

# Distributed games and distributed control for asynchronous systems

P. Gastin, B. Lerman, M. Zeitoun

LIAFA, CNRS & Univ. Paris 7

# A control synthesis problem — sequential case

- Finite open transition system  $\mathcal{A}$  over alphabet



- Set of **good** behaviors: regular language  $\mathcal{G}$ .
- Find a controller  $\mathcal{C}$  (finite automaton) such that
  - $\mathcal{C}$  selects actions of  $\Sigma_0$  but cannot disable actions from  $\Sigma_1$ .
  - All  $\Sigma_0$ -maximal behaviors of  $\mathcal{A} \times \mathcal{C}$  should be in  $\mathcal{G}$ .

# A game associated with the control synthesis problem

Controller's positions:  $S_0 = Q \times \{0\}$

$(q, 0)$

$(p.b, 0)$

$(p, 0)$

$a \in \Sigma_0$

$b \in \Sigma_1$

$\varepsilon$

Environment's positions:  $S_1 = Q \times \{1\}$

$(q.a, 1)$

$(p, 1)$

$(p.b, 1)$

**Play:** finite or infinite sequence of moves starting from  $(s_{in}, 1)$ .

**Player 0 (the controller)** wins a play iff its label is in  $\mathcal{G}$ .

A strategy  $f$  for player 0 tells him how to play on  $S_0 = Q \times \{0\}$ .

A strategy  $f$  for player 0 is winning if all  $f$ -maximal  $f$ -plays are winning.

# Strategies

Memoryless: depends only on the current state.

$$f : S_0 \times \Sigma_0 \rightarrow S_1 \cup \text{Stop}$$

Perfect memory: depends on the whole prefix played so far.

$$f : S^*.S_0 \times \Sigma_0 \rightarrow S_1 \cup \text{Stop}$$

## Facts

- Player 0 has a winning strategy in the game iff there exists a controller for  $\mathcal{A}$ .
- Player 0 has a winning strategy in the game if and only if he has a **finite memory** winning strategy.
- One can decide whether player 0 has a winning strategy in that game.

# Summary

Games well suited to solve the control synthesis problem.

- System = game graph.
- Controller = a player.
- Environment = opponent.
- Good behavior = winning condition.
- Finding a controller = finding a winning strategy.

# Control synthesis problem — distributed case

- Communicating systems  $\mathcal{A}_i$  over a given architecture, on several locations.



- Set of **good** behaviors  $\mathcal{G}$ .

**Goal** Build **local controllers**  $\mathcal{C}_i$ , synchronized with  $\mathcal{A}_i$ , so that

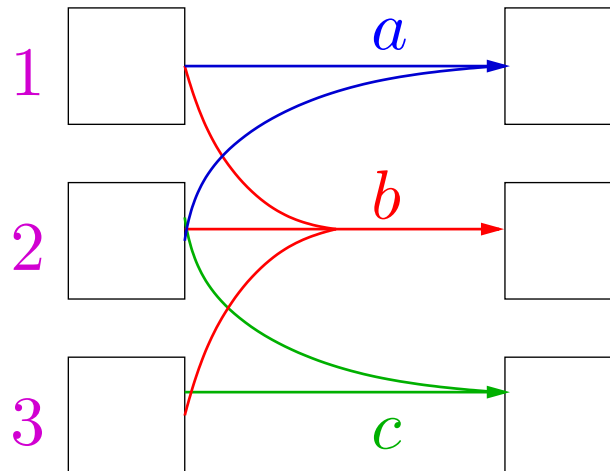
- $\mathcal{C}_i$  selects actions of  $\Sigma_{0,i}$  but cannot disable actions from  $\Sigma_{1,i}$ .
- The behavior of the overall system  $(\mathcal{A}_i \times \mathcal{C}_i)_i$  is in  $\mathcal{G}$  (regardless of the moves of the environment.)

# Distributed games

# Distributed architectures (Zielonka)

- $\Sigma$ : finite set of actions or players
- $\mathcal{P}$ : finite set of processes (memory cells)
- $R : \Sigma \rightarrow 2^{\mathcal{P}}$  assigns to each  $a \in \Sigma$  its read domain  $R(a)$
- $W : \Sigma \rightarrow \mathcal{P}$  assigns to each  $a \in \Sigma$  its write domain  $W(a)$
- Action  $a$  changes the state of process  $W(a)$  according to states read in  $R(a)$ .

$$\Sigma = \{a \ b \ c\}$$

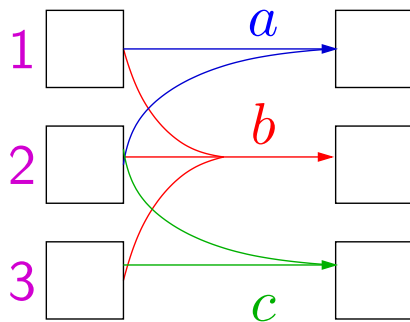


# Asynchronous distributed games (1)

$\langle (Q_i)_{i \in \mathcal{P}}, (T_a)_{a \in \Sigma}, q^0 \rangle$  distributed automaton over  $(\Sigma, \mathcal{P}, R, W)$

- $\forall i \in \mathcal{P}$ ,  $Q_i$  is the set of local states for process  $i$
- $\forall a \in \Sigma$ ,  $T_a \subseteq Q_{R(a)} \times Q_{W(a)}$ , possible local moves of action  $a$
- $q^0 \in Q = \prod_{i \in \mathcal{P}} Q_i$  is the global starting position

Architecture

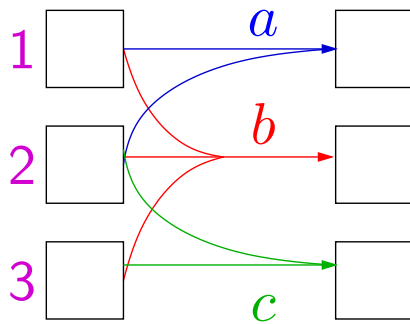


# Asynchronous distributed games (1)

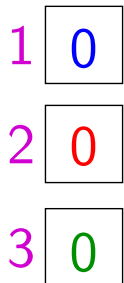
$\langle (Q_i)_{i \in \mathcal{P}}, (T_a)_{a \in \Sigma}, q^0 \rangle$  distributed automaton over  $(\Sigma, \mathcal{P}, R, W)$

- $\forall i \in \mathcal{P}$ ,  $Q_i$  is the set of local states for process  $i$
- $\forall a \in \Sigma$ ,  $T_a \subseteq Q_{R(a)} \times Q_{W(a)}$ , possible local moves of action  $a$
- $q^0 \in Q = \prod_{i \in \mathcal{P}} Q_i$  is the global starting position

Architecture



Global state

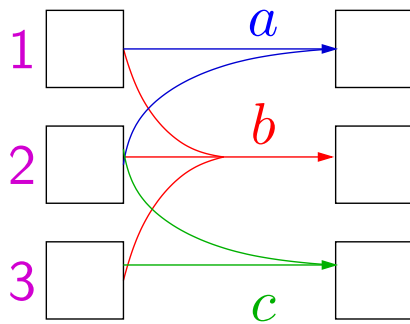


# Asynchronous distributed games (1)

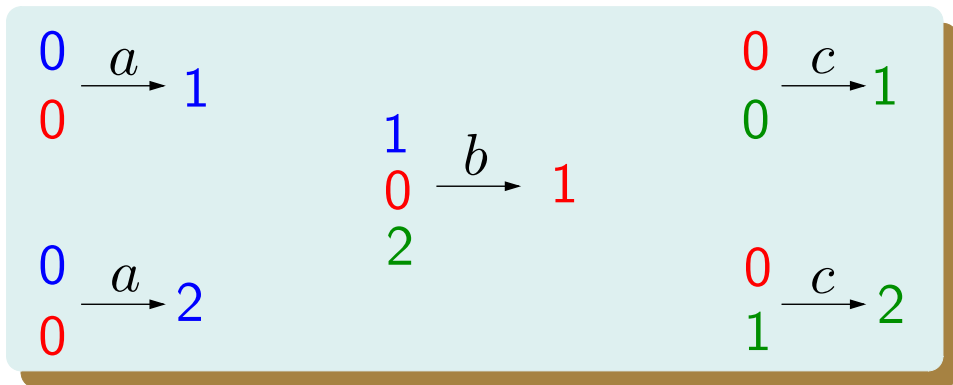
$\langle (Q_i)_{i \in \mathcal{P}}, (T_a)_{a \in \Sigma}, q^0 \rangle$  distributed automaton over  $(\Sigma, \mathcal{P}, R, W)$

- $\forall i \in \mathcal{P}$ ,  $Q_i$  is the set of local states for process  $i$
- $\forall a \in \Sigma$ ,  $T_a \subseteq Q_{R(a)} \times Q_{W(a)}$ , possible local moves of action  $a$
- $q^0 \in Q = \prod_{i \in \mathcal{P}} Q_i$  is the global starting position

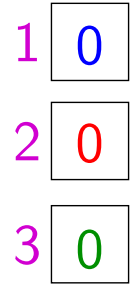
Architecture



Transitions



Global state

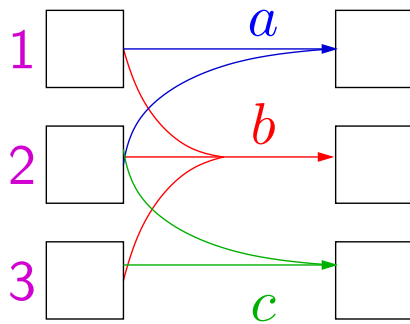


# Asynchronous distributed games (1)

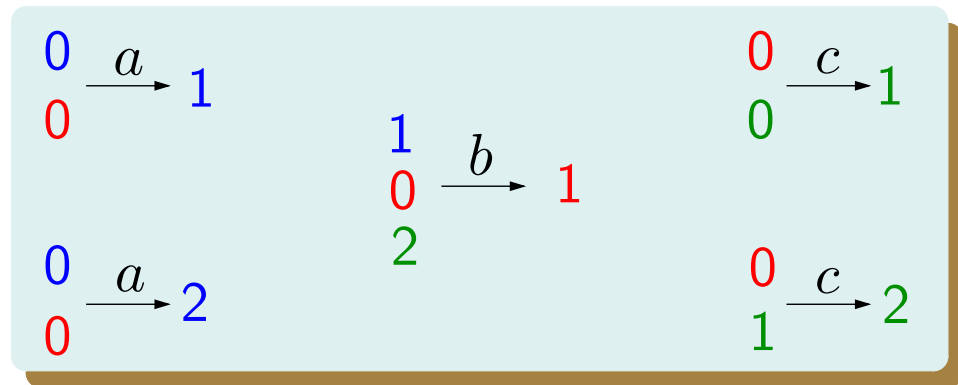
$\langle (Q_i)_{i \in \mathcal{P}}, (T_a)_{a \in \Sigma}, q^0 \rangle$  distributed automaton over  $(\Sigma, \mathcal{P}, R, W)$

- $\forall i \in \mathcal{P}$ ,  $Q_i$  is the set of local states for process  $i$
- $\forall a \in \Sigma$ ,  $T_a \subseteq Q_{R(a)} \times Q_{W(a)}$ , possible local moves of action  $a$
- $q^0 \in Q = \prod_{i \in \mathcal{P}} Q_i$  is the global starting position

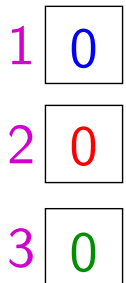
Architecture



Transitions



Global state

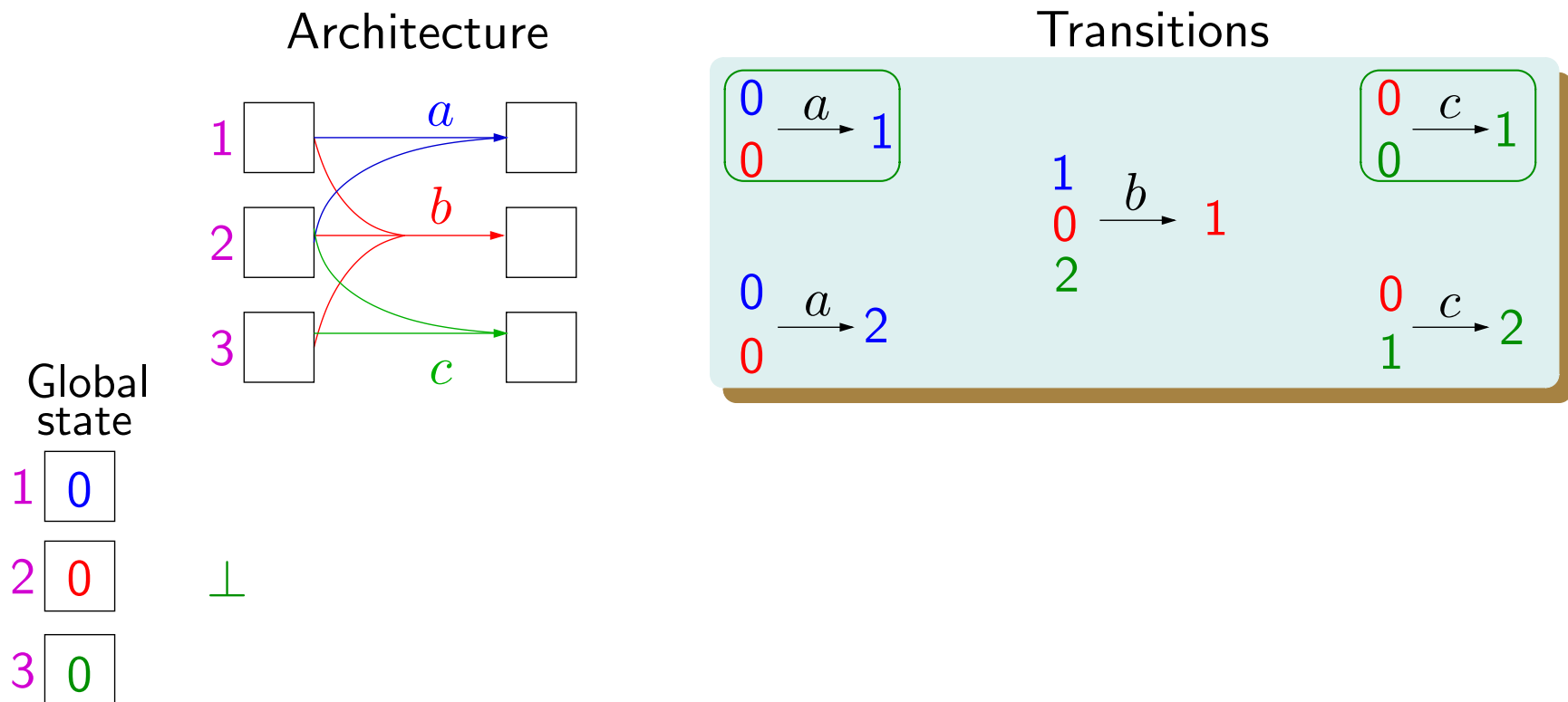


⊥

# Asynchronous distributed games (1)

$\langle (Q_i)_{i \in \mathcal{P}}, (T_a)_{a \in \Sigma}, q^0 \rangle$  distributed automaton over  $(\Sigma, \mathcal{P}, R, W)$

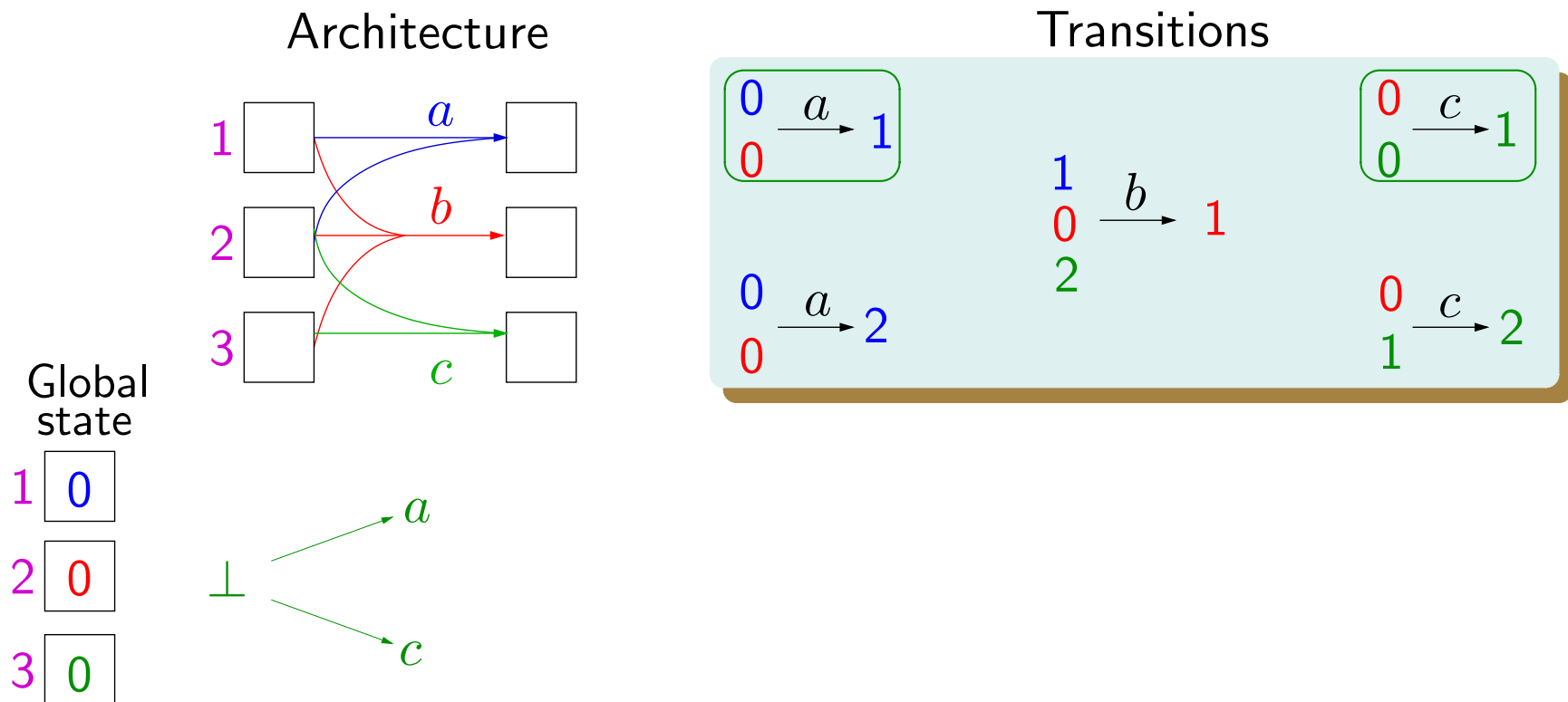
- $\forall i \in \mathcal{P}$ ,  $Q_i$  is the set of local states for process  $i$
- $\forall a \in \Sigma$ ,  $T_a \subseteq Q_{R(a)} \times Q_{W(a)}$ , possible local moves of action  $a$
- $q^0 \in Q = \prod_{i \in \mathcal{P}} Q_i$  is the global starting position



# Asynchronous distributed games (1)

$\langle (Q_i)_{i \in \mathcal{P}}, (T_a)_{a \in \Sigma}, q^0 \rangle$  distributed automaton over  $(\Sigma, \mathcal{P}, R, W)$

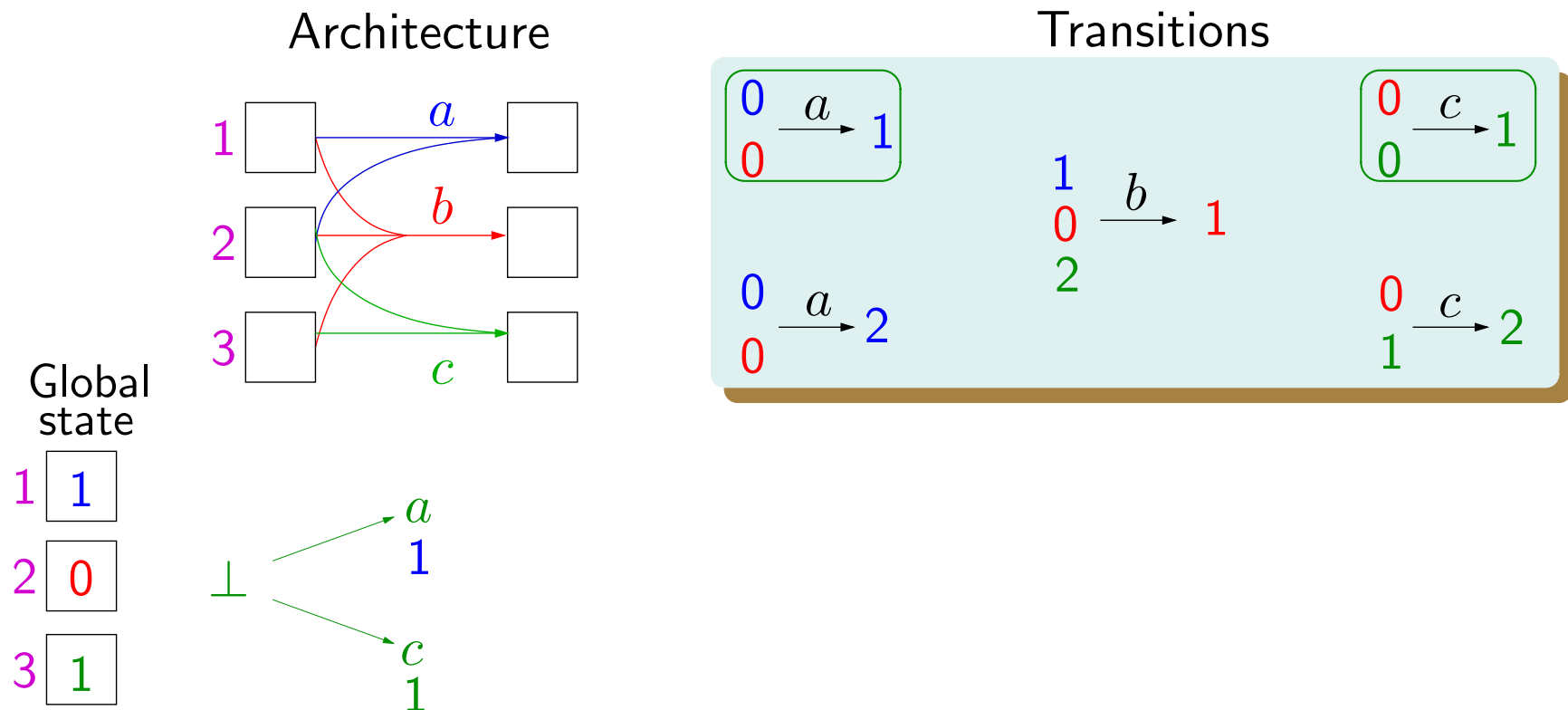
- $\forall i \in \mathcal{P}$ ,  $Q_i$  is the set of local states for process  $i$
- $\forall a \in \Sigma$ ,  $T_a \subseteq Q_{R(a)} \times Q_{W(a)}$ , possible local moves of action  $a$
- $q^0 \in Q = \prod_{i \in \mathcal{P}} Q_i$  is the global starting position



# Asynchronous distributed games (1)

$\langle (Q_i)_{i \in \mathcal{P}}, (T_a)_{a \in \Sigma}, q^0 \rangle$  distributed automaton over  $(\Sigma, \mathcal{P}, R, W)$

- $\forall i \in \mathcal{P}$ ,  $Q_i$  is the set of local states for process  $i$
- $\forall a \in \Sigma$ ,  $T_a \subseteq Q_{R(a)} \times Q_{W(a)}$ , possible local moves of action  $a$
- $q^0 \in Q = \prod_{i \in \mathcal{P}} Q_i$  is the global starting position

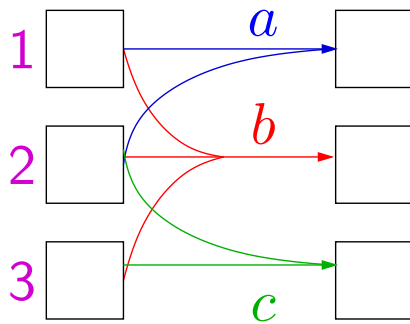


# Asynchronous distributed games (1)

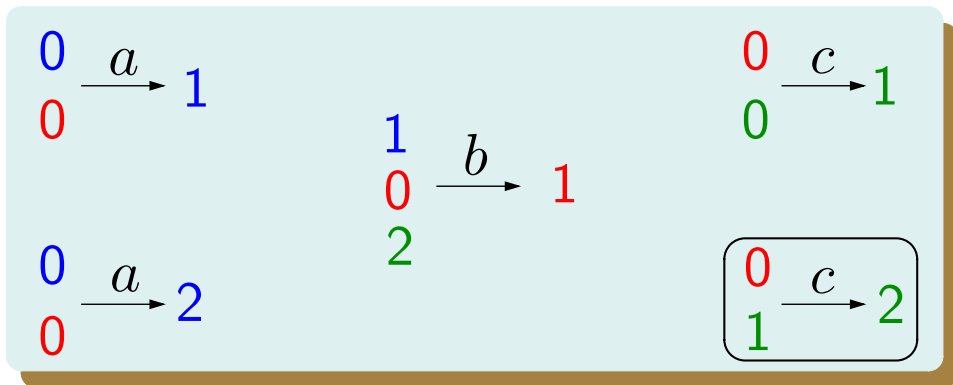
$\langle (Q_i)_{i \in \mathcal{P}}, (T_a)_{a \in \Sigma}, q^0 \rangle$  distributed automaton over  $(\Sigma, \mathcal{P}, R, W)$

- $\forall i \in \mathcal{P}$ ,  $Q_i$  is the set of local states for process  $i$
- $\forall a \in \Sigma$ ,  $T_a \subseteq Q_{R(a)} \times Q_{W(a)}$ , possible local moves of action  $a$
- $q^0 \in Q = \prod_{i \in \mathcal{P}} Q_i$  is the global starting position

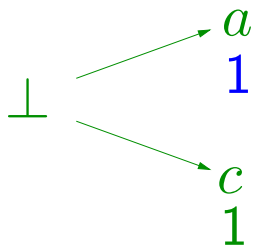
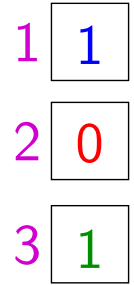
Architecture



Transitions



Global state

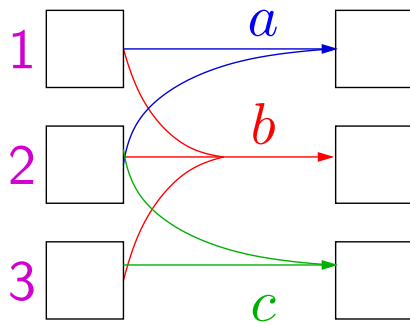


# Asynchronous distributed games (1)

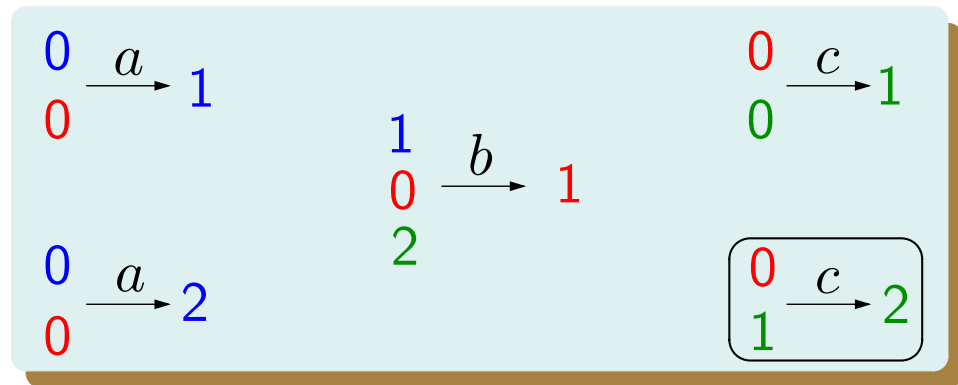
$\langle (Q_i)_{i \in \mathcal{P}}, (T_a)_{a \in \Sigma}, q^0 \rangle$  distributed automaton over  $(\Sigma, \mathcal{P}, R, W)$

- $\forall i \in \mathcal{P}$ ,  $Q_i$  is the set of local states for process  $i$
- $\forall a \in \Sigma$ ,  $T_a \subseteq Q_{R(a)} \times Q_{W(a)}$ , possible local moves of action  $a$
- $q^0 \in Q = \prod_{i \in \mathcal{P}} Q_i$  is the global starting position

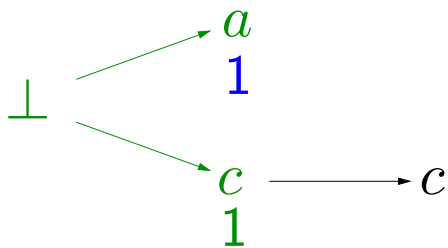
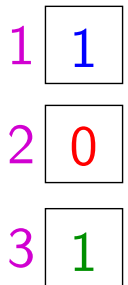
Architecture



Transitions



Global state

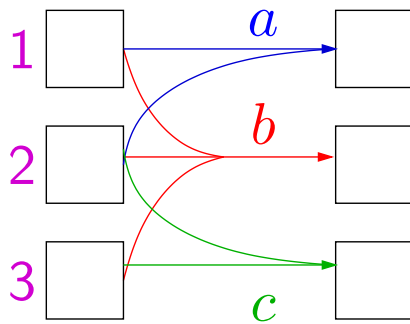


# Asynchronous distributed games (1)

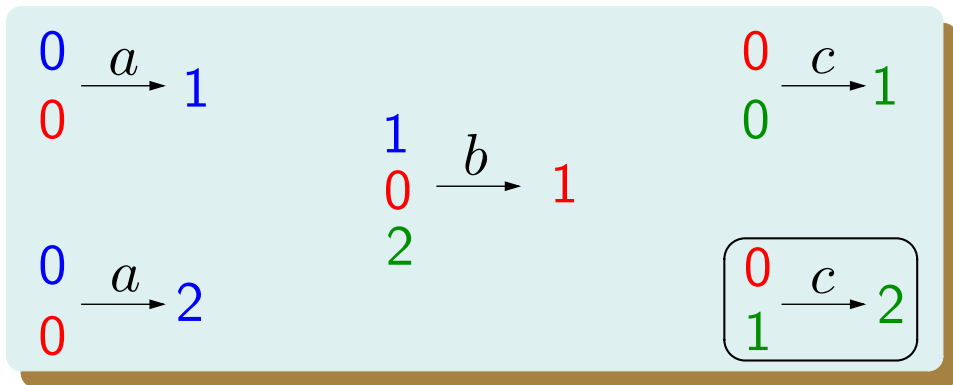
$\langle (Q_i)_{i \in \mathcal{P}}, (T_a)_{a \in \Sigma}, q^0 \rangle$  distributed automaton over  $(\Sigma, \mathcal{P}, R, W)$

- $\forall i \in \mathcal{P}$ ,  $Q_i$  is the set of local states for process  $i$
- $\forall a \in \Sigma$ ,  $T_a \subseteq Q_{R(a)} \times Q_{W(a)}$ , possible local moves of action  $a$
- $q^0 \in Q = \prod_{i \in \mathcal{P}} Q_i$  is the global starting position

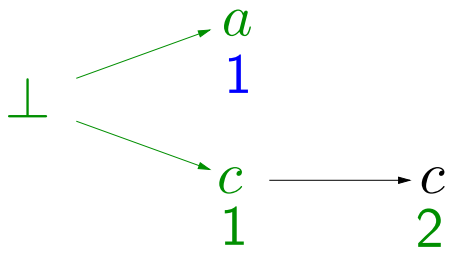
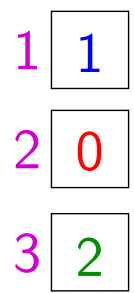
Architecture



Transitions



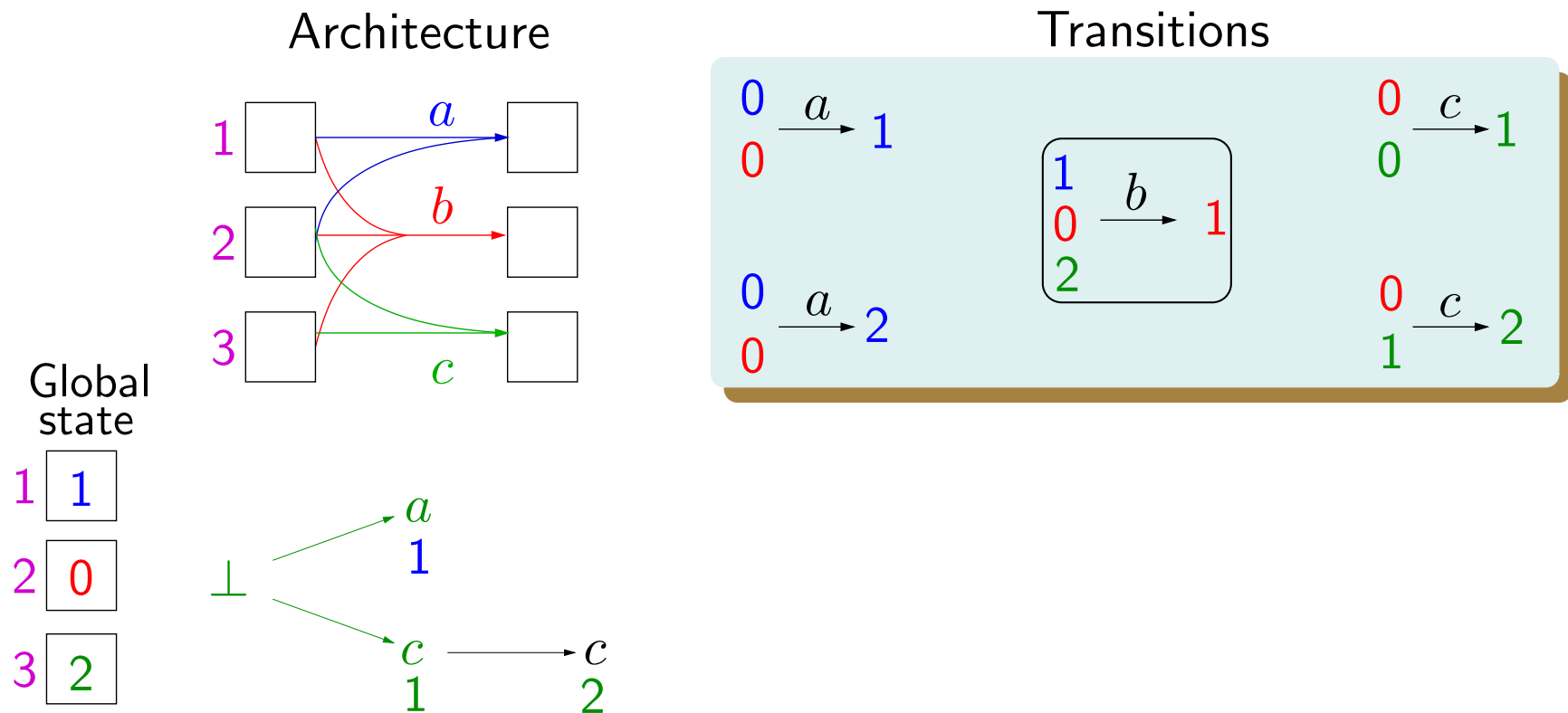
Global state



# Asynchronous distributed games (1)

$\langle (Q_i)_{i \in \mathcal{P}}, (T_a)_{a \in \Sigma}, q^0 \rangle$  distributed automaton over  $(\Sigma, \mathcal{P}, R, W)$

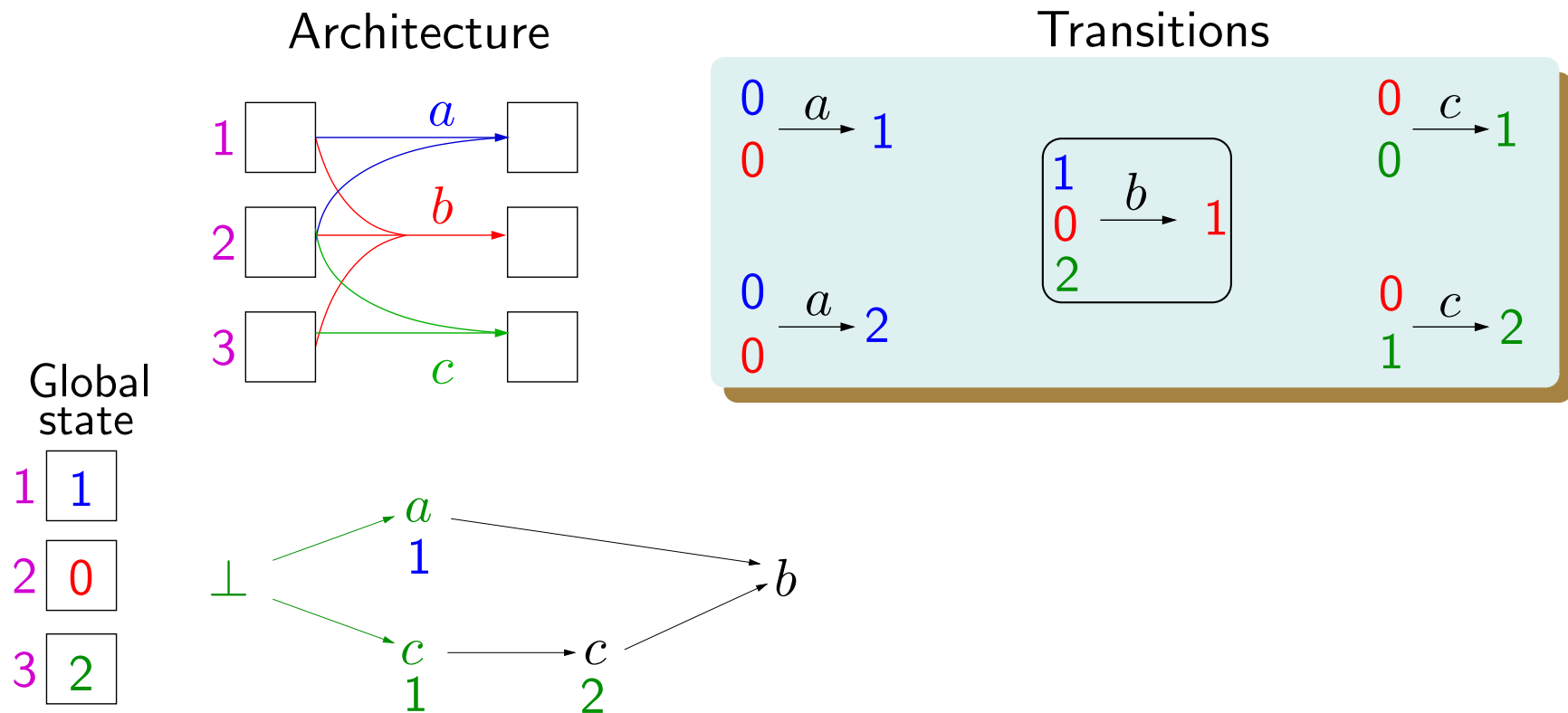
- $\forall i \in \mathcal{P}$ ,  $Q_i$  is the set of local states for process  $i$
- $\forall a \in \Sigma$ ,  $T_a \subseteq Q_{R(a)} \times Q_{W(a)}$ , possible local moves of action  $a$
- $q^0 \in Q = \prod_{i \in \mathcal{P}} Q_i$  is the global starting position



# Asynchronous distributed games (1)

$\langle (Q_i)_{i \in \mathcal{P}}, (T_a)_{a \in \Sigma}, q^0 \rangle$  distributed automaton over  $(\Sigma, \mathcal{P}, R, W)$

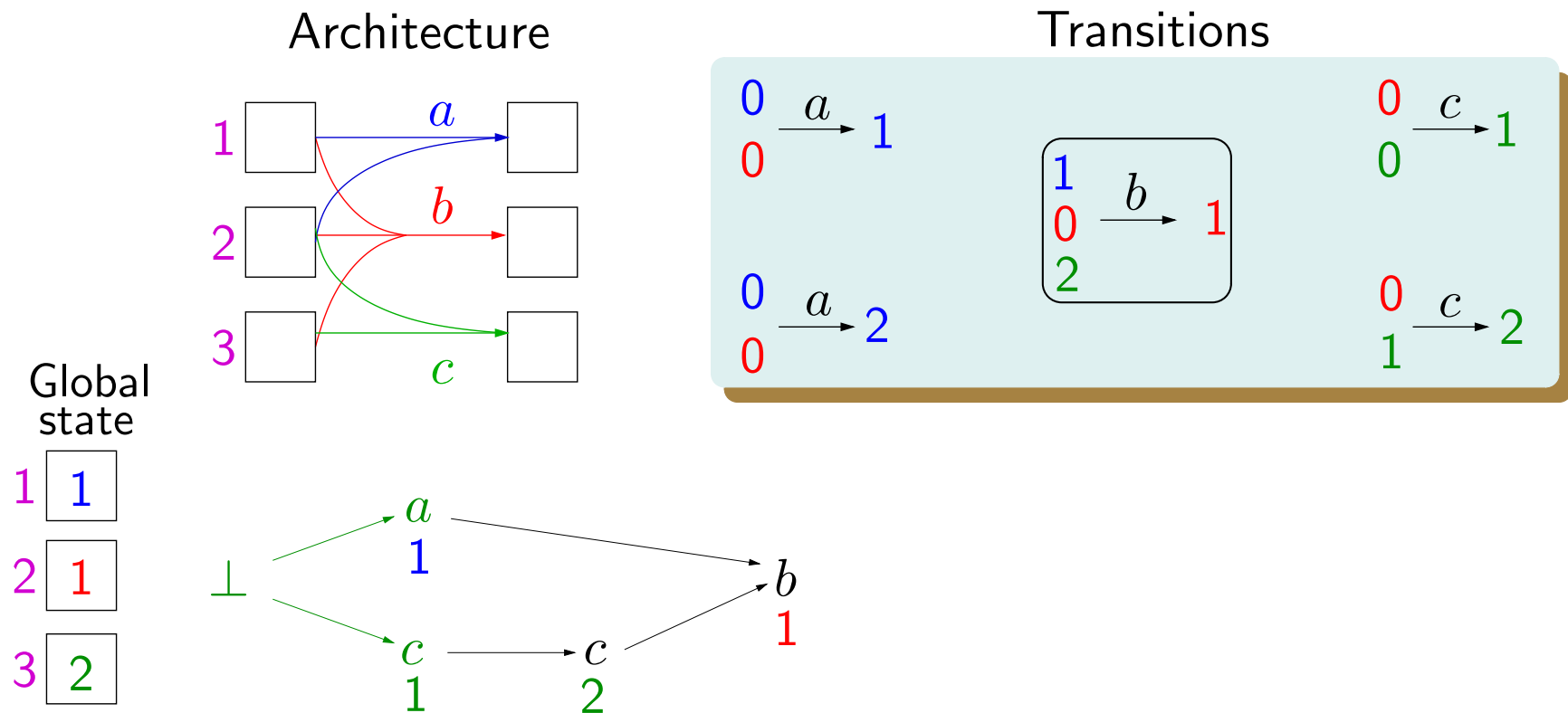
- $\forall i \in \mathcal{P}$ ,  $Q_i$  is the set of local states for process  $i$
- $\forall a \in \Sigma$ ,  $T_a \subseteq Q_{R(a)} \times Q_{W(a)}$ , possible local moves of action  $a$
- $q^0 \in Q = \prod_{i \in \mathcal{P}} Q_i$  is the global starting position



# Asynchronous distributed games (1)

$\langle (Q_i)_{i \in \mathcal{P}}, (T_a)_{a \in \Sigma}, q^0 \rangle$  distributed automaton over  $(\Sigma, \mathcal{P}, R, W)$

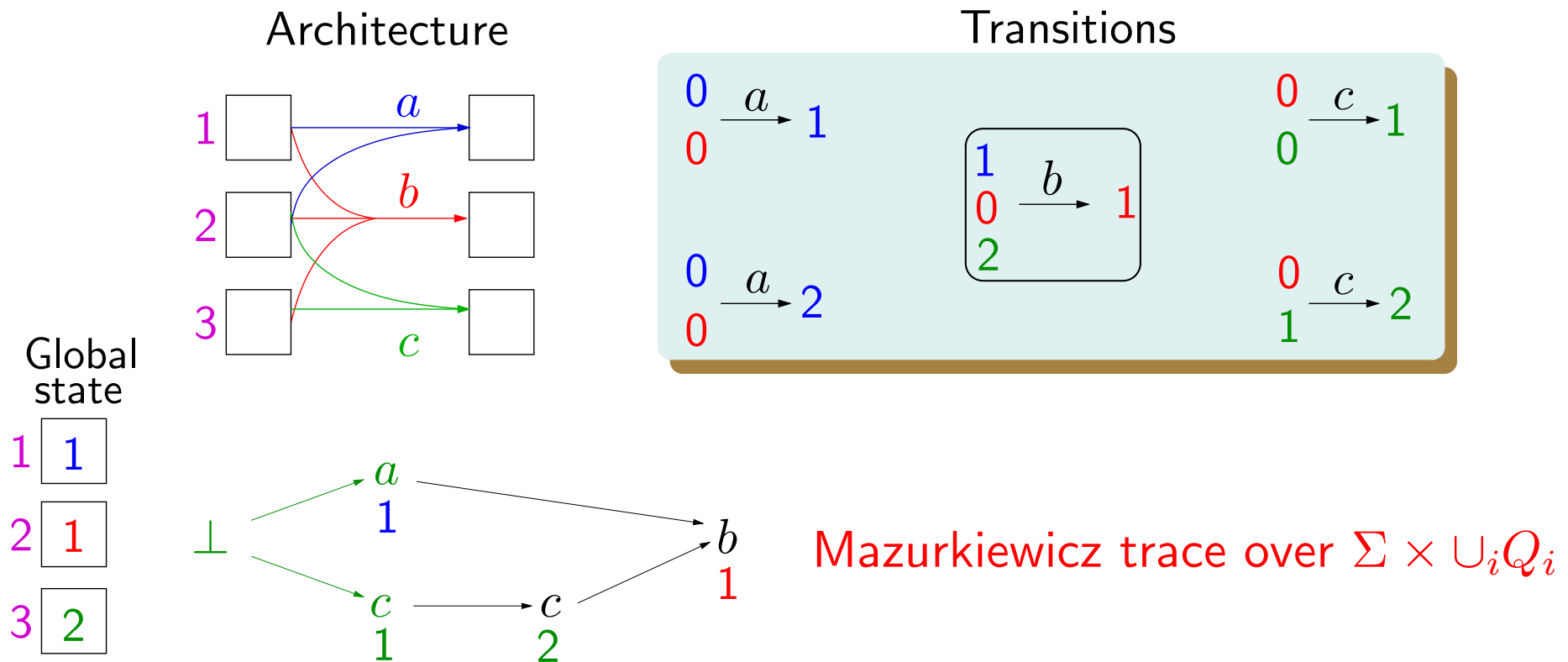
- $\forall i \in \mathcal{P}$ ,  $Q_i$  is the set of local states for process  $i$
- $\forall a \in \Sigma$ ,  $T_a \subseteq Q_{R(a)} \times Q_{W(a)}$ , possible local moves of action  $a$
- $q^0 \in Q = \prod_{i \in \mathcal{P}} Q_i$  is the global starting position



# Asynchronous distributed games (1)

$\langle (Q_i)_{i \in \mathcal{P}}, (T_a)_{a \in \Sigma}, q^0 \rangle$  distributed automaton over  $(\Sigma, \mathcal{P}, R, W)$

- $\forall i \in \mathcal{P}$ ,  $Q_i$  is the set of local states for process  $i$
- $\forall a \in \Sigma$ ,  $T_a \subseteq Q_{R(a)} \times Q_{W(a)}$ , possible local moves of action  $a$
- $q^0 \in Q = \prod_{i \in \mathcal{P}} Q_i$  is the global starting position



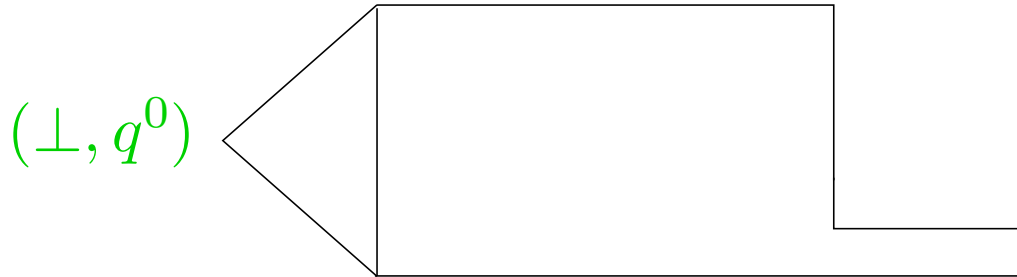
## Asynchronous distributed games (2)

- $\Sigma = \Sigma_0 \uplus \Sigma_1$ : partition of the set of players (actions) in teams 0 and 1.  
Players of team 0 cooperate together against team 1.
- Winning condition: set of finite or infinite traces  $\mathcal{W} \subseteq \mathbb{R}(\Sigma', D)$ .  
Team 0 wins plays of  $\mathcal{W}$  and loses plays of  $\mathbb{R}(\Sigma', D) \setminus \mathcal{W}$ .

# Distributed strategies

# Distributed strategies with local memory

Recall that a **distributed play** is a rooted trace of  $\mathbb{R}(\Sigma', D)$ .



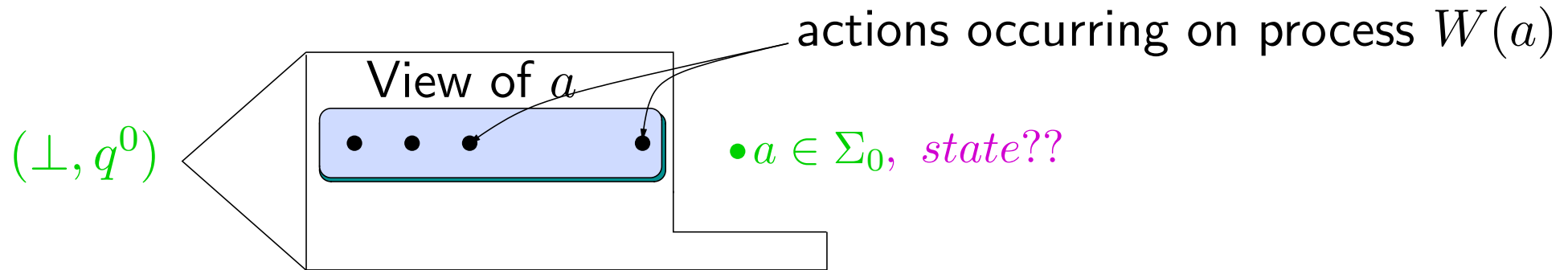
# Distributed strategies with local memory

Recall that a **distributed play** is a rooted trace of  $\mathbb{R}(\Sigma', D)$ .



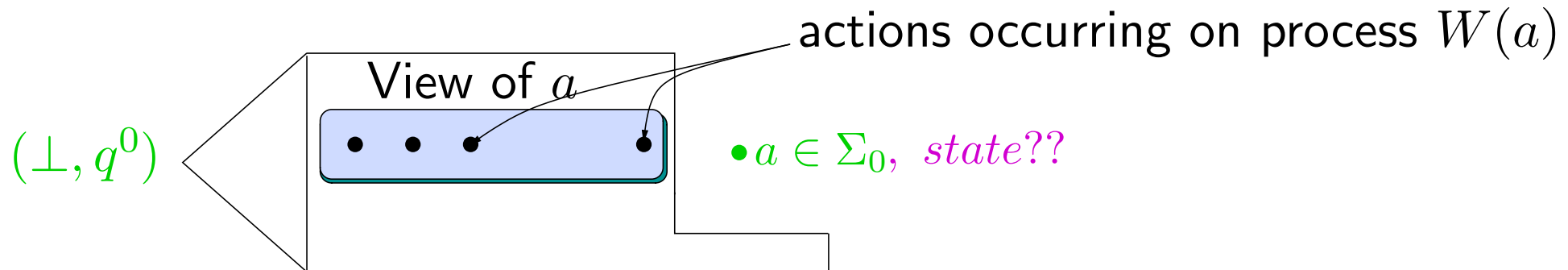
# Distributed strategies with local memory

Recall that a **distributed play** is a rooted trace of  $\mathbb{R}(\Sigma', D)$ .



# Distributed strategies with local memory

Recall that a **distributed play** is a rooted trace of  $\mathbb{R}(\Sigma', D)$ .



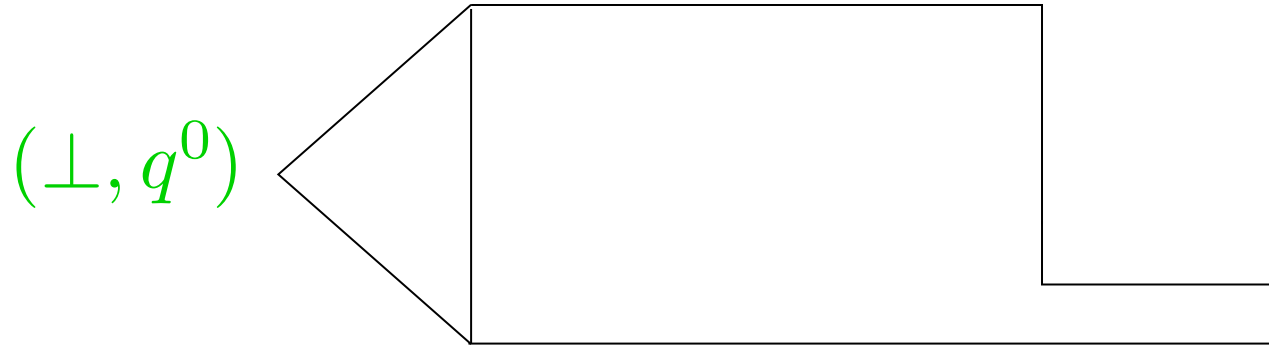
$$f : \text{View of } a, a \mapsto Q_a \cup \{\text{Stop}\}$$

The strategy for  $a$  does **not** use the **full history**, only  $a$ 's local view.

Used by [Thiagarajan-Madhusudan 02,03] and by [Bernet-Janin-Walukiewicz 04].

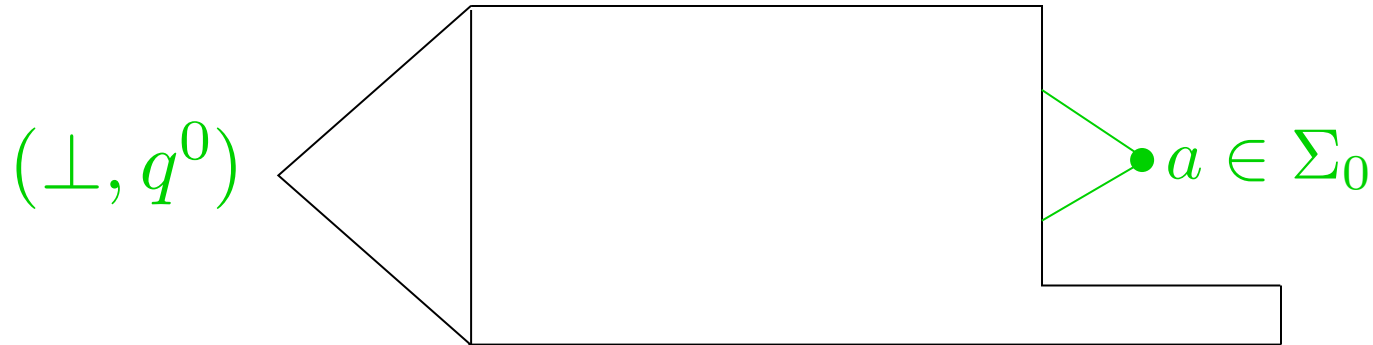
# Distributed strategies with causal memory

Recall that a **distributed play** is a rooted trace of  $\mathbb{R}(\Sigma', D)$ .



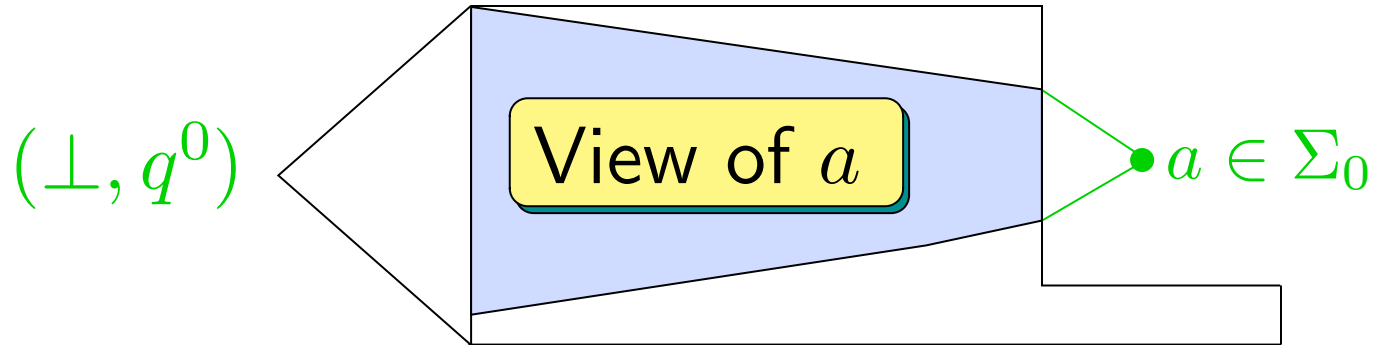
# Distributed strategies with causal memory

Recall that a **distributed play** is a rooted trace of  $\mathbb{R}(\Sigma', D)$ .



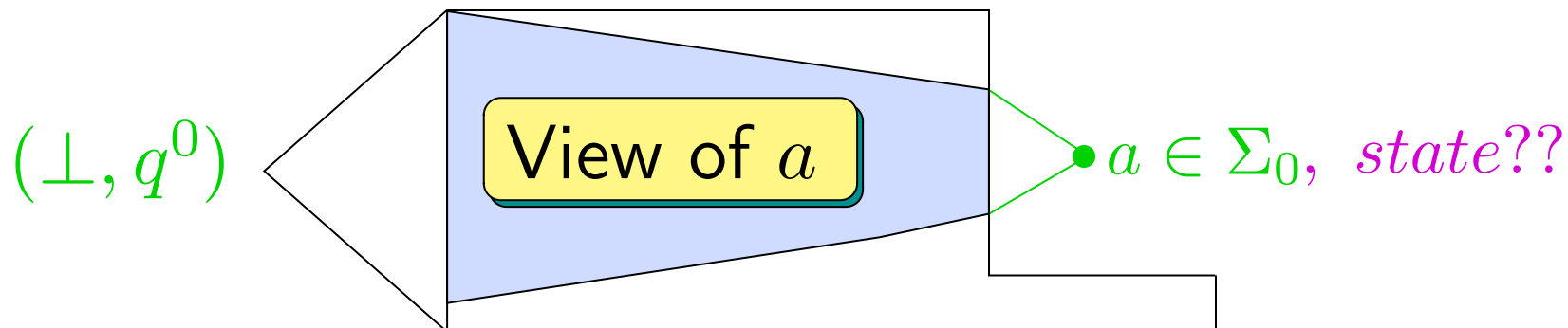
# Distributed strategies with causal memory

Recall that a **distributed play** is a rooted trace of  $\mathbb{R}(\Sigma', D)$ .



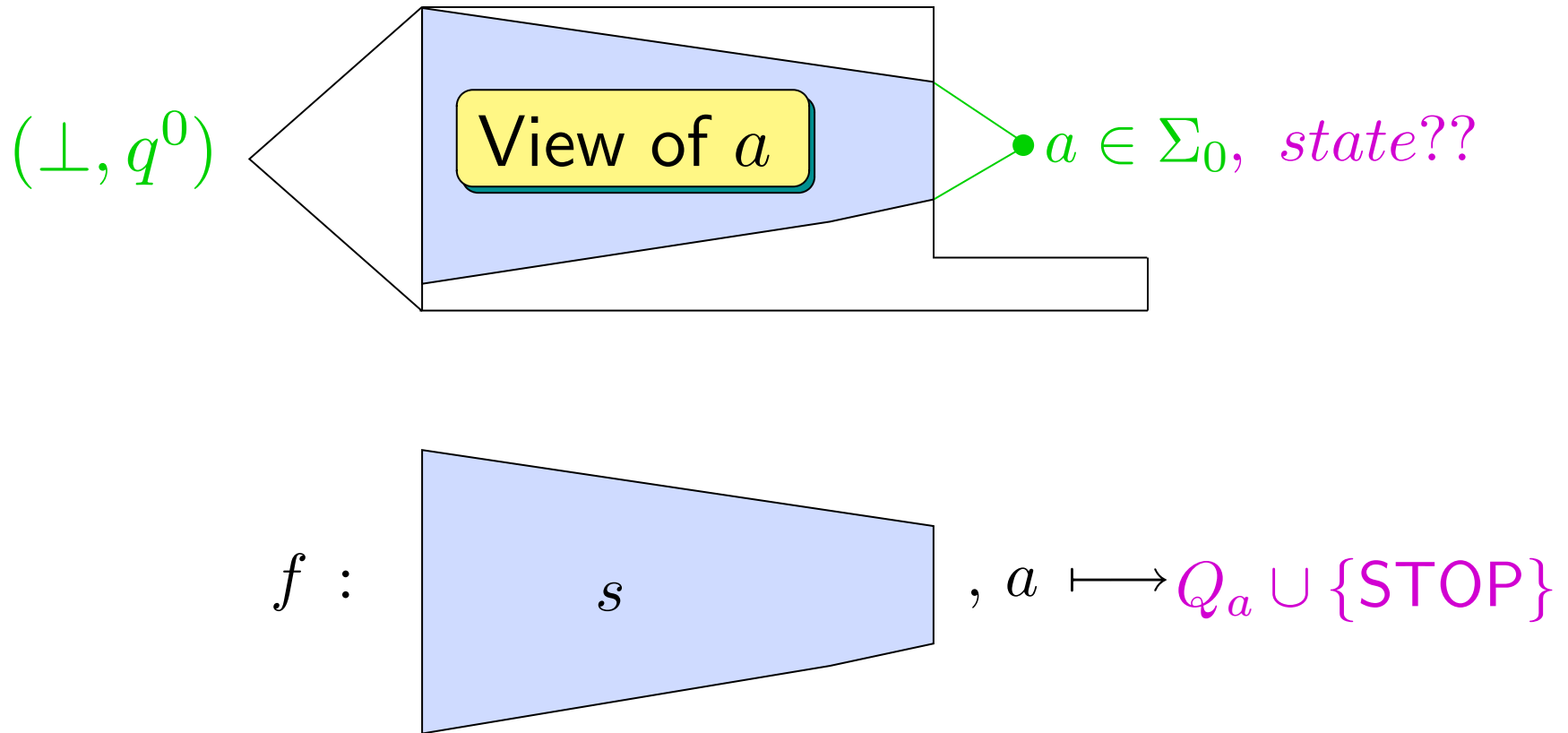
# Distributed strategies with causal memory

Recall that a **distributed play** is a rooted trace of  $\mathbb{R}(\Sigma', D)$ .



# Distributed strategies with causal memory

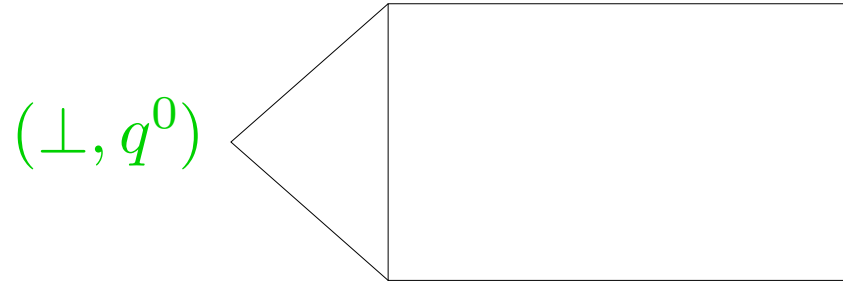
Recall that a **distributed play** is a rooted trace of  $\mathbb{R}(\Sigma', D)$ .



All other causal distributed strategies are abstractions of this perfect memory distributed strategy.

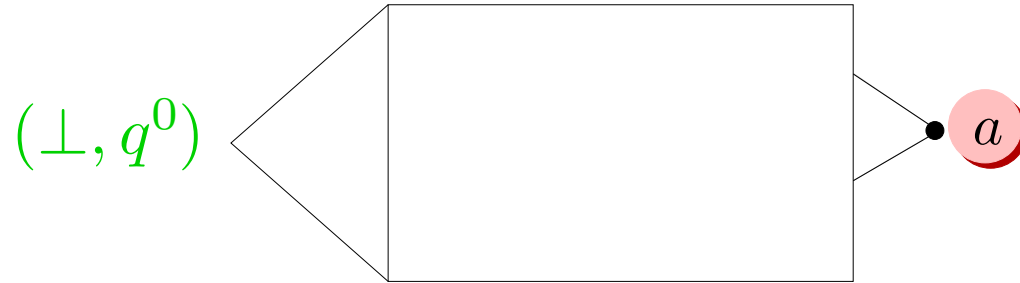
# Memoryless distributed strategies

Coarse abstraction of the causal perfect memory:



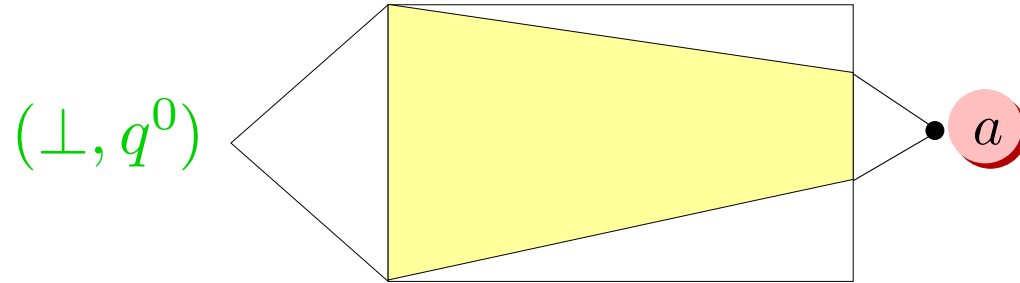
# Memoryless distributed strategies

Coarse abstraction of the causal perfect memory:



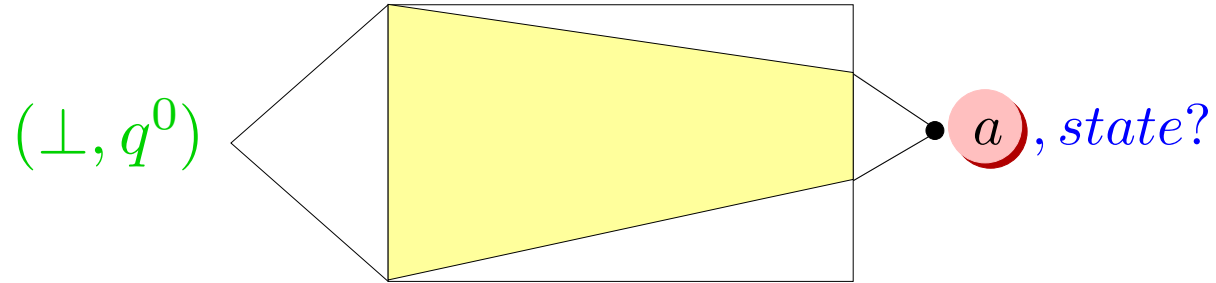
# Memoryless distributed strategies

Coarse abstraction of the causal perfect memory:



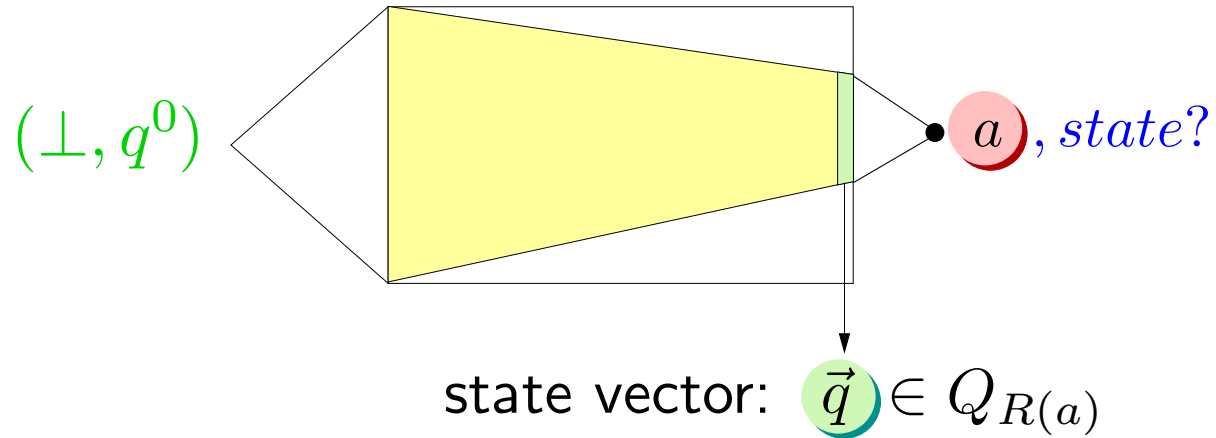
# Memoryless distributed strategies

Coarse abstraction of the causal perfect memory:



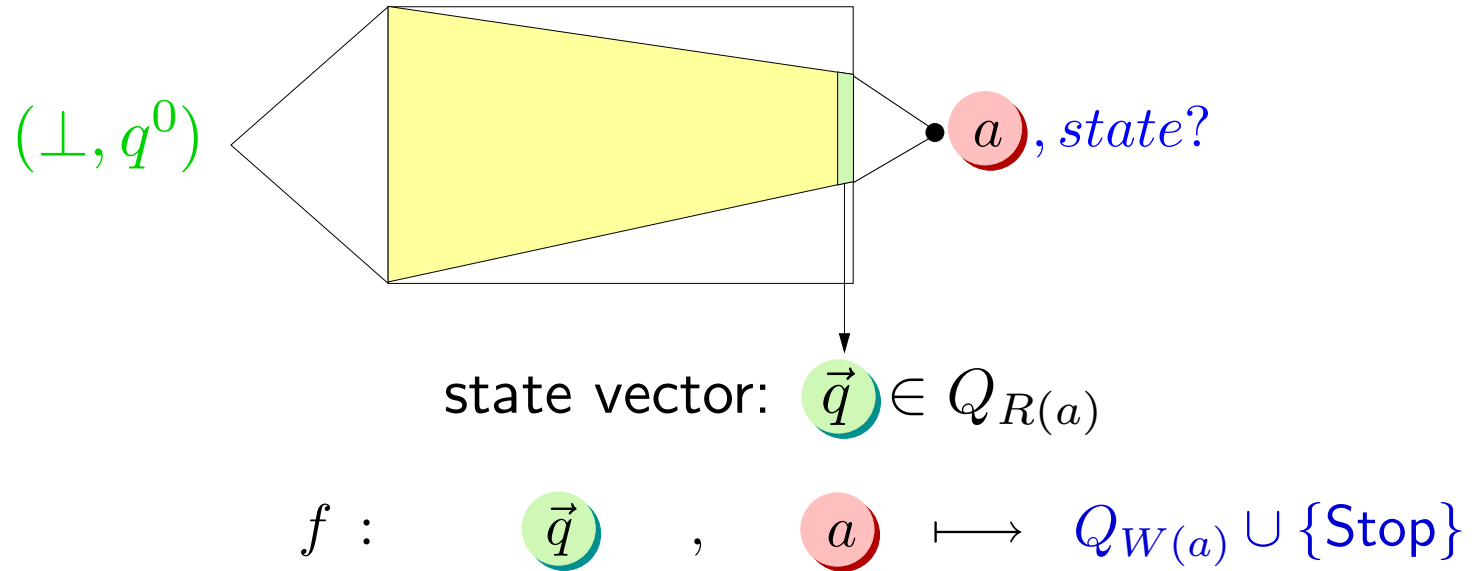
# Memoryless distributed strategies

Coarse abstraction of the causal perfect memory:



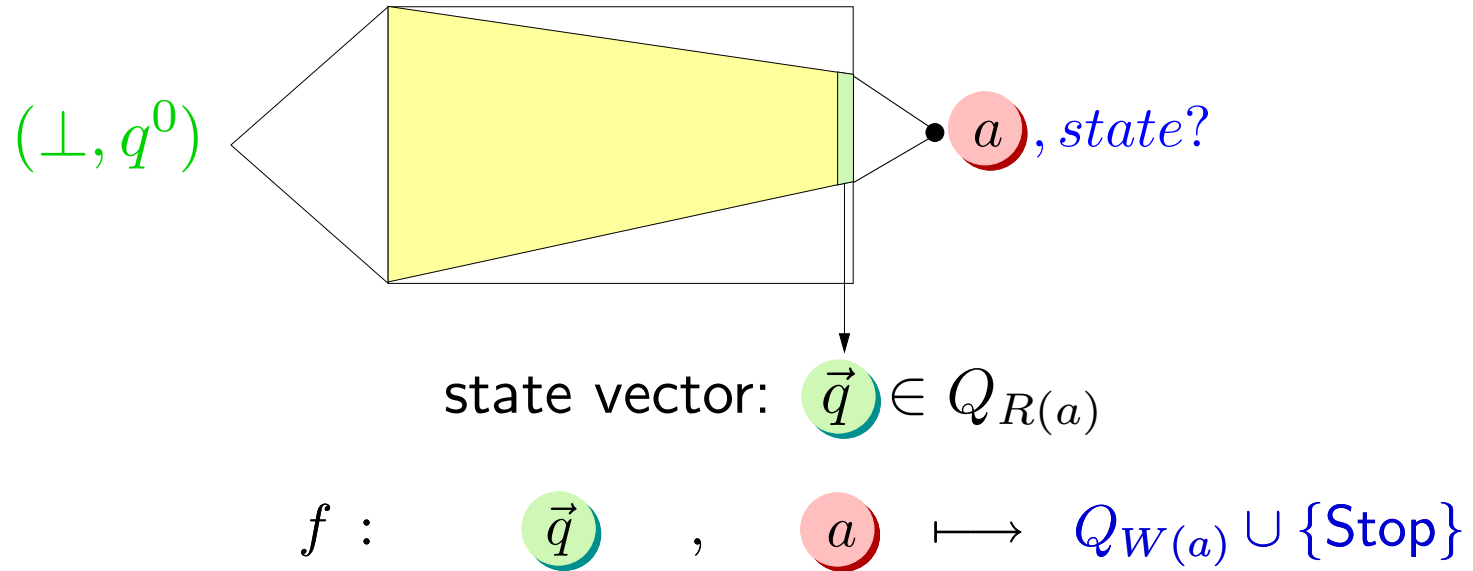
# Memoryless distributed strategies

Coarse abstraction of the causal perfect memory:



# Memoryless distributed strategies

Coarse abstraction of the causal perfect memory:



Finer abstractions of perfect memory  $\rightarrow$  distributed memories  $\mu$  and strategies.

**Proposition** For a distributed game  $G$  and a distributed memory  $\mu$ , one can build a game  $G^\mu$  such that team 0 has a WDS in  $G$  with memory  $\mu$  iff it has a memoryless WDS in  $G^\mu$ .

**Proof**  $G^\mu = G \times \mu$ .

# Deciding distributed strategies

**Proposition** Deciding whether team 0 has a causal distributed (memoryless) WS is undecidable for rational winning conditions.

**Proof.** Simple reduction of the universality problem for rational trace languages.

**Theorem** Deciding whether team 0 has a winning local distributed strategy is undecidable [BJW04] even:

- for reachability or safety winning conditions.
- with 3 players.

With causal memory, this undecidability result (with 3 players) does not hold.

# Deciding distributed strategies

**Proposition** Deciding whether team 0 has a causal distributed (memoryless) WS is undecidable for rational winning conditions.

**Proof.** Simple reduction of the universality problem for rational trace languages.

**Theorem** Deciding whether team 0 has a winning local distributed strategy is undecidable [BJW04] even:

- for reachability or safety winning conditions.
- with 3 players.

With causal memory, this undecidability result (with 3 players) does not hold.

**Conjecture** Whether team 0 has a causal distributed (memoryless) WS for reachability/safety/recognizable winning conditions is decidable.

# From a distributed game to its associated sequential game

**Theorem** Given a finite distributed game  $(G, \mathcal{W})$ , we can effectively build a finite sequential 2-players game  $(\tilde{G}, \tilde{\mathcal{W}})$  st. the following are equivalent:

1. There exists a **memoryless distributed** WS for team 0 in  $(G, \mathcal{W})$ .
2. There exists a memoryless WS for player 0 in  $(\tilde{G}, \tilde{\mathcal{W}})$ .
3. There exists a WS for player 0 in  $(\tilde{G}, \tilde{\mathcal{W}})$ .

# From a distributed game to its associated sequential game

**Theorem** Given a finite distributed game  $(G, \mathcal{W})$ , we can effectively build a finite sequential 2-players game  $(\tilde{G}, \tilde{\mathcal{W}})$  st. the following are equivalent:

1. There exists a **memoryless distributed** WS for team 0 in  $(G, \mathcal{W})$ .
2. There exists a memoryless WS for player 0 in  $(\tilde{G}, \tilde{\mathcal{W}})$ .
3. There exists a WS for player 0 in  $(\tilde{G}, \tilde{\mathcal{W}})$ .

**Naive idea** Consider the game on the global transition system.

# From a distributed game to its associated sequential game

**Theorem** Given a finite distributed game  $(G, \mathcal{W})$ , we can effectively build a finite sequential 2-players game  $(\tilde{G}, \tilde{\mathcal{W}})$  st. the following are equivalent:

1. There exists a **memoryless distributed** WS for team 0 in  $(G, \mathcal{W})$ .
2. There exists a memoryless WS for player 0 in  $(\tilde{G}, \tilde{\mathcal{W}})$ .
3. There exists a WS for player 0 in  $(\tilde{G}, \tilde{\mathcal{W}})$ .

**Naive idea** Consider the game on the global transition system.

**Main problem** The controller has more information than its causal memory.

# From a distributed game to its associated sequential game

**Theorem** Given a finite distributed game  $(G, \mathcal{W})$ , we can effectively build a finite sequential 2-players game  $(\tilde{G}, \tilde{\mathcal{W}})$  st. the following are equivalent:

1. There exists a **memoryless distributed** WS for team 0 in  $(G, \mathcal{W})$ .
2. There exists a memoryless WS for player 0 in  $(\tilde{G}, \tilde{\mathcal{W}})$ .
3. There exists a WS for player 0 in  $(\tilde{G}, \tilde{\mathcal{W}})$ .

**Naive idea** Consider the game on the global transition system.

**Main problem** The controller has more information than its causal memory.

## Solution

- The opponent controls the linearization to be played.
- Using resets moves, he can replay different linearizations for the same play.
- The winning condition  $\tilde{\mathcal{W}}$  makes sure that the strategy followed by the controller is indeed distributed.

# Sequentialized game $\tilde{G}$

**Player 1** wins if he shows that player 0

- does not reach  $\mathcal{W}$ , or
- does not follow a **distributed** strategy.

**Player 1** can

- decide which actions are used and in which order.
- investigate all possible linearizations of distributed plays with **reset moves**

Positions:  $Z = \underbrace{Q \times \Sigma_0}_{Z_0} \uplus \underbrace{Q \times (\Sigma_1 \cup \{0, 1, 2\})}_{Z_1}$ ,      player  $i$  plays on  $Z_i$ .

- $(q, a) \in Z$
- $q$  stores the global state;
  - $a \in \Sigma_i$ : player  $i$  has to play  $a$ . Player 0 may **refuse** the move
  - $a = 0$ : a reset move has just be performed by player 1.
  - $a = 1$ : coming from  $(p, b)$ , the  $b$ -move has been performed
  - $a = 2$ : player 0 refused the previous move.

# Sequentialized game: example

2 processes 1, 2.  $\Sigma_i = \{a_i, b_i\}$ ,  $R(a_i) = W(a_i) = 1$  and  $R(b_i) = W(b_i) = 2$ .

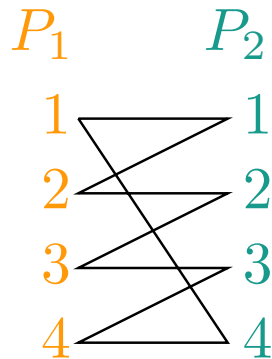
$\Sigma_1$ -transitions:  $q^0 \xrightarrow{a_1} \text{odd} \vee \text{even}$  and  $q^0 \xrightarrow{b_1} \text{odd} \vee \text{even}$

$\Sigma_0$ -transitions:  $\text{odd} \xrightarrow{a_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{a_0} 2 \vee 4$

$\text{odd} \xrightarrow{b_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{b_0} 2 \vee 4$

Plays will have the following shape:  $(\perp, q^0)$

$\mathcal{W} = \{(i, j) \mid i - j \equiv 0 \text{ or } 1 \pmod{4}\} \cup \text{Shape}_1$



# Sequentialized game: example

2 processes 1, 2.  $\Sigma_i = \{a_i, b_i\}$ ,  $R(a_i) = W(a_i) = 1$  and  $R(b_i) = W(b_i) = 2$ .

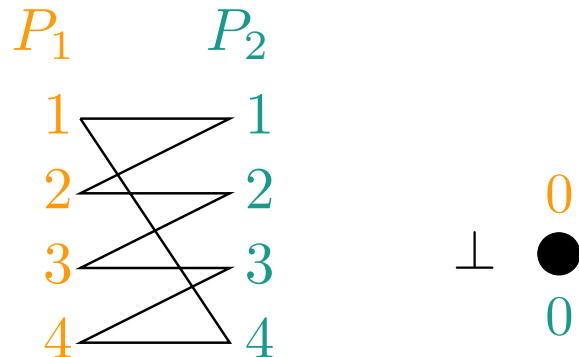
$\Sigma_1$ -transitions:  $q^0 \xrightarrow{a_1} \text{odd} \vee \text{even}$  and  $q^0 \xrightarrow{b_1} \text{odd} \vee \text{even}$

$\Sigma_0$ -transitions:  $\text{odd} \xrightarrow{a_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{a_0} 2 \vee 4$

$\text{odd} \xrightarrow{b_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{b_0} 2 \vee 4$

Plays will have the following shape:  $(\perp, q^0)$

$\mathcal{W} = \{(i, j) \mid i - j \equiv 0 \text{ or } 1 \pmod{4}\} \cup \text{Shape}_1$



# Sequentialized game: example

2 processes 1, 2.  $\Sigma_i = \{a_i, b_i\}$ ,  $R(a_i) = W(a_i) = 1$  and  $R(b_i) = W(b_i) = 2$ .

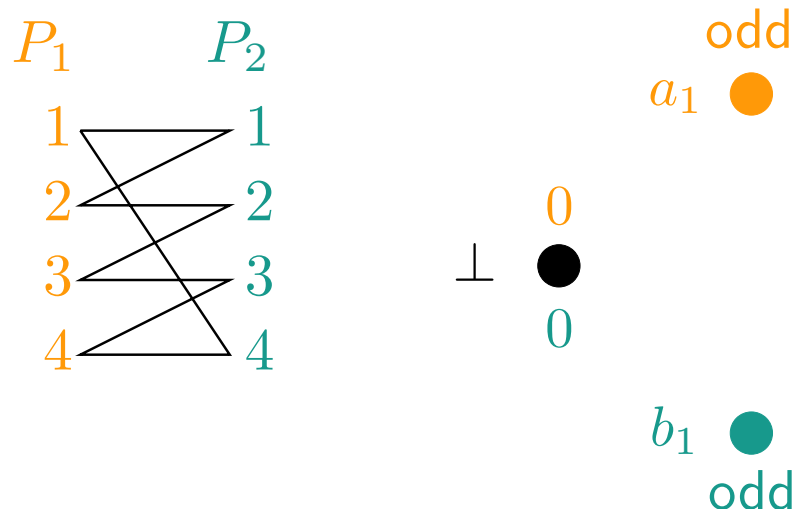
$\Sigma_1$ -transitions:  $q^0 \xrightarrow{a_1} \text{odd} \vee \text{even}$  and  $q^0 \xrightarrow{b_1} \text{odd} \vee \text{even}$

$\Sigma_0$ -transitions:  $\text{odd} \xrightarrow{a_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{a_0} 2 \vee 4$

$\text{odd} \xrightarrow{b_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{b_0} 2 \vee 4$

Plays will have the following shape:  $(\perp, q^0)$

$\mathcal{W} = \{(i, j) \mid i - j \equiv 0 \text{ or } 1 \pmod{4}\} \cup \text{Shape}_1$



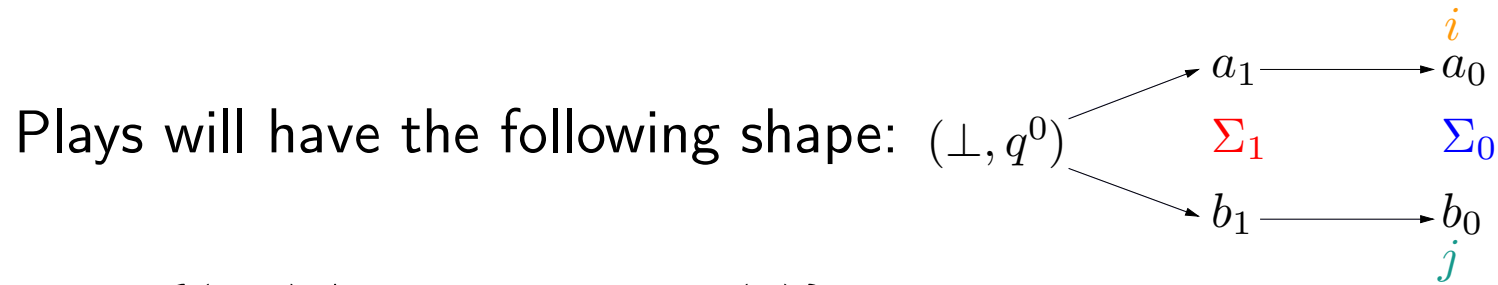
# Sequentialized game: example

2 processes 1, 2.  $\Sigma_i = \{a_i, b_i\}$ ,  $R(a_i) = W(a_i) = 1$  and  $R(b_i) = W(b_i) = 2$ .

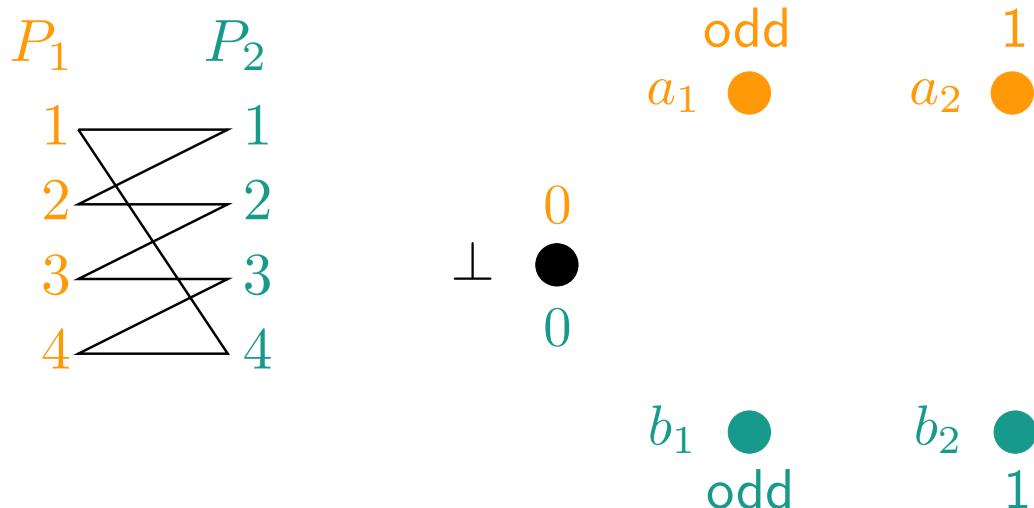
$\Sigma_1$ -transitions:  $q^0 \xrightarrow{a_1} \text{odd} \vee \text{even}$  and  $q^0 \xrightarrow{b_1} \text{odd} \vee \text{even}$

$\Sigma_0$ -transitions:  $\text{odd} \xrightarrow{a_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{a_0} 2 \vee 4$

$\text{odd} \xrightarrow{b_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{b_0} 2 \vee 4$



$$\mathcal{W} = \{(i, j) \mid i - j \equiv 0 \text{ or } 1 \pmod{4}\} \cup \text{Shape}_1$$



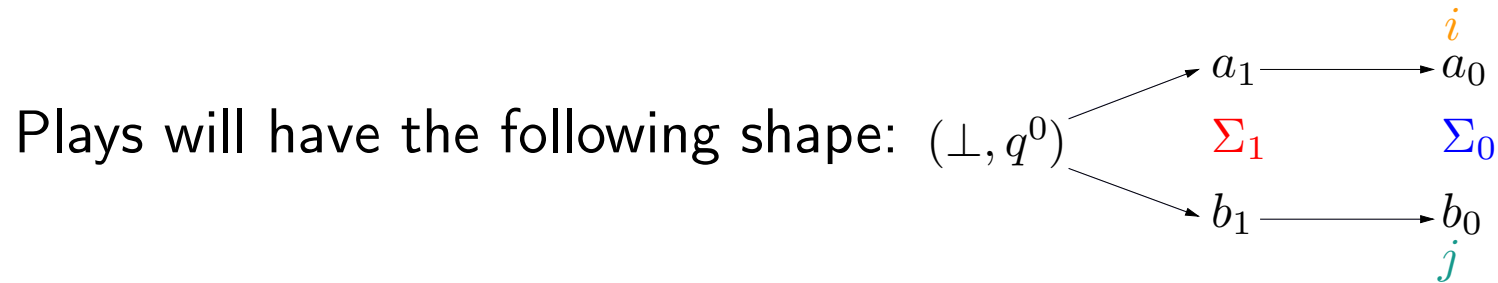
# Sequentialized game: example

2 processes 1, 2.  $\Sigma_i = \{a_i, b_i\}$ ,  $R(a_i) = W(a_i) = 1$  and  $R(b_i) = W(b_i) = 2$ .

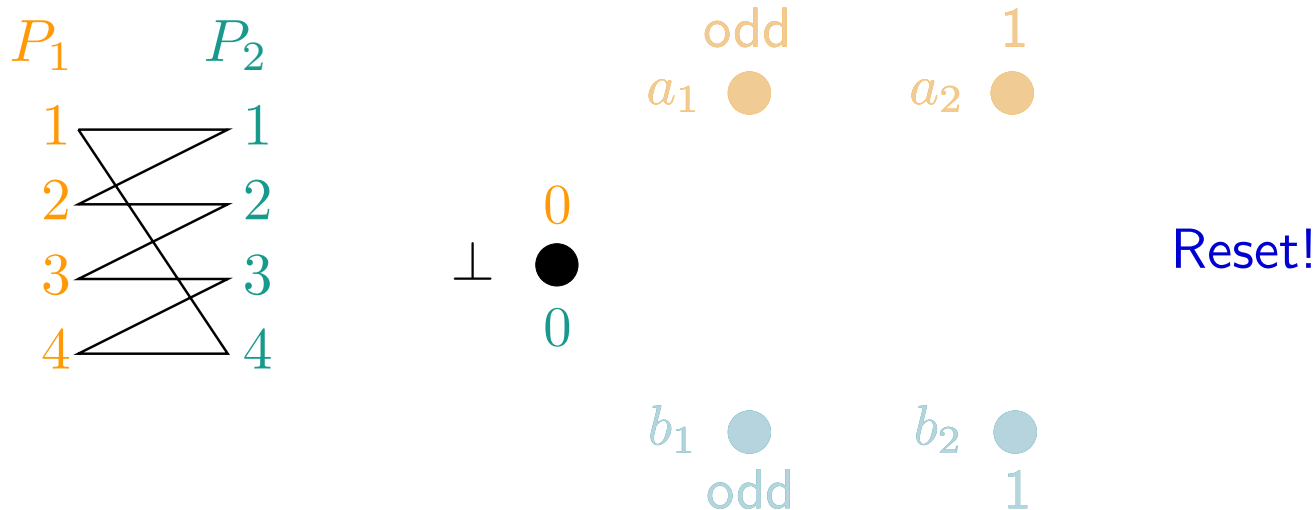
$\Sigma_1$ -transitions:  $q^0 \xrightarrow{a_1} \text{odd} \vee \text{even}$  and  $q^0 \xrightarrow{b_1} \text{odd} \vee \text{even}$

$\Sigma_0$ -transitions:  $\text{odd} \xrightarrow{a_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{a_0} 2 \vee 4$

$\text{odd} \xrightarrow{b_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{b_0} 2 \vee 4$



$$\mathcal{W} = \{(i, j) \mid i - j \equiv 0 \text{ or } 1 \pmod{4}\} \cup \text{Shape}_1$$



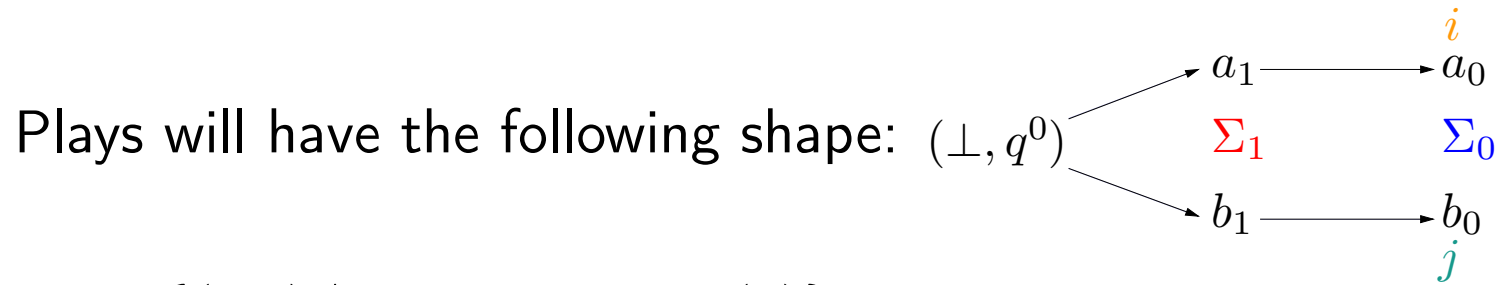
# Sequentialized game: example

2 processes 1, 2.  $\Sigma_i = \{a_i, b_i\}$ ,  $R(a_i) = W(a_i) = 1$  and  $R(b_i) = W(b_i) = 2$ .

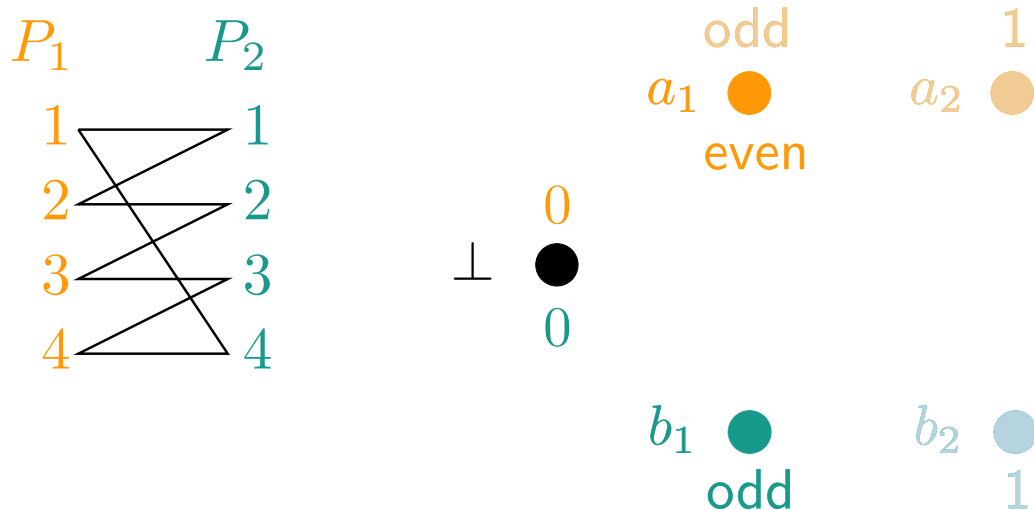
$\Sigma_1$ -transitions:  $q^0 \xrightarrow{a_1} \text{odd} \vee \text{even}$  and  $q^0 \xrightarrow{b_1} \text{odd} \vee \text{even}$

$\Sigma_0$ -transitions:  $\text{odd} \xrightarrow{a_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{a_0} 2 \vee 4$

$\text{odd} \xrightarrow{b_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{b_0} 2 \vee 4$



$$\mathcal{W} = \{(i, j) \mid i - j \equiv 0 \text{ or } 1 \pmod{4}\} \cup \text{Shape}_1$$



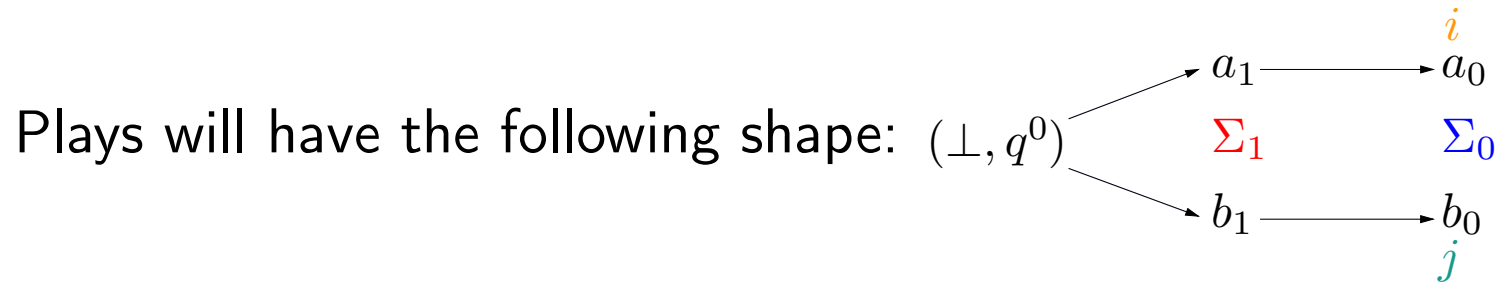
# Sequentialized game: example

2 processes 1, 2.  $\Sigma_i = \{a_i, b_i\}$ ,  $R(a_i) = W(a_i) = 1$  and  $R(b_i) = W(b_i) = 2$ .

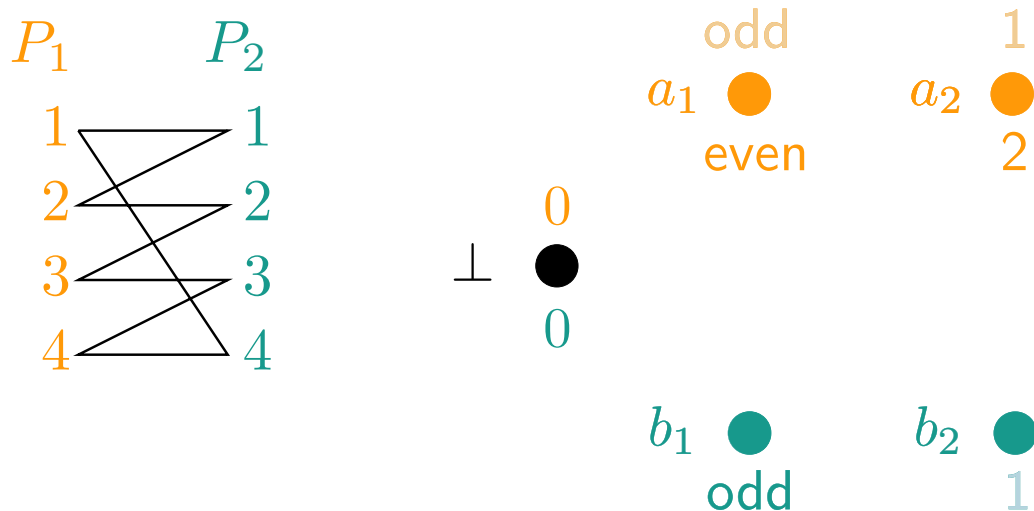
$\Sigma_1$ -transitions:  $q^0 \xrightarrow{a_1} \text{odd} \vee \text{even}$  and  $q^0 \xrightarrow{b_1} \text{odd} \vee \text{even}$

$\Sigma_0$ -transitions:  $\text{odd} \xrightarrow{a_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{a_0} 2 \vee 4$

$\text{odd} \xrightarrow{b_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{b_0} 2 \vee 4$



$$\mathcal{W} = \{(i, j) \mid i - j \equiv 0 \text{ or } 1 \pmod{4}\} \cup \text{Shape}_1$$



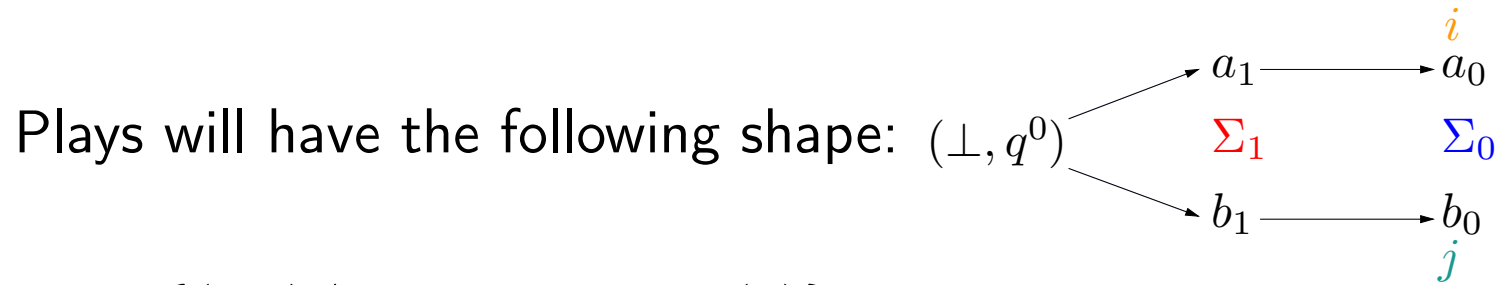
# Sequentialized game: example

2 processes 1, 2.  $\Sigma_i = \{a_i, b_i\}$ ,  $R(a_i) = W(a_i) = 1$  and  $R(b_i) = W(b_i) = 2$ .

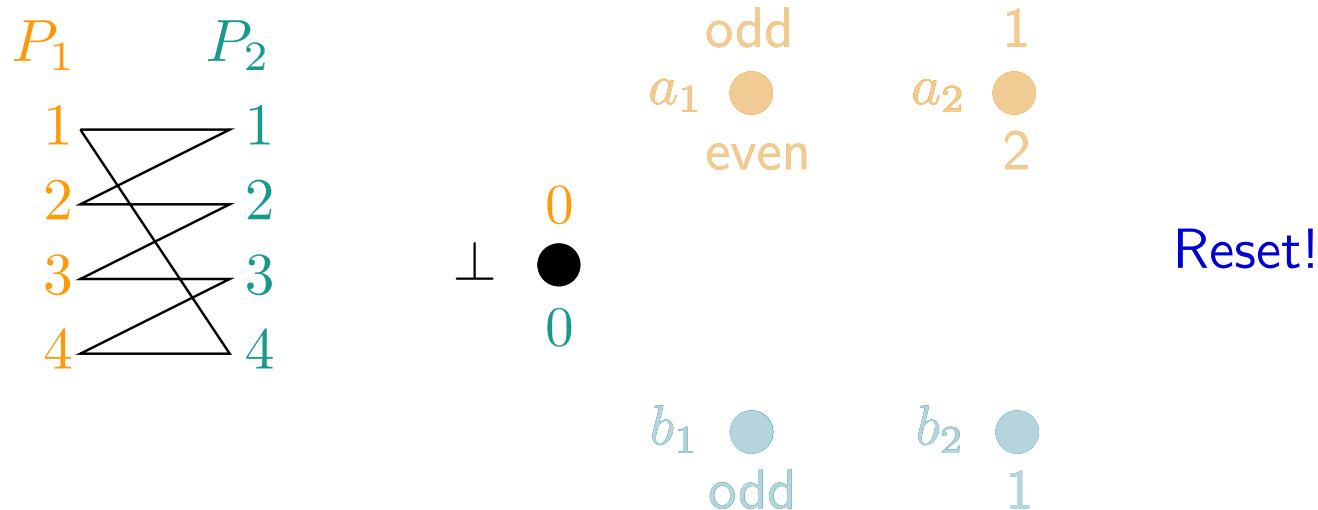
$\Sigma_1$ -transitions:  $q^0 \xrightarrow{a_1} \text{odd} \vee \text{even}$  and  $q^0 \xrightarrow{b_1} \text{odd} \vee \text{even}$

$\Sigma_0$ -transitions:  $\text{odd} \xrightarrow{a_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{a_0} 2 \vee 4$

$\text{odd} \xrightarrow{b_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{b_0} 2 \vee 4$



$$\mathcal{W} = \{(i, j) \mid i - j \equiv 0 \text{ or } 1 \pmod{4}\} \cup \text{Shape}_1$$



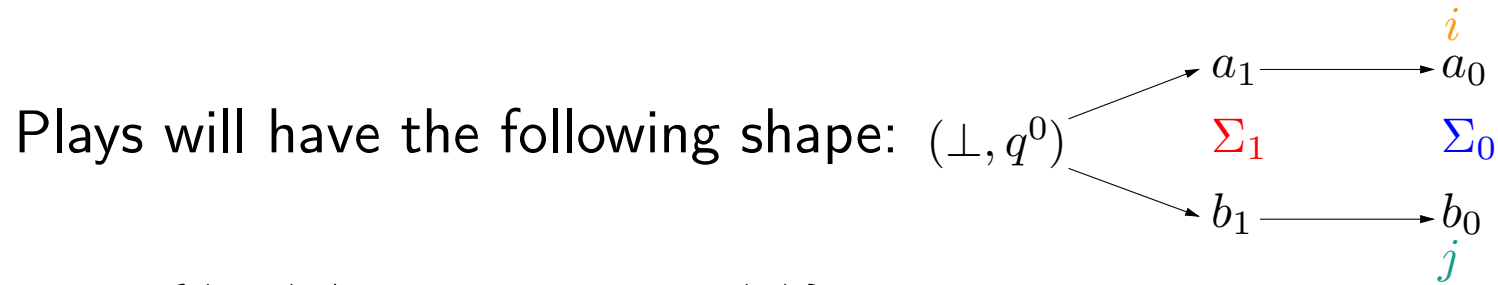
# Sequentialized game: example

2 processes 1, 2.  $\Sigma_i = \{a_i, b_i\}$ ,  $R(a_i) = W(a_i) = 1$  and  $R(b_i) = W(b_i) = 2$ .

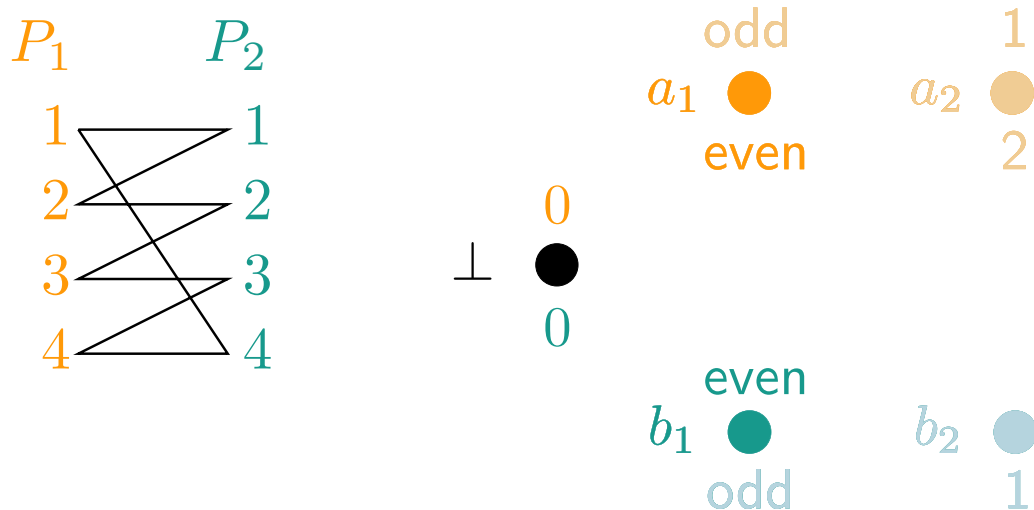
$\Sigma_1$ -transitions:  $q^0 \xrightarrow{a_1} \text{odd} \vee \text{even}$  and  $q^0 \xrightarrow{b_1} \text{odd} \vee \text{even}$

$\Sigma_0$ -transitions:  $\text{odd} \xrightarrow{a_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{a_0} 2 \vee 4$

$\text{odd} \xrightarrow{b_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{b_0} 2 \vee 4$



$$\mathcal{W} = \{(i, j) \mid i - j \equiv 0 \text{ or } 1 \pmod{4}\} \cup \text{Shape}_1$$



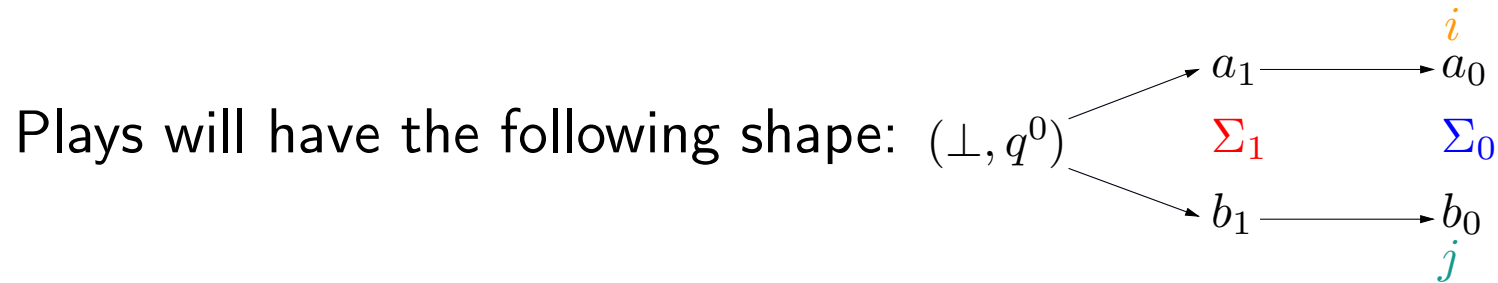
# Sequentialized game: example

2 processes 1, 2.  $\Sigma_i = \{a_i, b_i\}$ ,  $R(a_i) = W(a_i) = 1$  and  $R(b_i) = W(b_i) = 2$ .

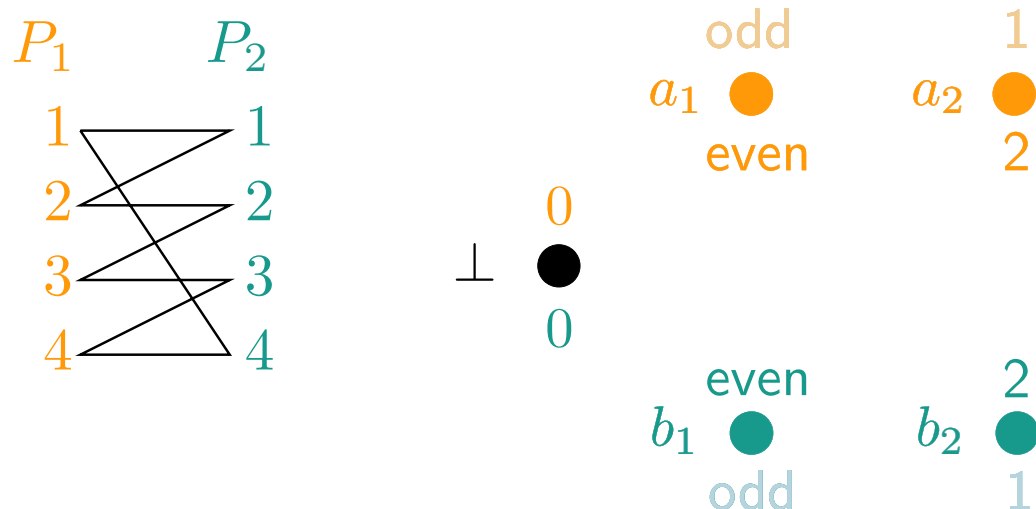
$\Sigma_1$ -transitions:  $q^0 \xrightarrow{a_1} \text{odd} \vee \text{even}$  and  $q^0 \xrightarrow{b_1} \text{odd} \vee \text{even}$

$\Sigma_0$ -transitions:  $\text{odd} \xrightarrow{a_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{a_0} 2 \vee 4$

$\text{odd} \xrightarrow{b_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{b_0} 2 \vee 4$



$$\mathcal{W} = \{(i, j) \mid i - j \equiv 0 \text{ or } 1 \pmod{4}\} \cup \text{Shape}_1$$



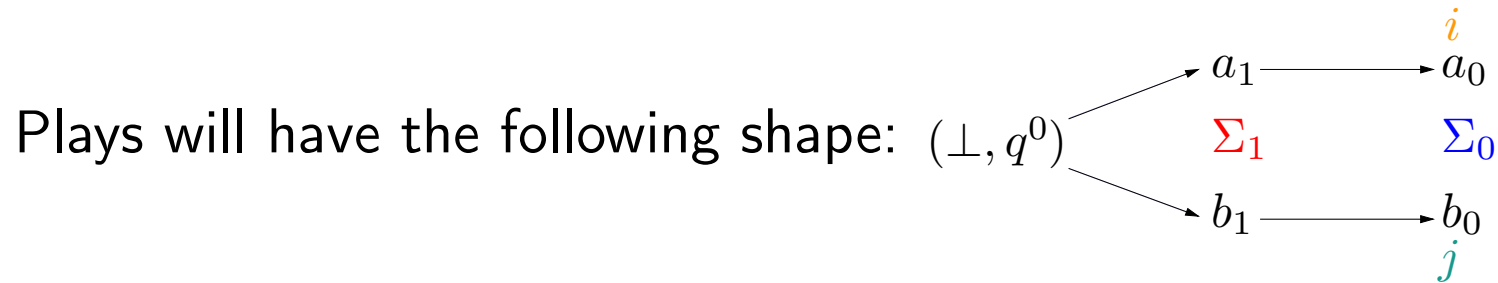
# Sequentialized game: example

2 processes 1, 2.  $\Sigma_i = \{a_i, b_i\}$ ,  $R(a_i) = W(a_i) = 1$  and  $R(b_i) = W(b_i) = 2$ .

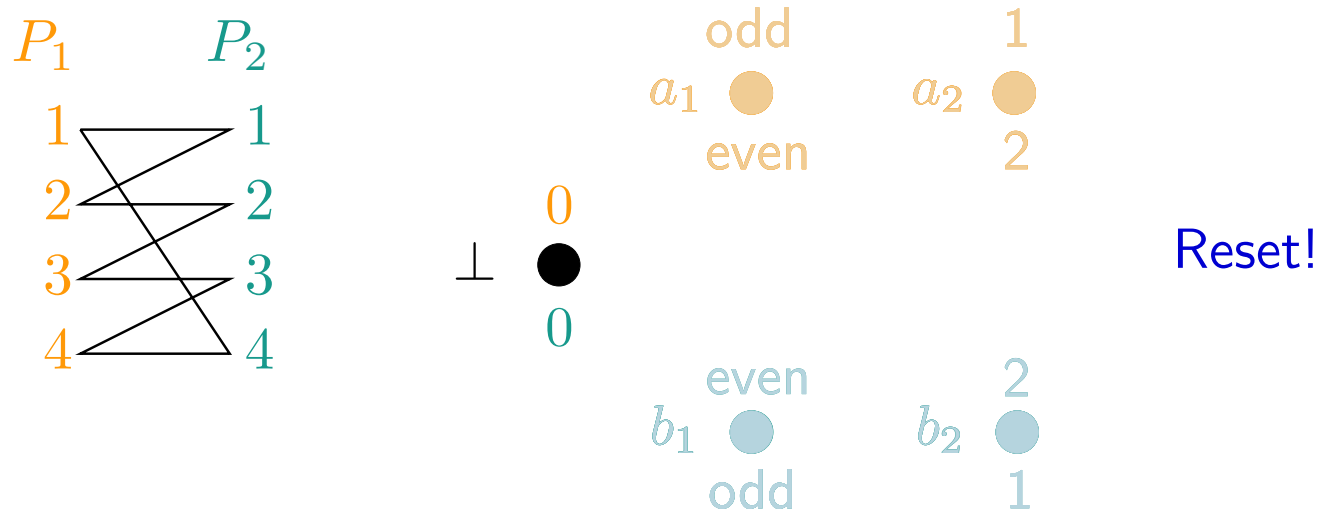
$\Sigma_1$ -transitions:  $q^0 \xrightarrow{a_1} \text{odd} \vee \text{even}$  and  $q^0 \xrightarrow{b_1} \text{odd} \vee \text{even}$

$\Sigma_0$ -transitions:  $\text{odd} \xrightarrow{a_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{a_0} 2 \vee 4$

$\text{odd} \xrightarrow{b_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{b_0} 2 \vee 4$



$$\mathcal{W} = \{(i, j) \mid i - j \equiv 0 \text{ or } 1 \pmod{4}\} \cup \text{Shape}_1$$



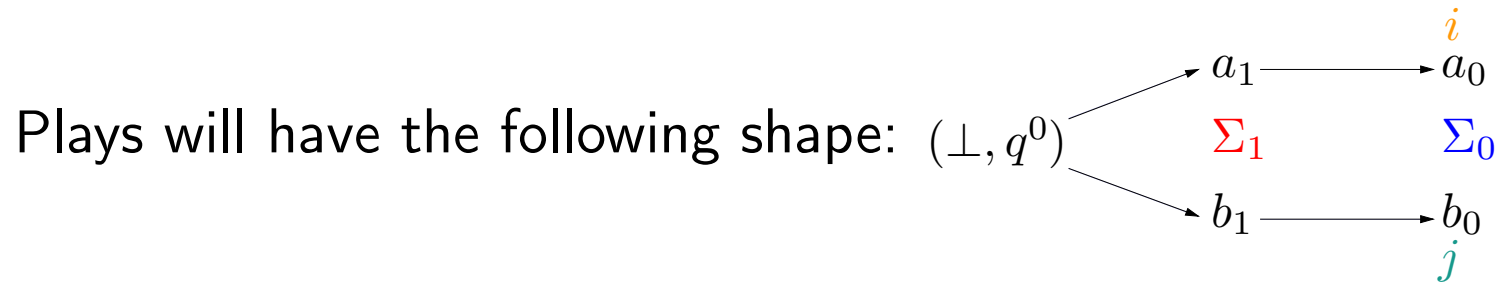
# Sequentialized game: example

2 processes 1, 2.  $\Sigma_i = \{a_i, b_i\}$ ,  $R(a_i) = W(a_i) = 1$  and  $R(b_i) = W(b_i) = 2$ .

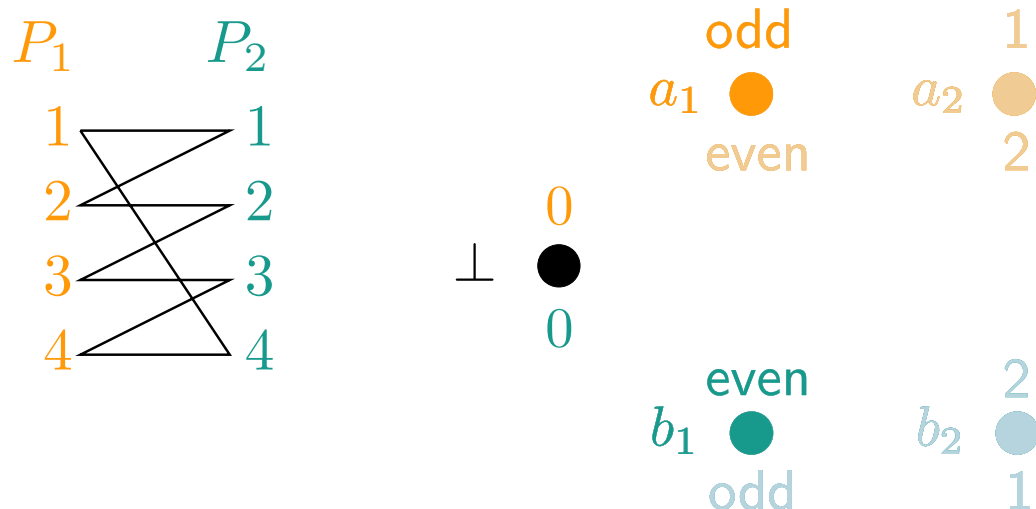
$\Sigma_1$ -transitions:  $q^0 \xrightarrow{a_1} \text{odd} \vee \text{even}$  and  $q^0 \xrightarrow{b_1} \text{odd} \vee \text{even}$

$\Sigma_0$ -transitions:  $\text{odd} \xrightarrow{a_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{a_0} 2 \vee 4$

$\text{odd} \xrightarrow{b_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{b_0} 2 \vee 4$



$$\mathcal{W} = \{(i, j) \mid i - j \equiv 0 \text{ or } 1 \pmod{4}\} \cup \text{Shape}_1$$



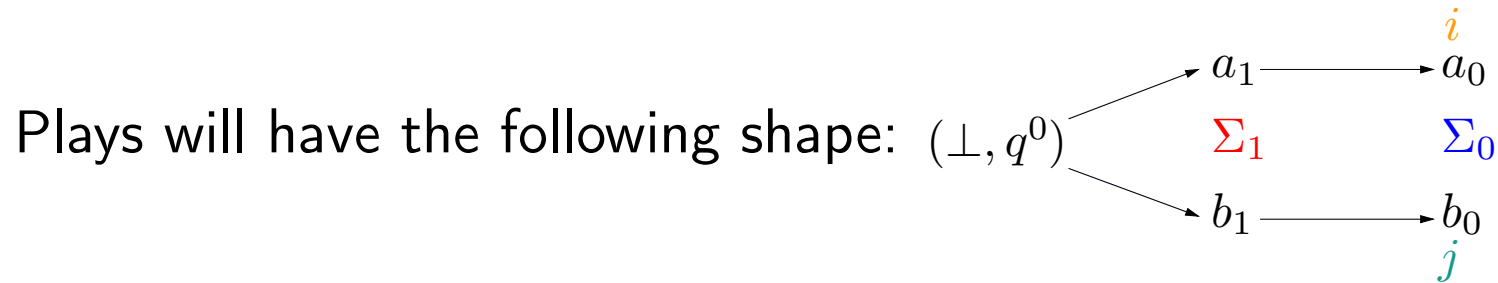
# Sequentialized game: example

2 processes 1, 2.  $\Sigma_i = \{a_i, b_i\}$ ,  $R(a_i) = W(a_i) = 1$  and  $R(b_i) = W(b_i) = 2$ .

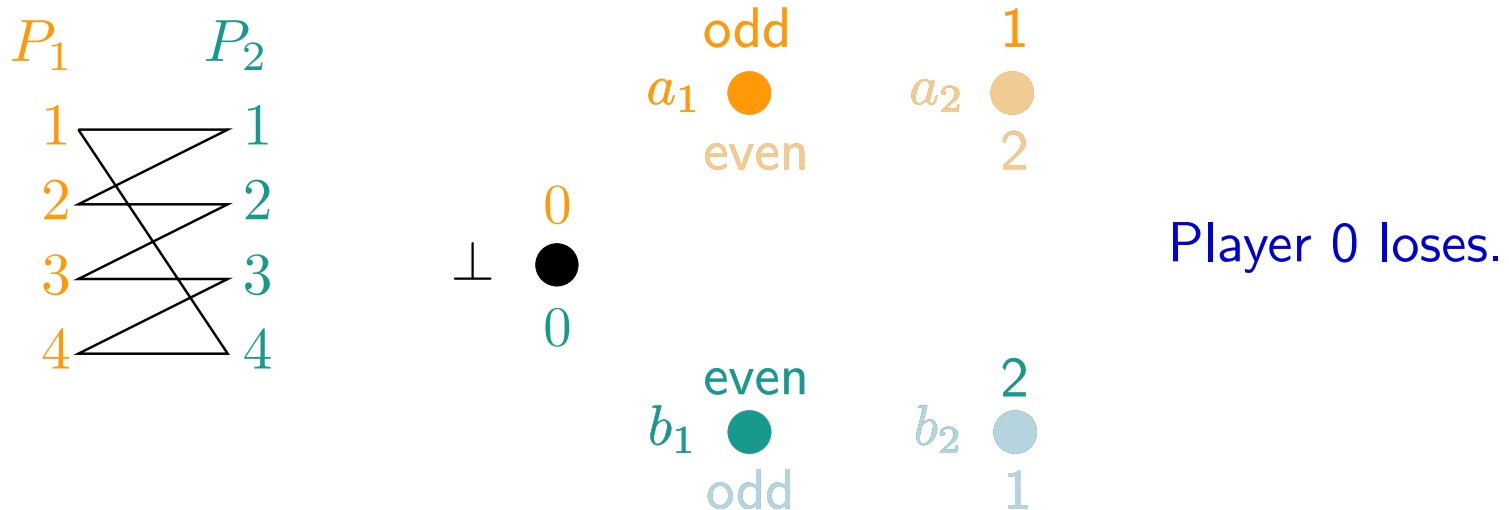
$\Sigma_1$ -transitions:  $q^0 \xrightarrow{a_1} \text{odd} \vee \text{even}$  and  $q^0 \xrightarrow{b_1} \text{odd} \vee \text{even}$

$\Sigma_0$ -transitions:  $\text{odd} \xrightarrow{a_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{a_0} 2 \vee 4$

$\text{odd} \xrightarrow{b_0} 1 \vee 3$ ,  $\text{even} \xrightarrow{b_0} 2 \vee 4$



$$\mathcal{W} = \{(i, j) \mid i - j \equiv 0 \text{ or } 1 \pmod{4}\} \cup \text{Shape}_1$$



# Open problems

- Given a game and a “simple” winning condition:
  - reachability,
  - safety,
  - recognizable.

is it decidable whether there is a causal winning strategy?

- We conjecture that the answer is yes.