S| \log(|V| \cdot |S|))$

# COMPARISON OF THE TWO APPROACHES

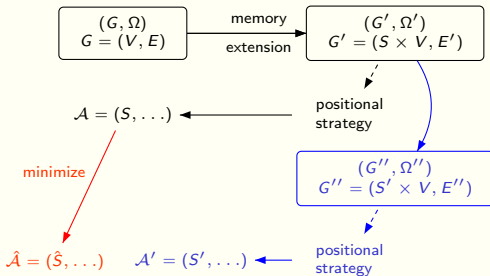


# COMPARISON OF THE TWO APPROACHES

- (1) **Minimizing strategy automata:** Minimize strategy automaton as Mealy machine

**Advantage:** efficient and independent of winning condition

**Drawback:** depends on the strategy

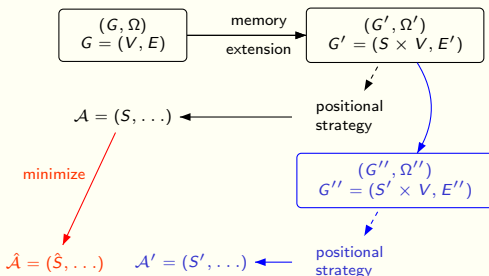


# COMPARISON OF THE TWO APPROACHES

- (1) **Minimizing strategy automata:** Minimize strategy automaton as Mealy machine

**Advantage:** efficient and independent of winning condition

**Drawback:** depends on the strategy



- (2) **Reducing memory extensions:** Apply techniques for reducing  $\omega$ -automata to shrink the memory extension of the given game

**Advantage:** does not depend on strategy

**Drawback:** only efficient for some winning conditions

# OUTLOOK

---

- Experimental evaluation of the two approaches
  - Boolean combinations of safety and reachability conditions  $\rightsquigarrow$  weak Büchi
  - “Request-Response” conditions  $\rightsquigarrow$  Büchi
- Symbolic algorithms?
- Can we use the partition into  $V_{\circ}$  and  $V_{\square}$  when reducing the memory extension?