Reactive Synthesis

Lecture 8

Swen Jacobs and Martin Zimmermann
(Saarland University)

Plan for Today

- Basic Games
- Algorithms & Data Structures
- Advanced Games

- Temporal Logic Synthesis

Plan for Today

- Basic Games
- Algorithms & Data Structures
- Advanced Games
  - Büchi Games
  - Co-Büchi Games
  - Request-response Games and Reductions
- Temporal Logic Synthesis

Introduction

Four foundational winning conditions:

Reachability reaching a goal (at least once)

Safety staying safe (at all times)

Introduction

Four foundational winning conditions:

Reachability reaching a goal (at least once)

Safety staying safe (at all times)

Recurrence reaching a goal infinitely often

Introduction

Four foundational winning conditions:

Reachability reaching a goal (at least once)

Safety staying safe (at all times)

Recurrence reaching a goal infinitely often

Persistence staying safe from some point onwards

Introduction

Four foundational winning conditions:

Reachability reaching a goal (at least once)

Safety staying safe (at all times)

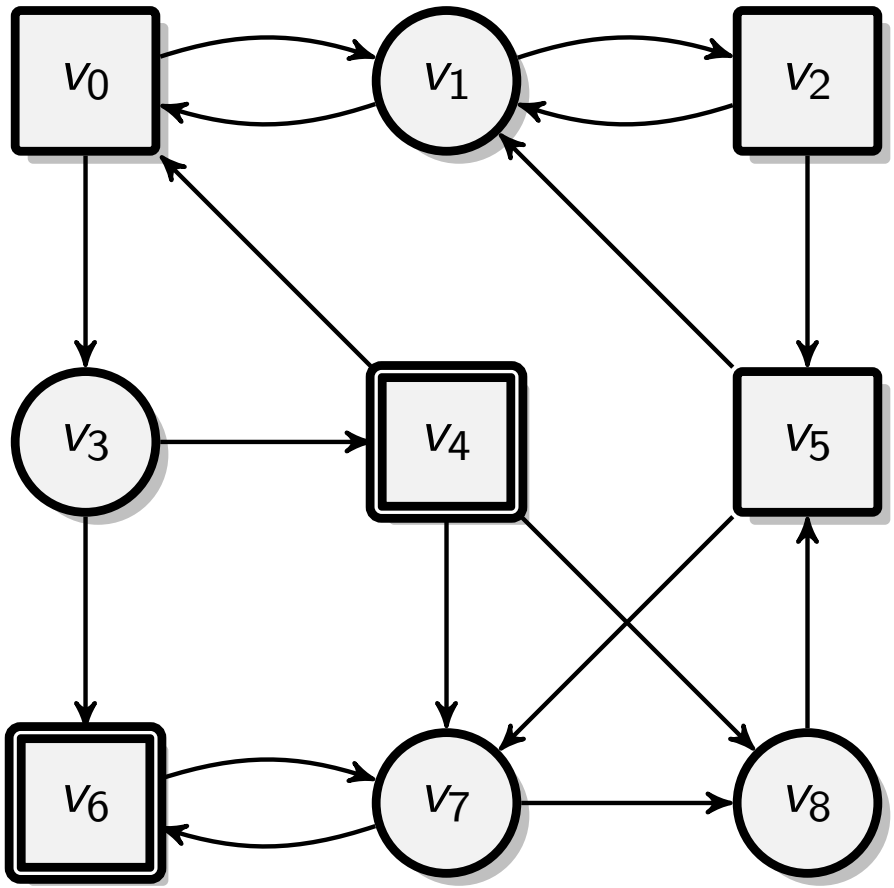
Recurrence reaching a goal infinitely often

Persistence staying safe from some point onwards

Remark

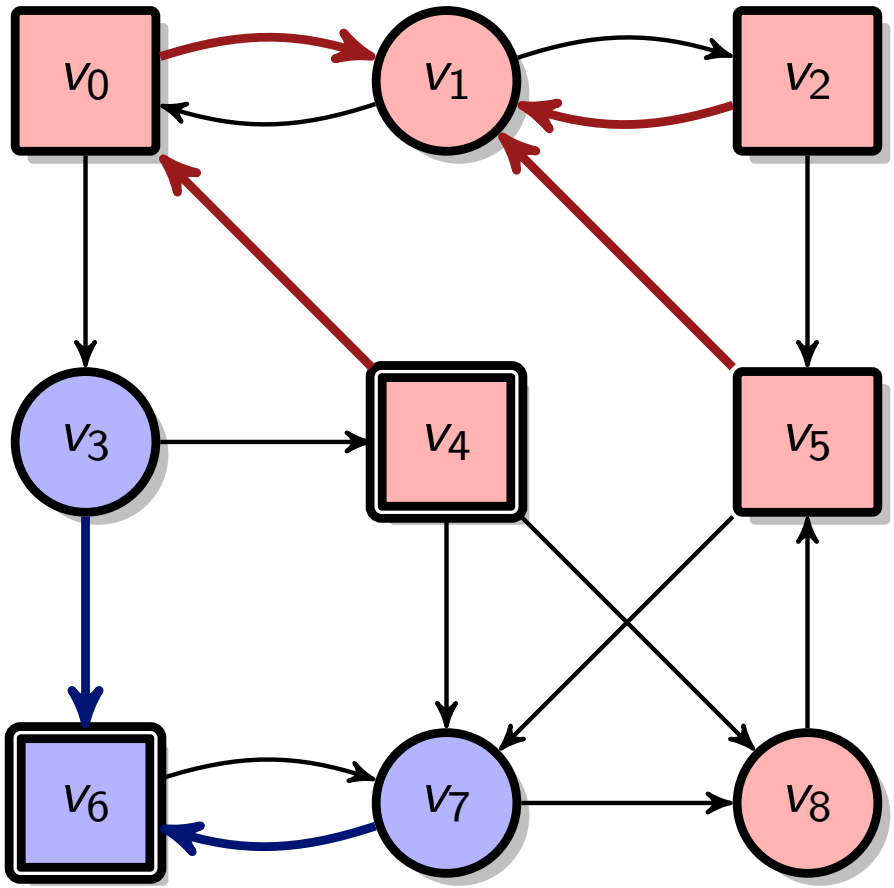
Recurrence conditions are typically called Büchi conditions after J. Richard Büchi, who introduced automata on infinite words with recurrence acceptance conditions (nowadays called Büchi automata).

Büchi Games



Player 0 wins a play if at least one doubly-lined vertex is visited infinitely often

Büchi Games



Player 0 wins a play if at least one doubly-lined vertex is visited infinitely often

Büchi Games

Notation

$\mathrm{Inf}(\rho) := \{v \in V \mid v_n = v \text{ for infinitely many } n\}$: the set of vertices occurring infinitely often in ρ.

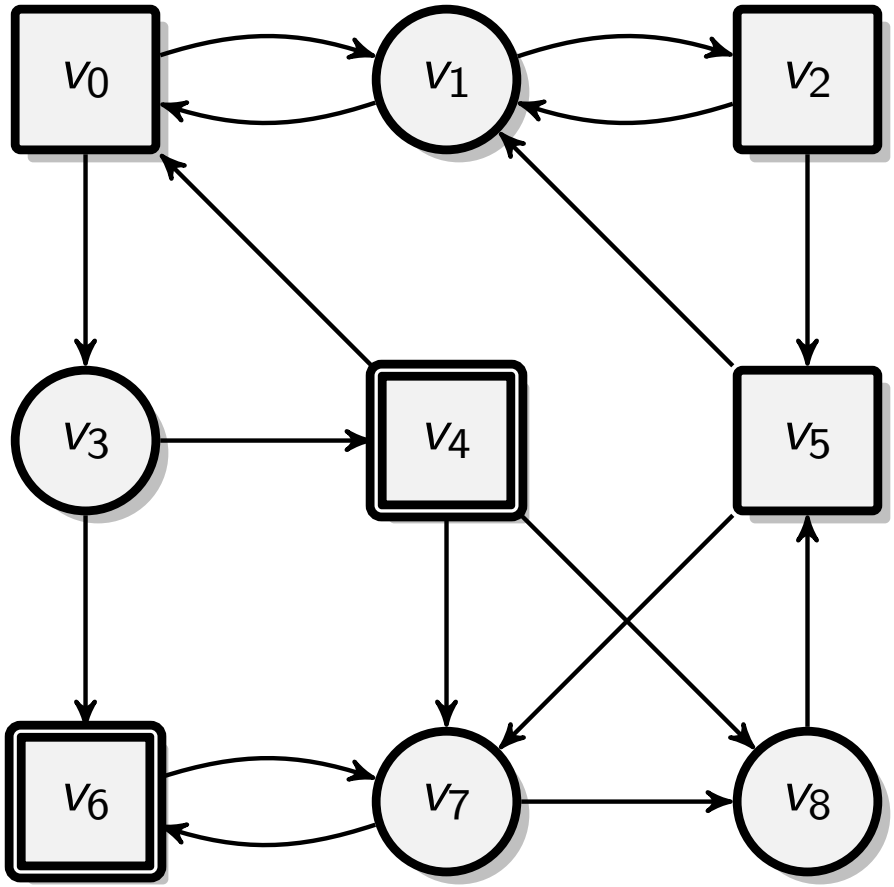
Definition

Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena and let $F \subseteq V$ be a subset of $\mathcal{A}$'s vertices. Then, the Büchi condition $\textsc{Büchi}(F)$ is defined as

$$\textsc{Büchi}(F) := \{\rho \in V^\omega \mid \mathrm{Inf}(\rho) \cap F \neq \emptyset\}.$$

We call a game $\mathcal{G} = (\mathcal{A}, \textsc{Büchi}(F))$ a *Büchi game*.

Back to the Example



Recurrence Construction

Recall

- $\mathrm{CPre}_i(R)$: the set of vertices from which Player i can enforce to reach R in one move.
- $\mathrm{Attr}_i(R)$: the set of vertices from which Player i can enforce to reach R in any number of moves.

Recurrence Construction

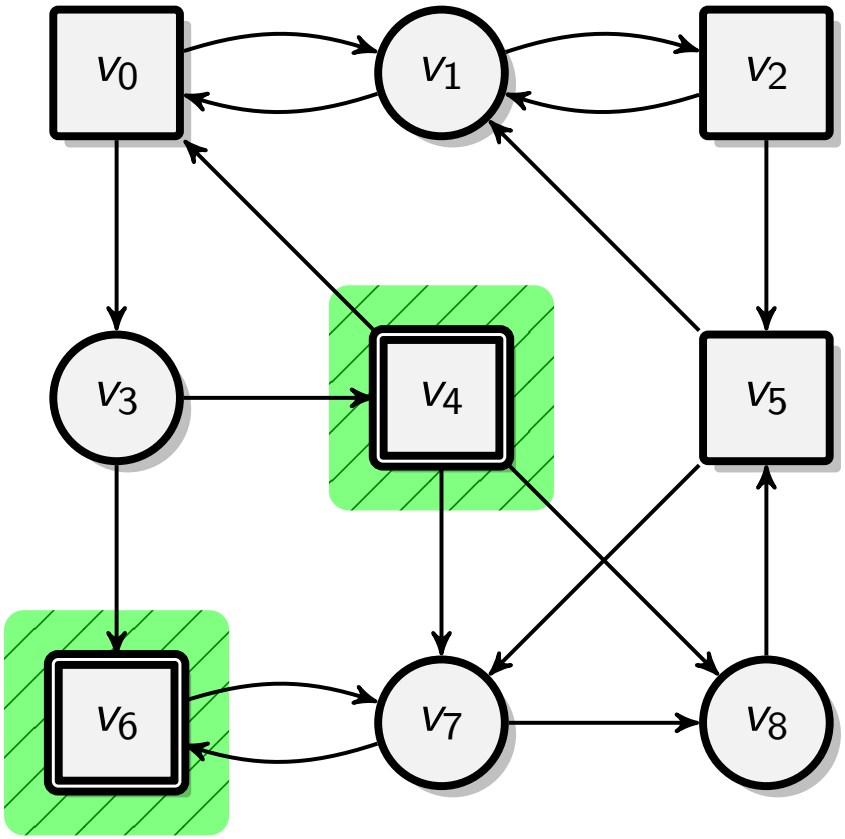
Recall

- $\mathrm{CPre}_i(R)$: the set of vertices from which Player i can enforce to reach R in one move.
- $\mathrm{Attr}_i(R)$: the set of vertices from which Player i can enforce to reach R in any number of moves.

Let $\mathcal{A} = (V, V_0, V_1, E)$ be arena and let $F \subseteq V$. The *recurrence construction* for Player i is defined inductively as

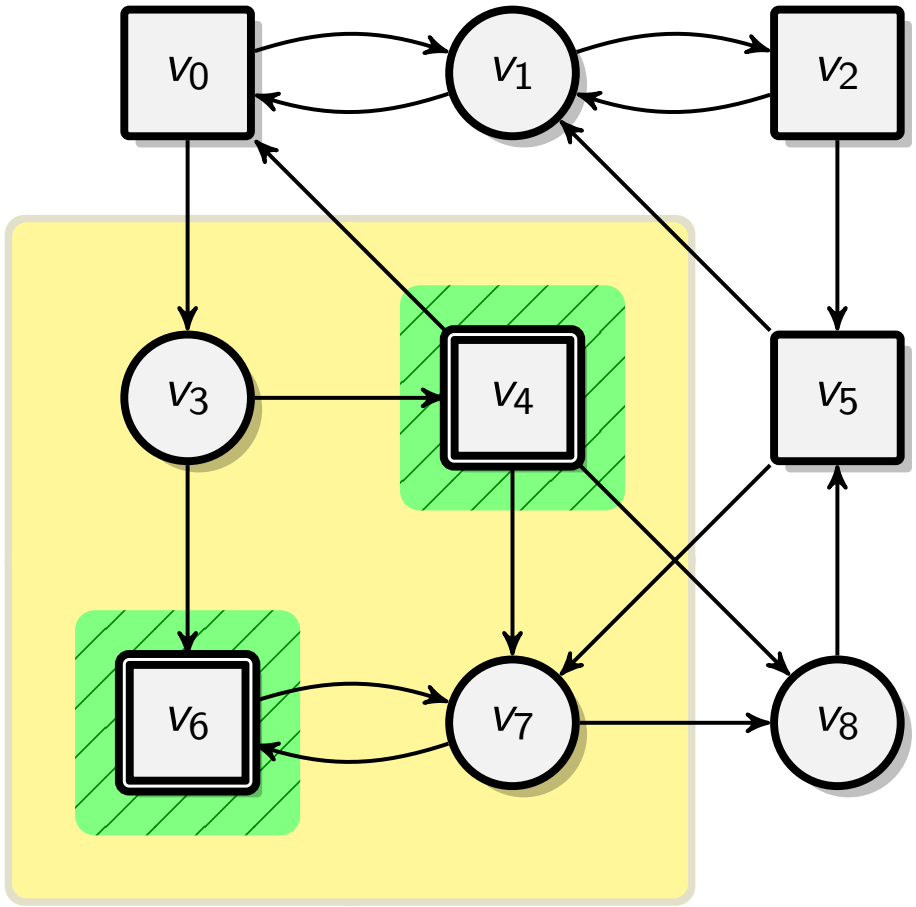
- $F^0 = F$,
- $W_1^n = V \setminus \mathrm{Attr}_0(F^n)$ for every $n \geq 0$, and
- $F^{n+1} = F \setminus \mathrm{CPre}_1(W_1^n)$ for every $n \geq 0$.

An Example



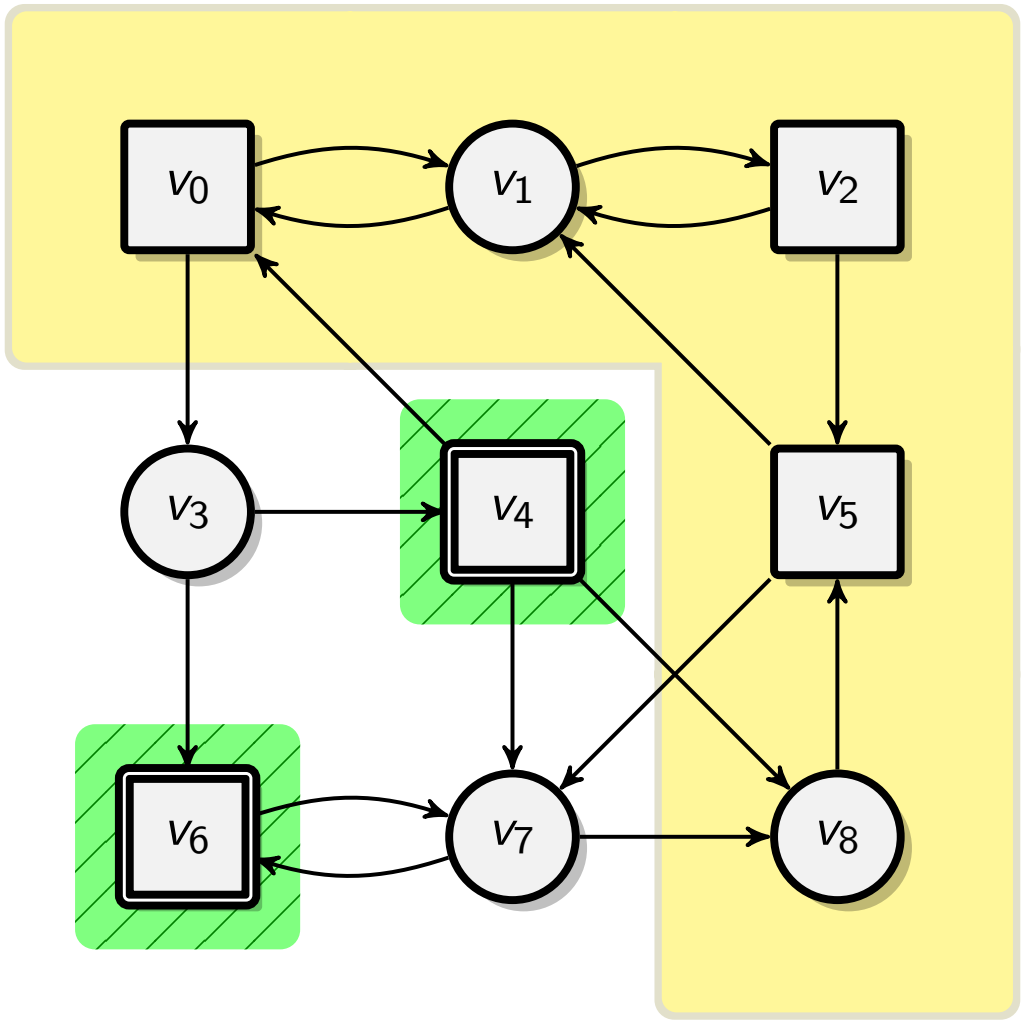
$$F^0 = F = \{v_4, v_6\}$$

An Example



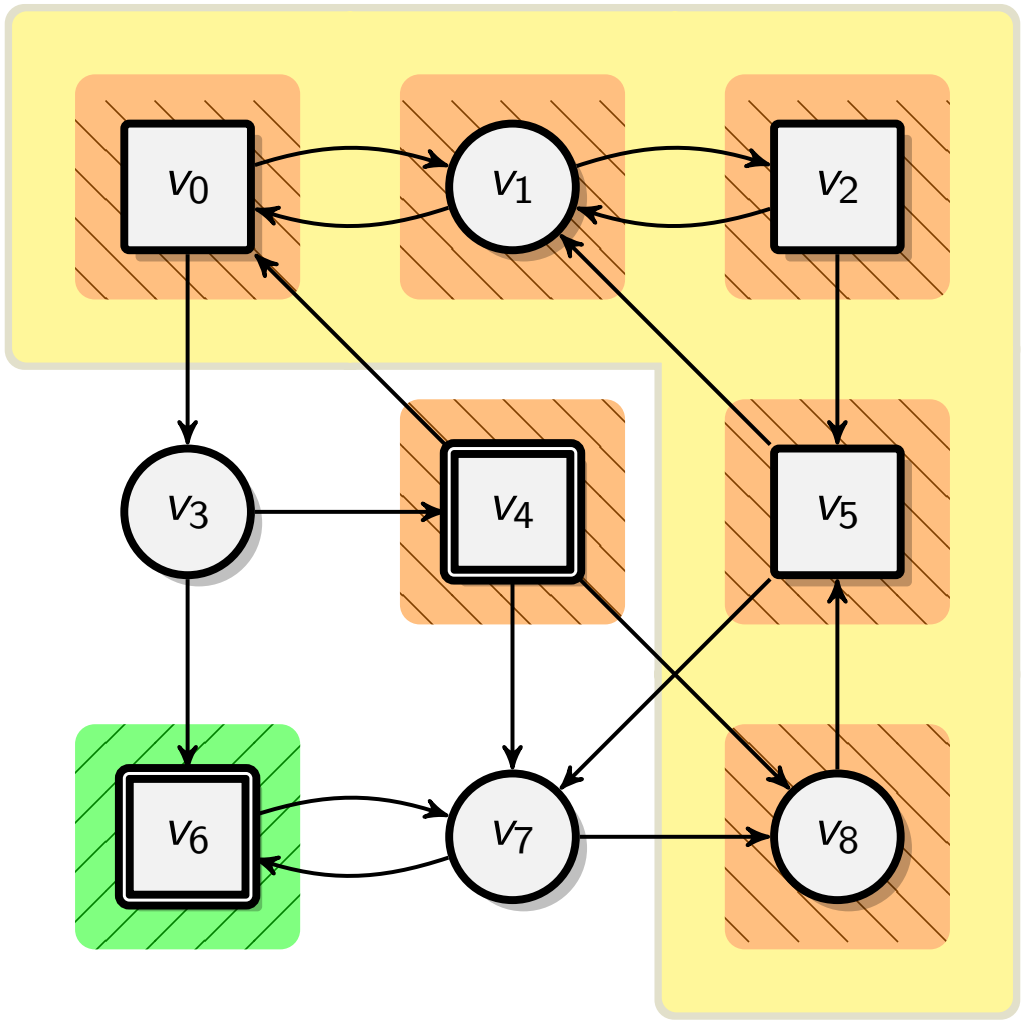
$$\mathrm{Attr}_0(\{v_4, v_6\}) = \{v_3, v_4, v_6, v_7\}$$

An Example



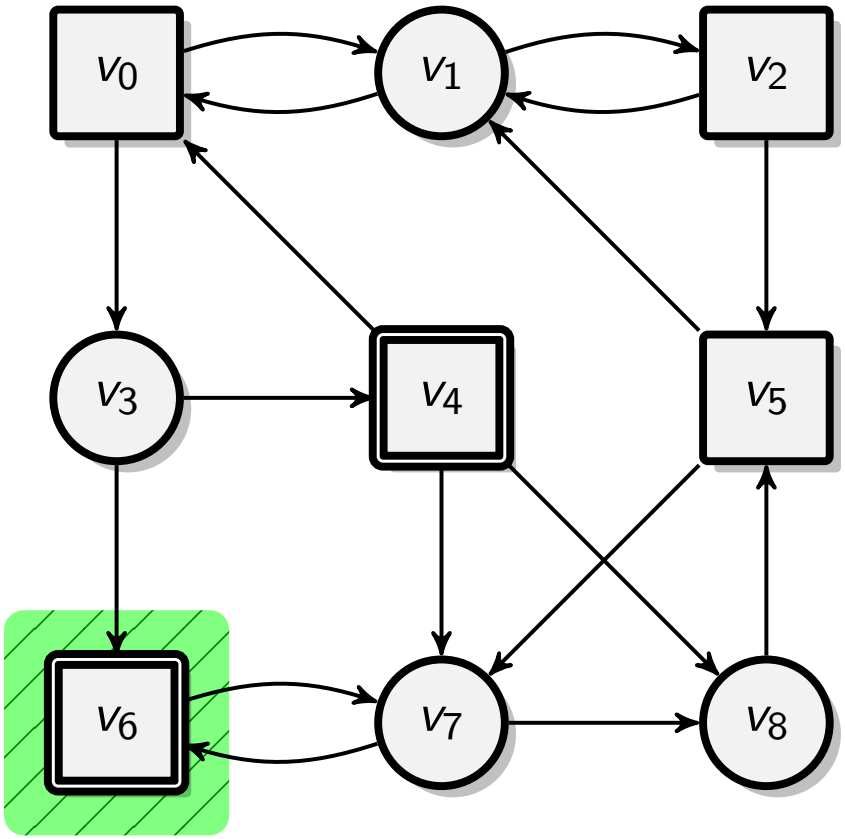
$$W_1^0 = \{v_0, v_1, v_2, v_5, v_8\}$$

An Example



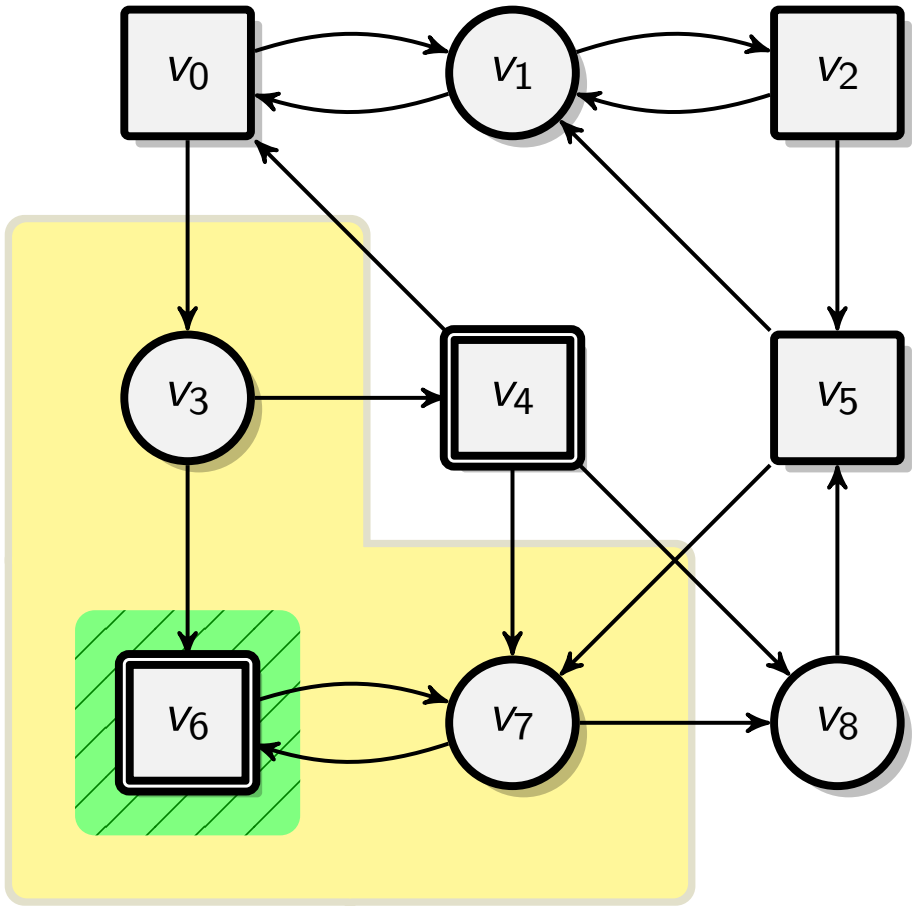
$$\mathrm{CPre}_1(\{v_0, v_1, v_2, v_5, v_8\}) = \{v_0, v_1, v_2, v_4, v_5, v_8\}$$

An Example



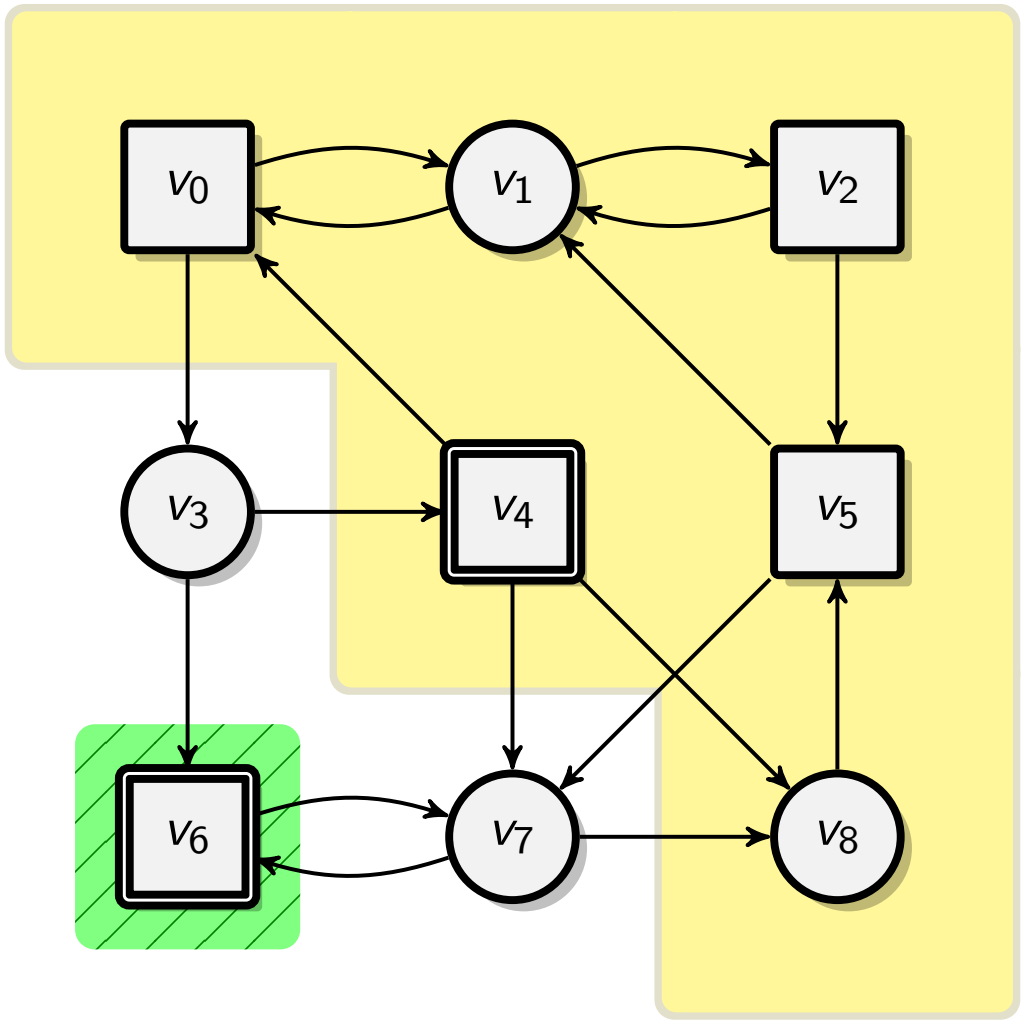
$$F^1 = \{v_6\}$$

An Example



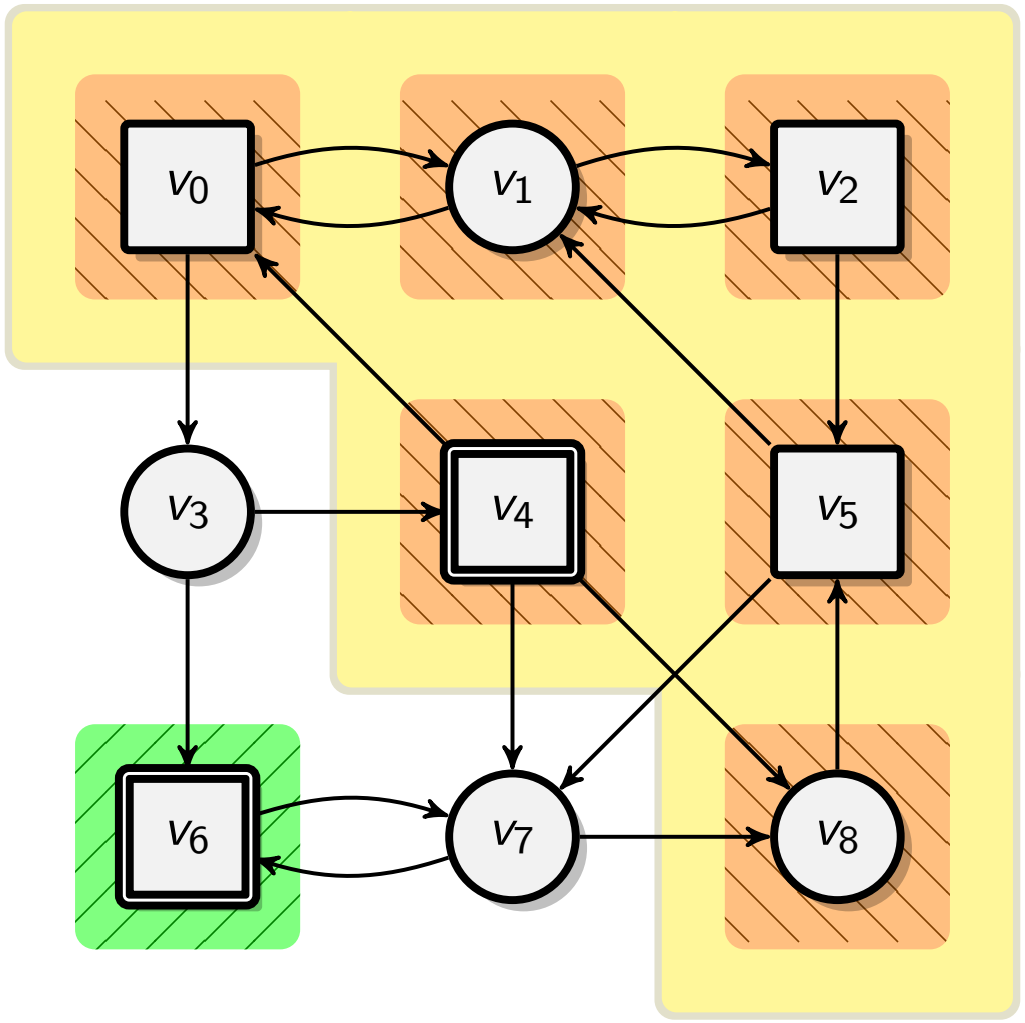
$$\mathrm{Attr}_0(\{v_6\}) = \{v_3, v_6, v_7\}$$

An Example



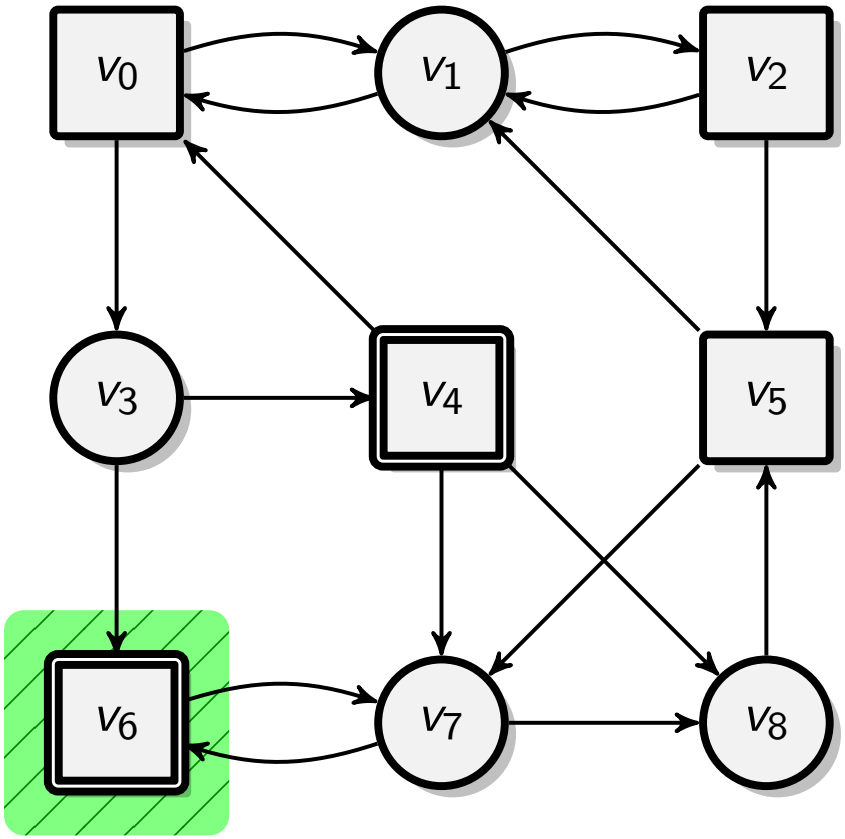
$$W_1^2 = \{v_0, v_1, v_2, v_4, v_5, v_8\}$$

An Example



$$\mathrm{CPre}_1(\{v_0, v_1, v_2, v_4, v_5, v_8\}) = \{v_0, v_1, v_2, v_4, v_5, v_8\}$$

An Example



$$F^2 = \{v_6\} = F^1$$

Results

Lemma

1. $F = F^0 \supseteq F^1 \supseteq \cdots \supseteq F^m = F^{m+1}$ *and*
2. $W_1^0 \subseteq W_1^1 \subseteq \cdots \subseteq W_1^m = W_1^{m+1}$

for some $m \leq |V|$.

Results

Lemma

1. $F = F^0 \supseteq F^1 \supseteq \cdots \supseteq F^m = F^{m+1}$ *and*
2. $W_1^0 \subseteq W_1^1 \subseteq \cdots \subseteq W_1^m = W_1^{m+1}$

for some $m \leq |V|$.

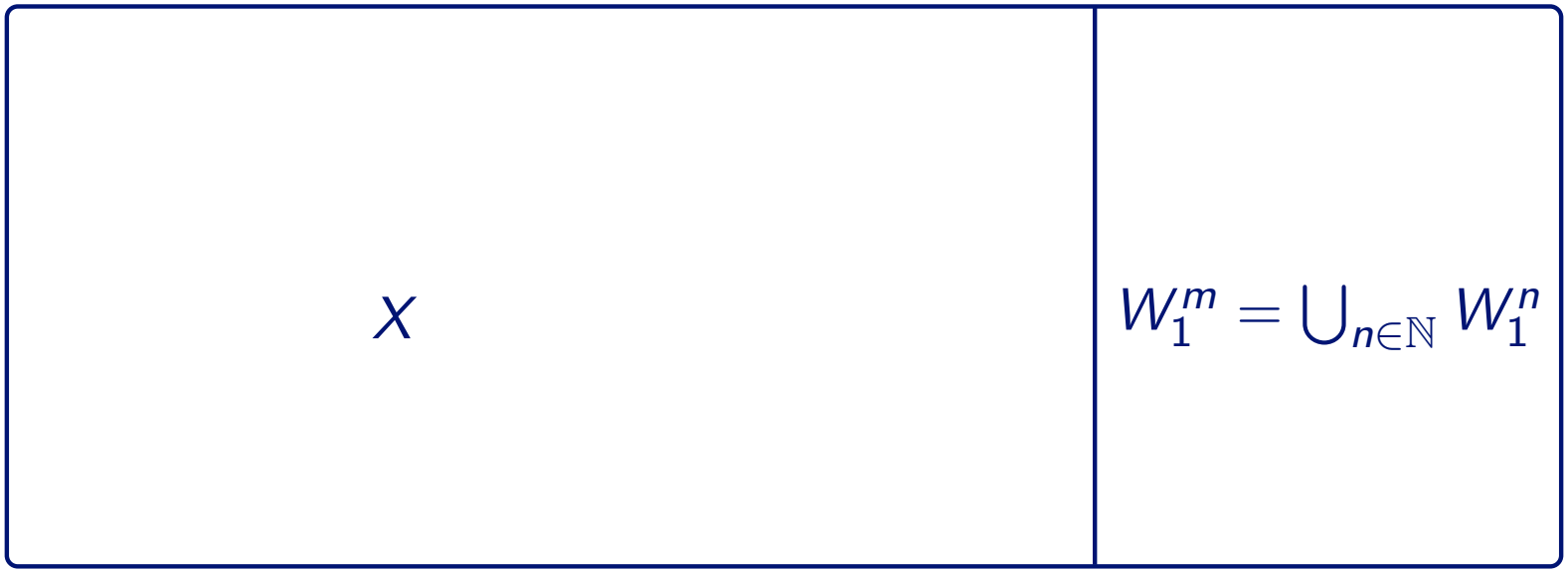
Lemma

Let $\mathcal{G} = (\mathcal{A}, \textsc{Büchi}(F))$ *be a Büchi game. Then,* $W_1(\mathcal{G}) = \bigcup_{n \in \mathbb{N}} W_1^n$ *and* $W_0(\mathcal{G}) = V \setminus W_1(\mathcal{G})$.

Proof Sketch

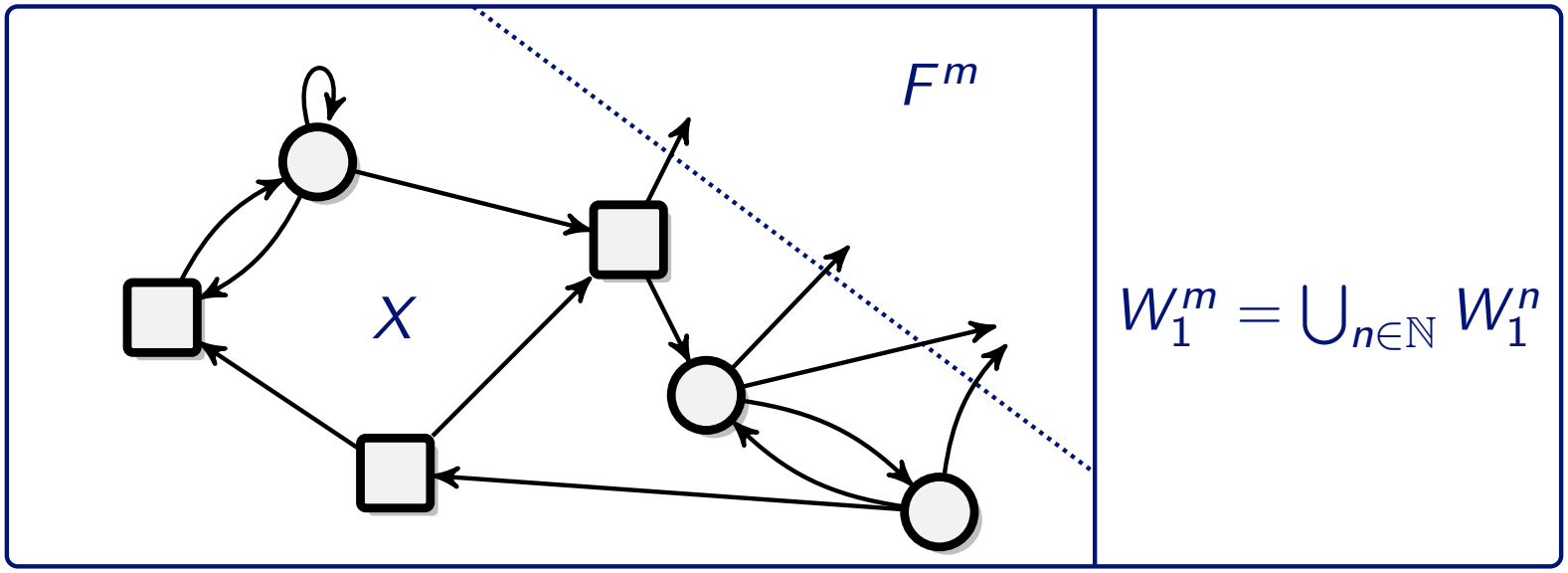
X	$W_1^m = \bigcup_{n \in \mathbb{N}} W_1^n$

Proof Sketch



First show $X \subseteq W_0(\mathcal{G})$:

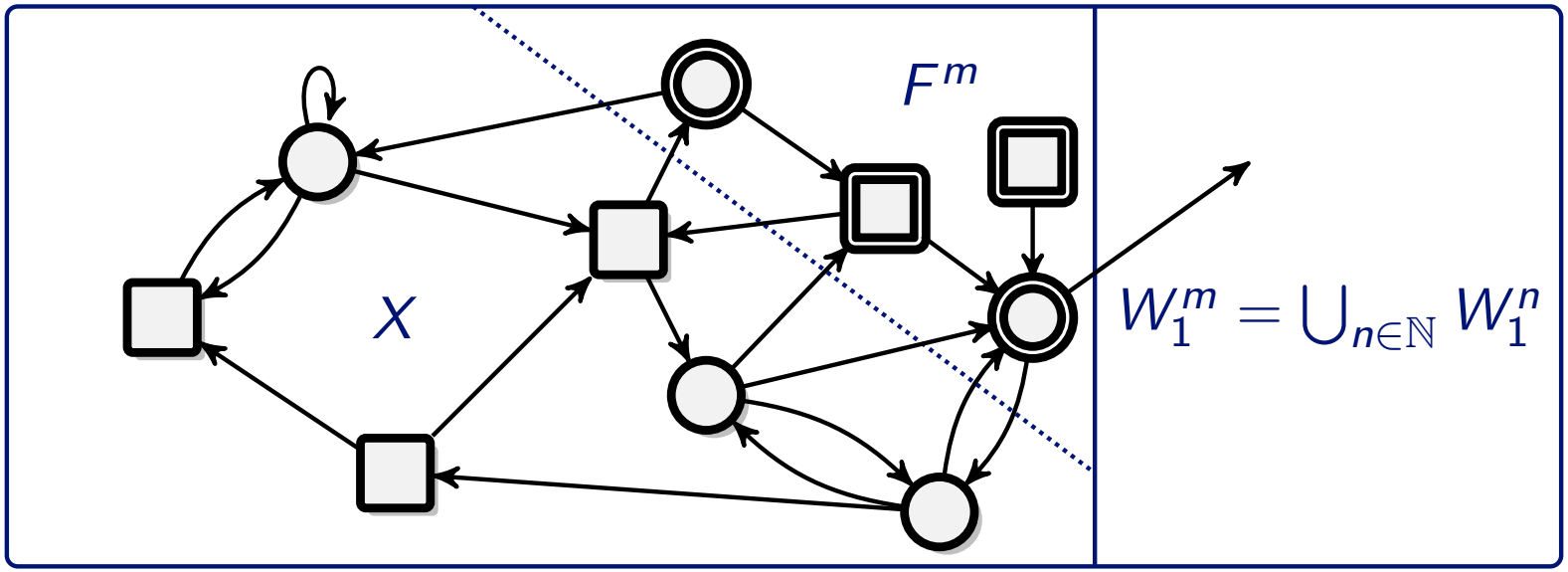
Proof Sketch



First show $X \subseteq W_0(\mathcal{G})$:

- By construction: $X = \mathrm{Attr}_0(F^m)$

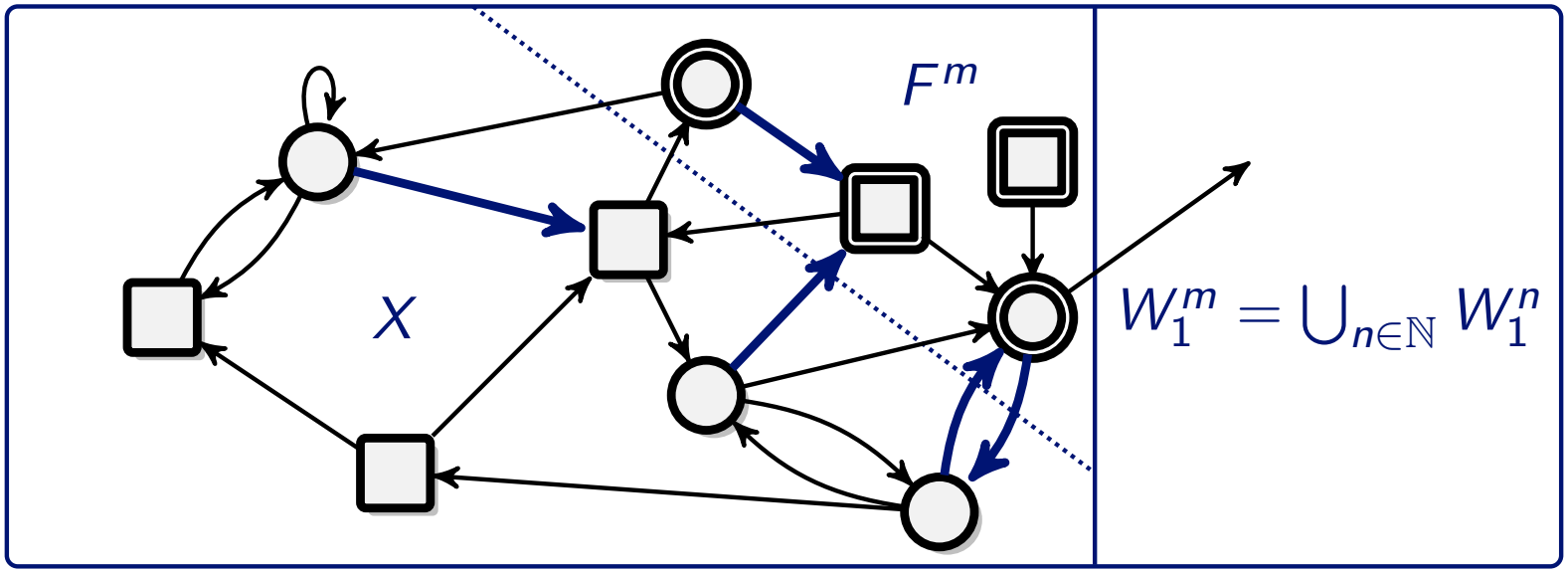
Proof Sketch



First show $X \subseteq W_0(\mathcal{G})$:

- By construction: $X = \mathrm{Attr}_0(F^m)$ and $F^m \subseteq \mathrm{CPre}_0(X)$.

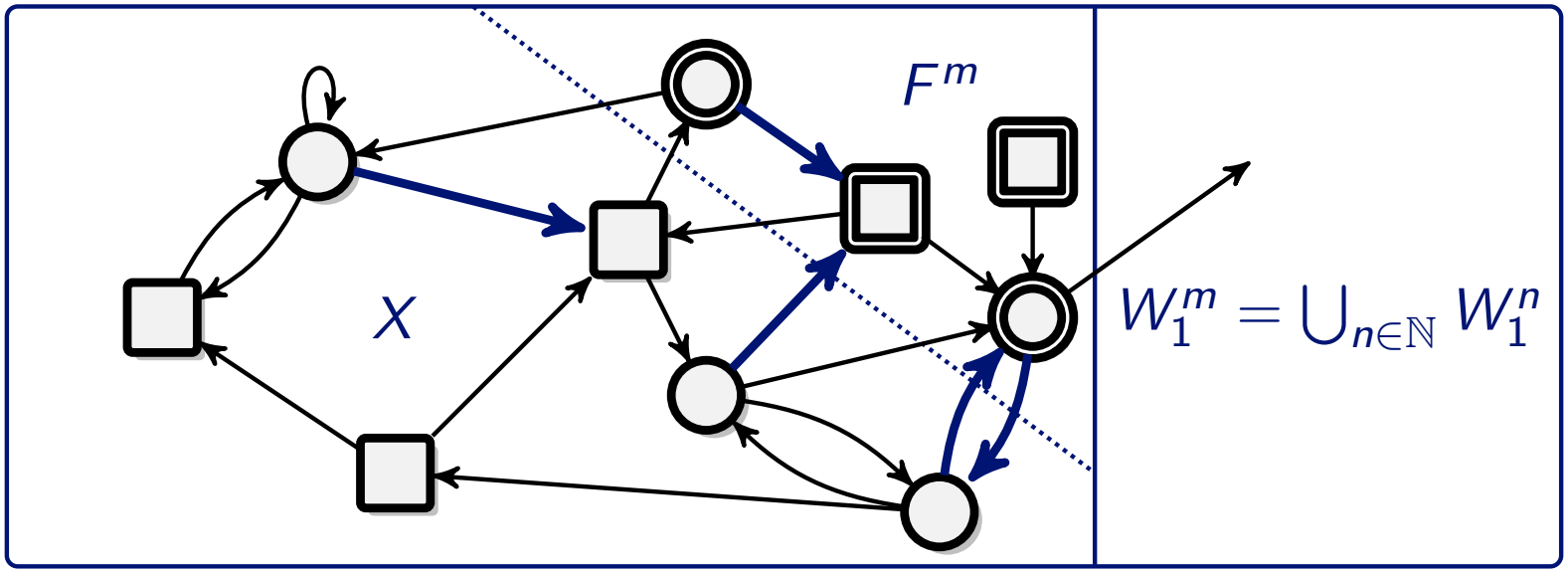
Proof Sketch



First show $X \subseteq W_0(\mathcal{G})$:

- By construction: $X = \mathrm{Attr}_0(F^m)$ and $F^m \subseteq \mathrm{CPre}_0(X)$.
- Consider the following strategy for Player 0 on X:
  - In $F^m \cap V_0$, move to X.
  - In $X \setminus F^m$, use attractor strategy to reach F^m.

Proof Sketch

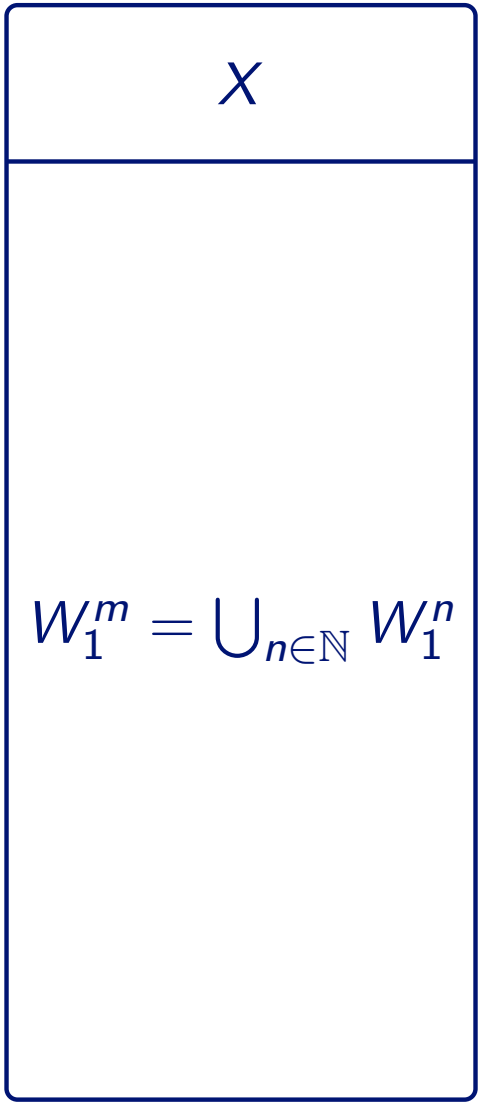


First show $X \subseteq W_0(\mathcal{G})$:

- By construction: $X = \mathrm{Attr}_0(F^m)$ and $F^m \subseteq \mathrm{CPre}_0(X)$.
- Consider the following strategy for Player 0 on X:
  - In $F^m \cap V_0$, move to X.
  - In $X \setminus F^m$, use attractor strategy to reach F^m.
- Every play starting in X and consistent with the strategy visits $F^m \subseteq F$ infinitely often.

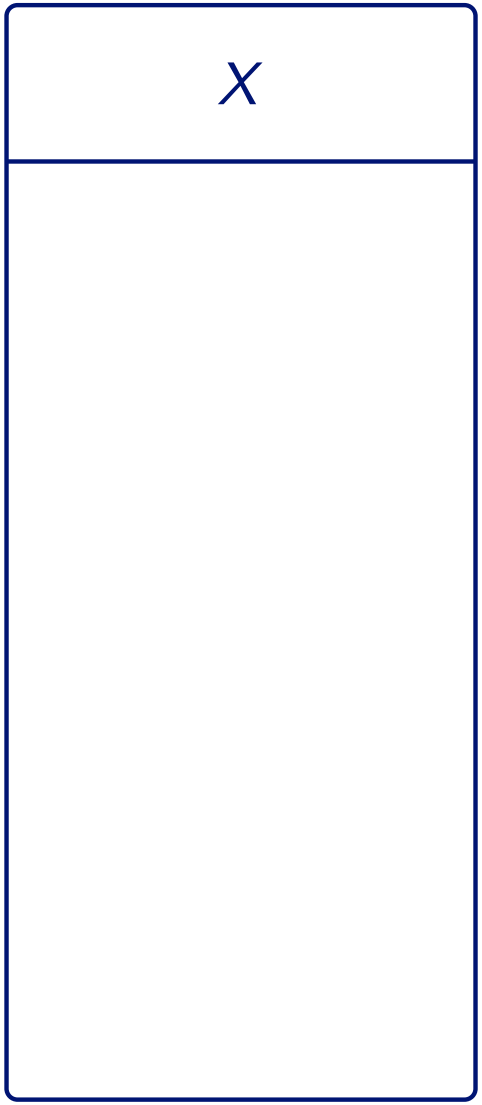
Proof Sketch

Now show $V \setminus X \subseteq W_1(\mathcal{G})$:



Proof Sketch

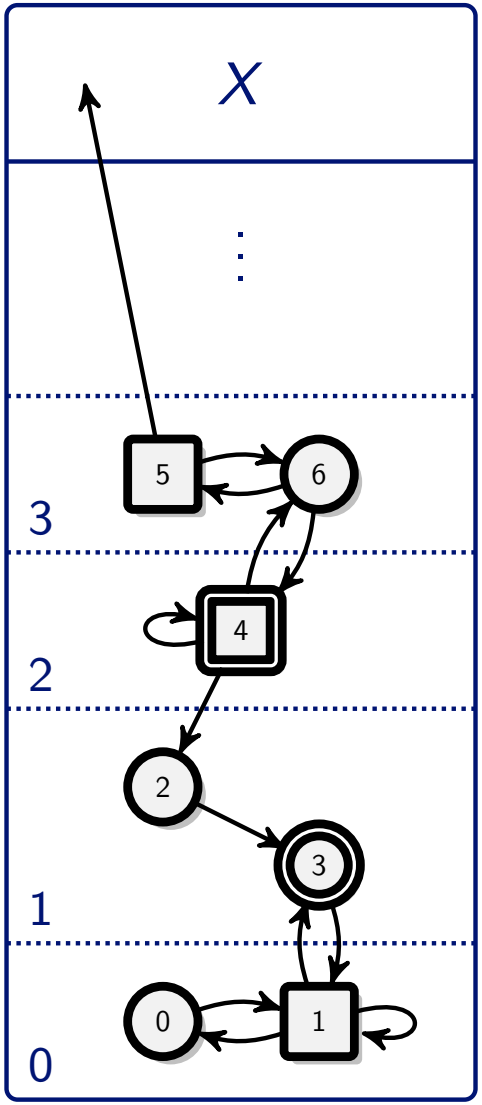
Now show $V \setminus X \subseteq W_1(\mathcal{G})$:



- Define $\delta(v) = \min\{n \mid v \in W_1^n\}$ for $v \in W_1^m$.

Proof Sketch

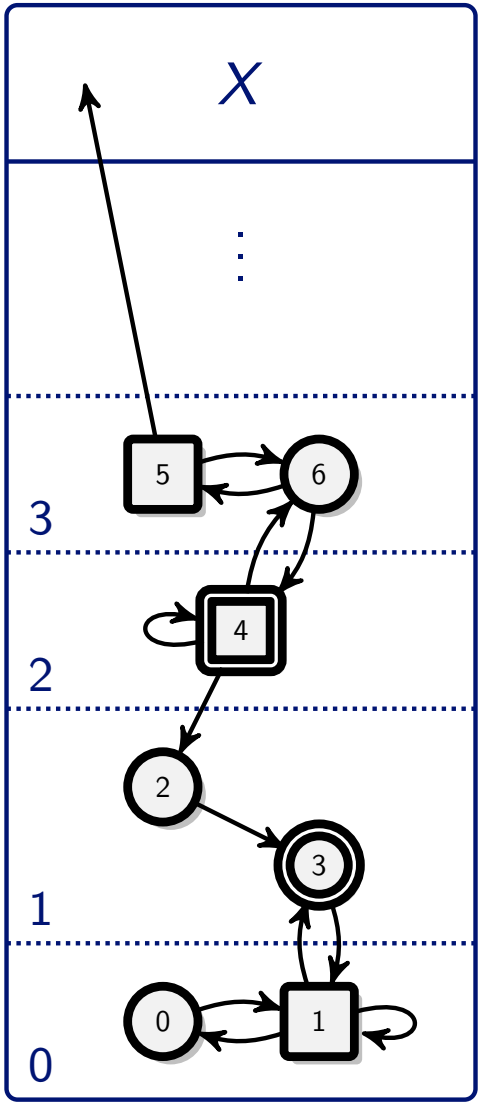
Now show $V \setminus X \subseteq W_1(\mathcal{G})$:



- Define $\delta(v) = \min\{n \mid v \in W_1^n\}$ for $v \in W_1^m$.

Proof Sketch

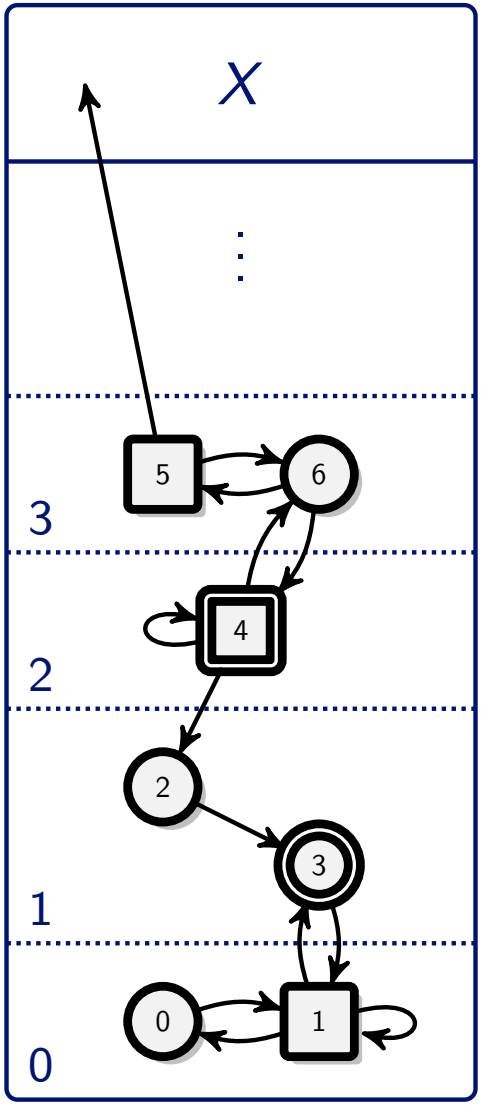
Now show $V \setminus X \subseteq W_1(\mathcal{G})$:



- Define $\delta(v) = \min\{n \mid v \in W_1^n\}$ for $v \in W_1^m$. Then:

Proof Sketch

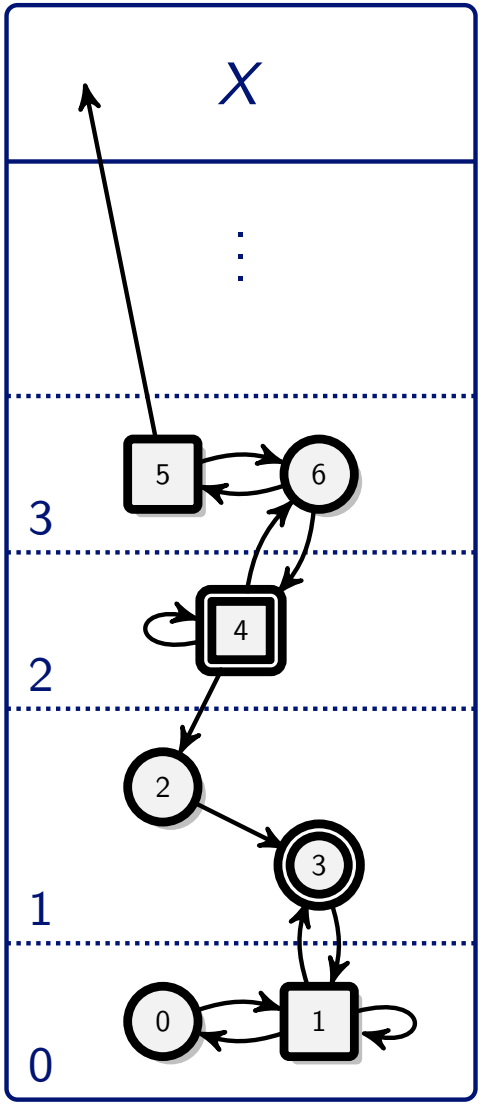
Now show $V \setminus X \subseteq W_1(\mathcal{G})$:



- Define $\delta(v) = \min\{n \mid v \in W_1^n\}$ for $v \in W_1^m$. Then:
  1. If $v \in F$, then $\delta(v) > 0$.

Proof Sketch

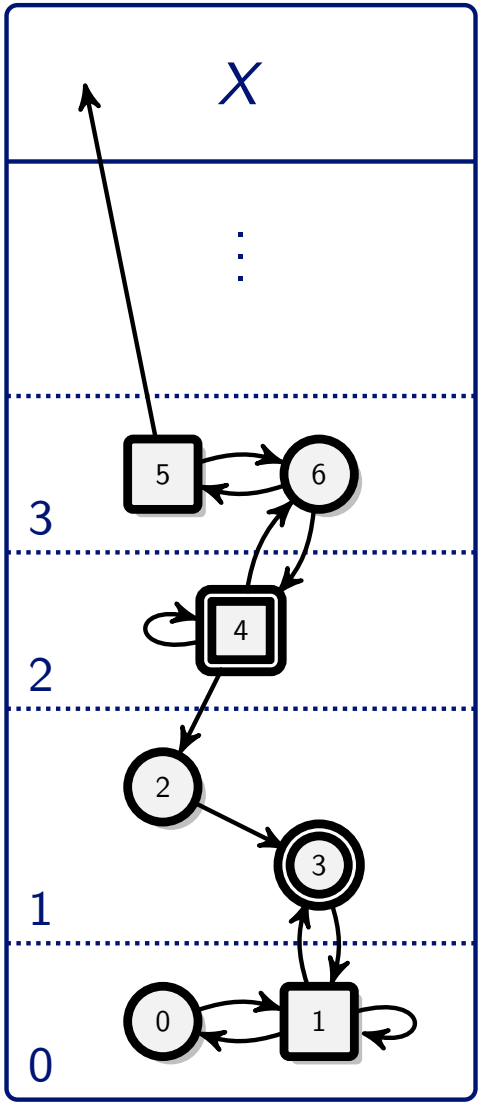
Now show $V \setminus X \subseteq W_1(\mathcal{G})$:



- Define $\delta(v) = \min\{n \mid v \in W_1^n\}$ for $v \in W_1^m$. Then:
  1. If $v \in F$, then $\delta(v) > 0$.
  2. If $v \in V_1$, then some successor v' of v satisfies $\delta(v) \geq \delta(v')$. Further, if $v \in F$, then even $\delta(v) > \delta(v')$.

Proof Sketch

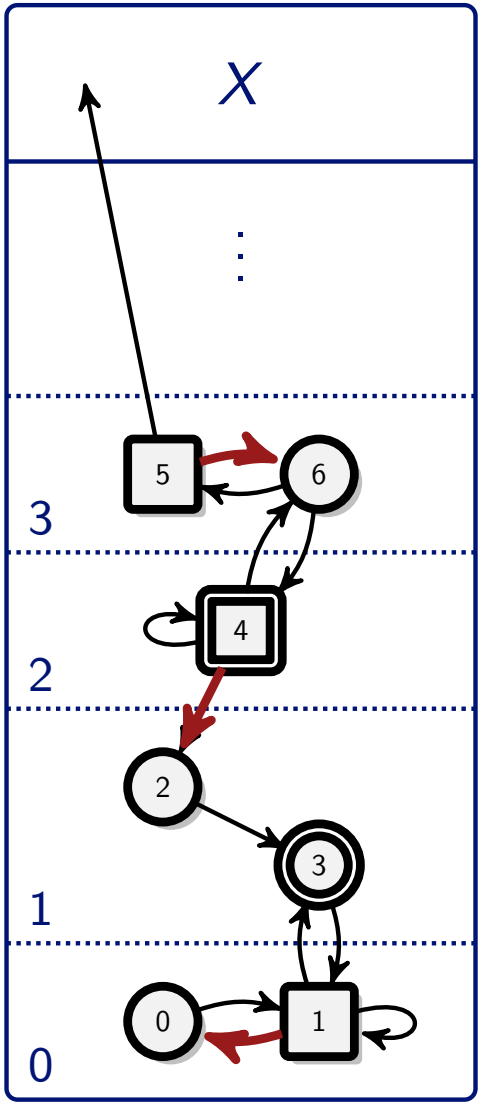
Now show $V \setminus X \subseteq W_1(\mathcal{G})$:



- Define $\delta(v) = \min\{n \mid v \in W_1^n\}$ for $v \in W_1^m$. Then:
  1. If $v \in F$, then $\delta(v) > 0$.
  2. If $v \in V_1$, then some successor v' of v satisfies $\delta(v) \geq \delta(v')$. Further, if $v \in F$, then even $\delta(v) > \delta(v')$.
  3. If $v \in V_0$, then all successors v' of v satisfy $\delta(v) \geq \delta(v')$. Further, if $v \in F$, then even $\delta(v) > \delta(v')$.

Proof Sketch

Now show $V \setminus X \subseteq W_1(\mathcal{G})$:



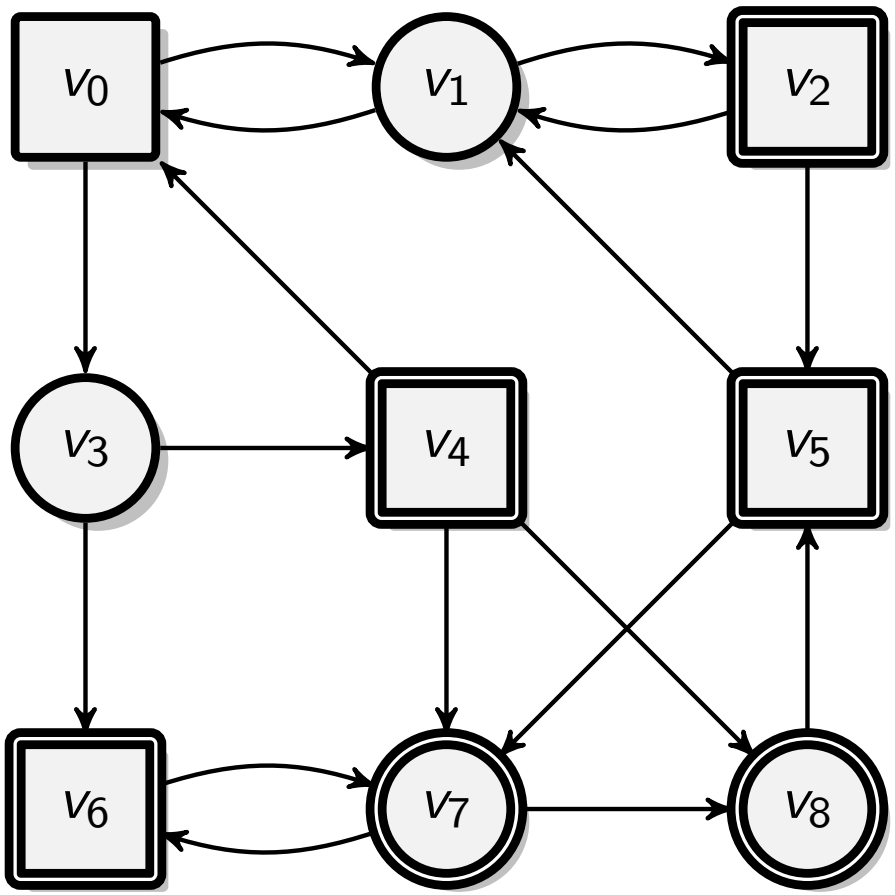
- Define $\delta(v) = \min\{n \mid v \in W_1^n\}$ for $v \in W_1^m$. Then:
  1. If $v \in F$, then $\delta(v) > 0$.
  2. If $v \in V_1$, then some successor v' of v satisfies $\delta(v) \geq \delta(v')$. Further, if $v \in F$, then even $\delta(v) > \delta(v')$.
  3. If $v \in V_0$, then all successors v' of v satisfy $\delta(v) \geq \delta(v')$. Further, if $v \in F$, then even $\delta(v) > \delta(v')$.
- Thus, always using a successor as guaranteed by **2.** yields a play where the δ-value never increases, but is decreased with every visit to F. Hence, F is only visited finitely often.

Main Theorem

Theorem

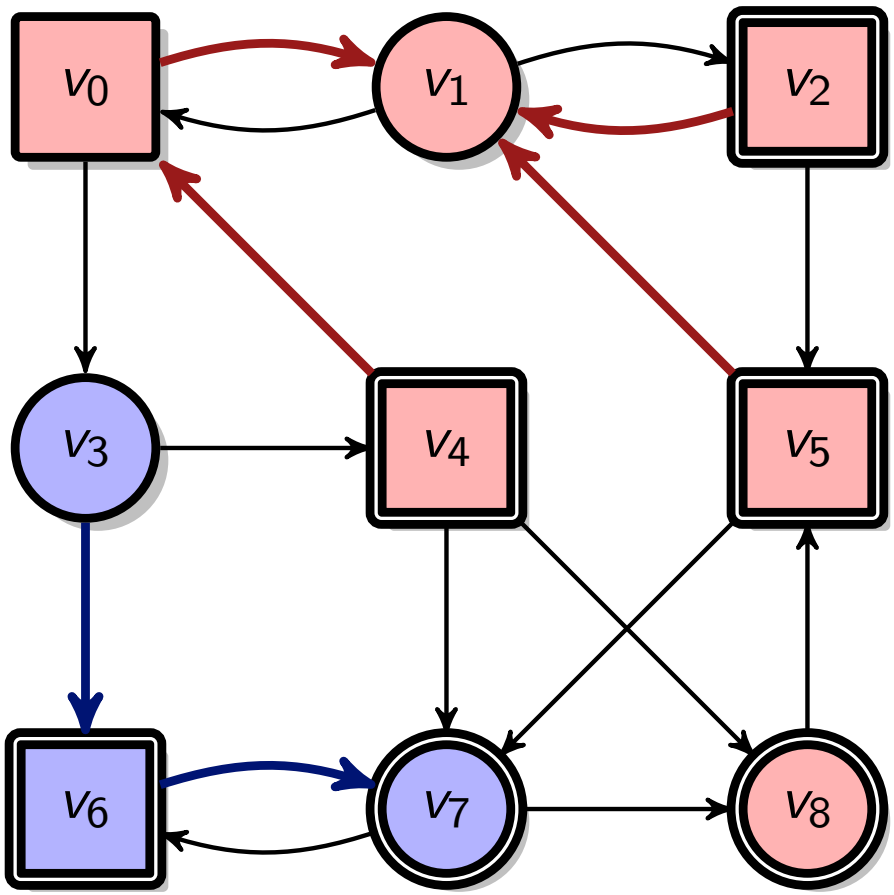
Büchi games are determined with positional winning strategies. They can be solved in polynomial time in the number of edges of the underlying arena.

Co-Büchi Games



Player 0 wins a play if from some point onwards only doubly-lined vertices are visited.

Co-Büchi Games



Player 0 wins a play if from some point onwards only doubly-lined vertices are visited.

Co-Büchi Games

Recall: $\mathrm{Inf}(\rho)$ is the set of vertices appearing infinitely often in ρ.

Definition

Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena and let $C \subseteq V$ be a subset of $\mathcal{A}$'s vertices. Then, the co-Büchi condition $\textsc{coBüchi}(C)$ is defined as

$$\textsc{coBüchi}(C) := \{\rho \in V^\omega \mid \mathrm{Inf}(\rho) \subseteq C\}.$$

We call a game $\mathcal{G} = (\mathcal{A}, \textsc{coBüchi}(C))$ a *co-Büchi game*.

Recap: Duality

Definition

Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena and let $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$.

1. The *dual arena* $\overline{\mathcal{A}}$ of $\mathcal{A}$ is defined as $\overline{\mathcal{A}} = (V, V_1, V_0, E)$.
2. The *dual game* $\overline{\mathcal{G}}$ of $\mathcal{G}$ is defined as $\overline{\mathcal{G}} = (\overline{\mathcal{A}}, V^\omega \setminus \mathrm{Win})$.

Lemma

Let $\mathcal{G}$ be a game. A winning strategy for Player i from v in $\mathcal{G}$ is a winning strategy for Player $1 - i$ from v in $\overline{\mathcal{G}}$.

Corollary

$W_i(\mathcal{G}) = W_{1-i}(\overline{\mathcal{G}})$ *for* $i \in \{0, 1\}$.

Co-Büchi and Büchi Games are Dual

- We have $\mathrm{Inf}(\rho) \not\subseteq C \Leftrightarrow \mathrm{Inf}(\rho) \cap (V \setminus C) \neq \emptyset$

Co-Büchi and Büchi Games are Dual

- We have $\mathrm{Inf}(\rho) \not\subseteq C \Leftrightarrow \mathrm{Inf}(\rho) \cap (V \setminus C) \neq \emptyset$
- Hence, $V^\omega \setminus \textsc{coBüchi}(C) = \textsc{Büchi}(V \setminus C)$.

Co-Büchi and Büchi Games are Dual

- We have $\mathrm{Inf}(\rho) \not\subseteq C \Leftrightarrow \mathrm{Inf}(\rho) \cap (V \setminus C) \neq \emptyset$
- Hence, $V^\omega \setminus \textsc{coBüchi}(C) = \textsc{Büchi}(V \setminus C)$.

Lemma

Co-Büchi and Büchi games are dual.

Co-Büchi and Büchi Games are Dual

- We have $\mathrm{Inf}(\rho) \not\subseteq C \Leftrightarrow \mathrm{Inf}(\rho) \cap (V \setminus C) \neq \emptyset$
- Hence, $V^\omega \setminus \textsc{coBüchi}(C) = \textsc{Büchi}(V \setminus C)$.

Lemma

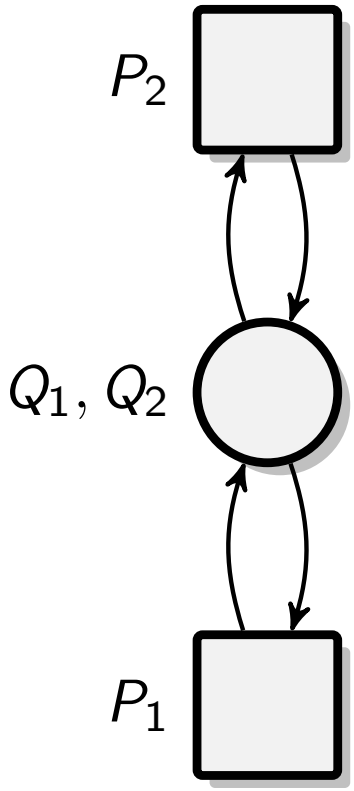
Co-Büchi and Büchi games are dual.

Theorem

Co-Büchi games are determined with positional winning strategies and can be solved in polynomial time in the number of edges of the underlying arena.

Request-response Games

A typical type of specification: answer requests.



Player 0 wins if every visit of an Q_j-vertex (a re**Q**uest) is followed by a visit of a P_j-vertex (a res**P**onse).

Request-response Games

Definition

Let $\mathcal{A} = (V, V_0, V_1, E)$ be an arena and let $(Q_j, P_j)_{j=1,\dots,k}$ be a finite family of subsets $Q_j, P_j \subseteq V$. Then, the request-response condition $\textsc{ReqRes}((Q_j, P_j)_{j=1,\dots,k})$ is defined as

$$\textsc{ReqRes}((Q_j, P_j)_{j=1,\dots,k}) = \{\rho \in V^\omega \mid \forall j = 1, \dots, k \, \forall n \in \mathbb{N} \\ \rho_n \in Q_j \text{ implies } \rho_{n'} \in P_j \text{ for some } n' \geq n\}.$$

We call a game $\mathcal{G} = (\mathcal{A}, \textsc{ReqRes}((Q_j, P_j)_{j=1,\dots,k}))$ a *request-response game*.

Intuitively, a visit to Q_j is a request that has to be answered by a later response, i.e., a visit to P_j.

Reductions

We do not give a direct algorithm to solve request-response games. Instead, we reduce them to a simpler game in a larger arena.

Reductions

We do not give a direct algorithm to solve request-response games. Instead, we reduce them to a simpler game in a larger arena.

- In general, a reduction transforms instances of a problem $\mathcal{P}$ into instances of another problem $\mathcal{P}'$ while "preserving" solutions. Then, solving $\mathcal{P}'$ also solves $\mathcal{P}$

Reductions

We do not give a direct algorithm to solve request-response games. Instead, we reduce them to a simpler game in a larger arena.

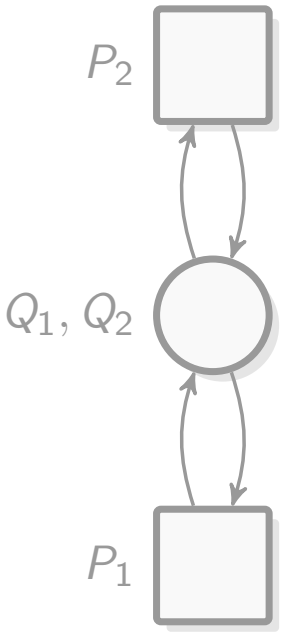
- In general, a reduction transforms instances of a problem $\mathcal{P}$ into instances of another problem $\mathcal{P}'$ while "preserving" solutions. Then, solving $\mathcal{P}'$ also solves $\mathcal{P}$
- We introduce a general framework of game reductions that preserve both winning regions and winning strategies.
- It also yields finite-state strategies, a generalization of positional strategies that allow to store a finite amount of information about a play's history.

Reductions

We do not give a direct algorithm to solve request-response games. Instead, we reduce them to a simpler game in a larger arena.

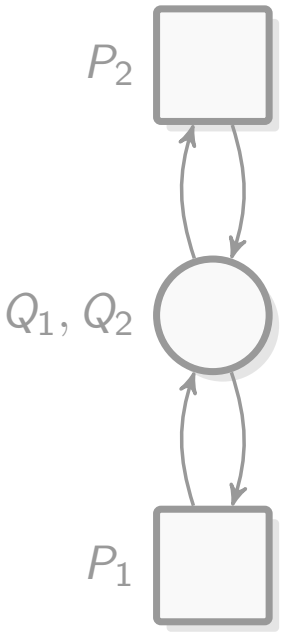
- In general, a reduction transforms instances of a problem $\mathcal{P}$ into instances of another problem $\mathcal{P}'$ while "preserving" solutions. Then, solving $\mathcal{P}'$ also solves $\mathcal{P}$
- We introduce a general framework of game reductions that preserve both winning regions and winning strategies.
- It also yields finite-state strategies, a generalization of positional strategies that allow to store a finite amount of information about a play's history.
- Application: reduce request-response games to Büchi games.

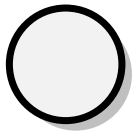
Back to the Example



Back to the Example

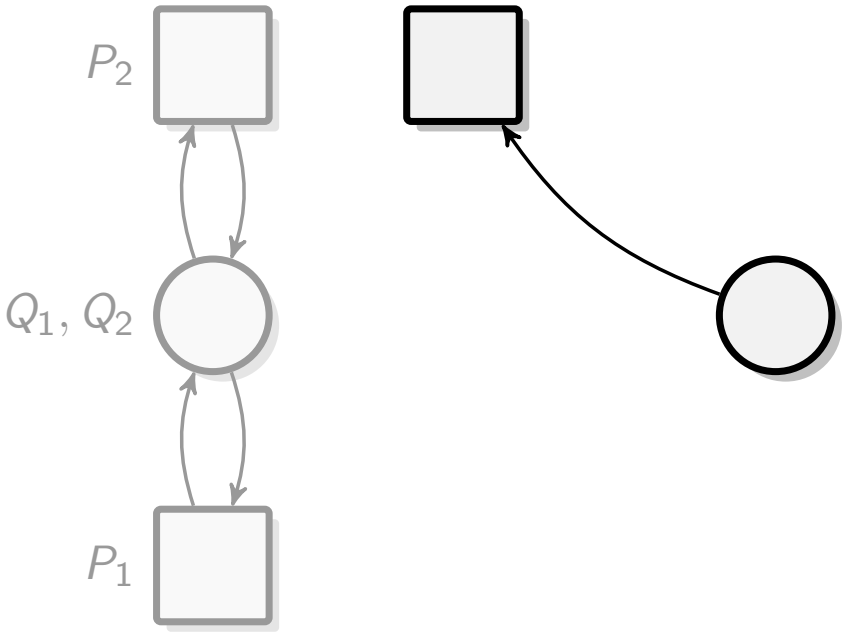
$\{1, 2\}, 1$





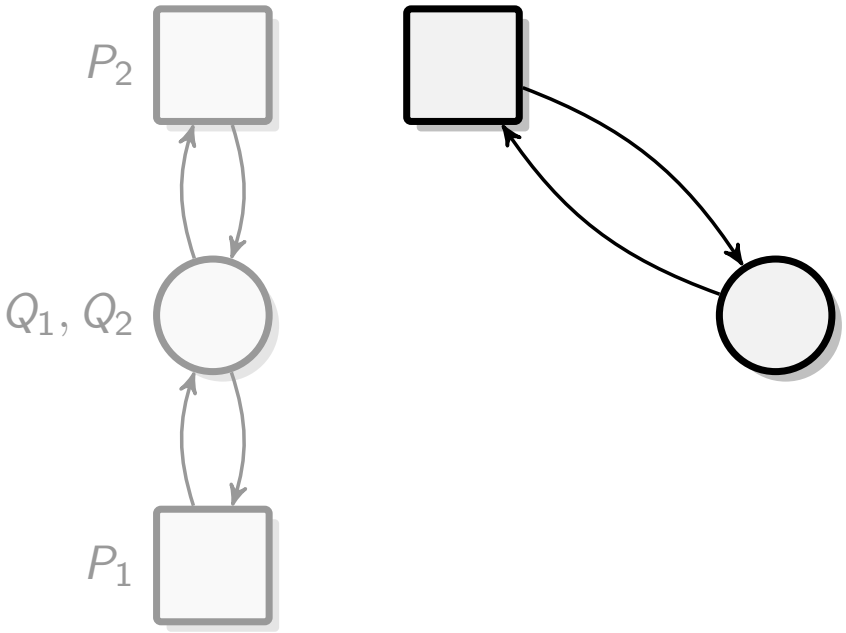
Back to the Example

$\{1\}, 1$ $\{1, 2\}, 1$

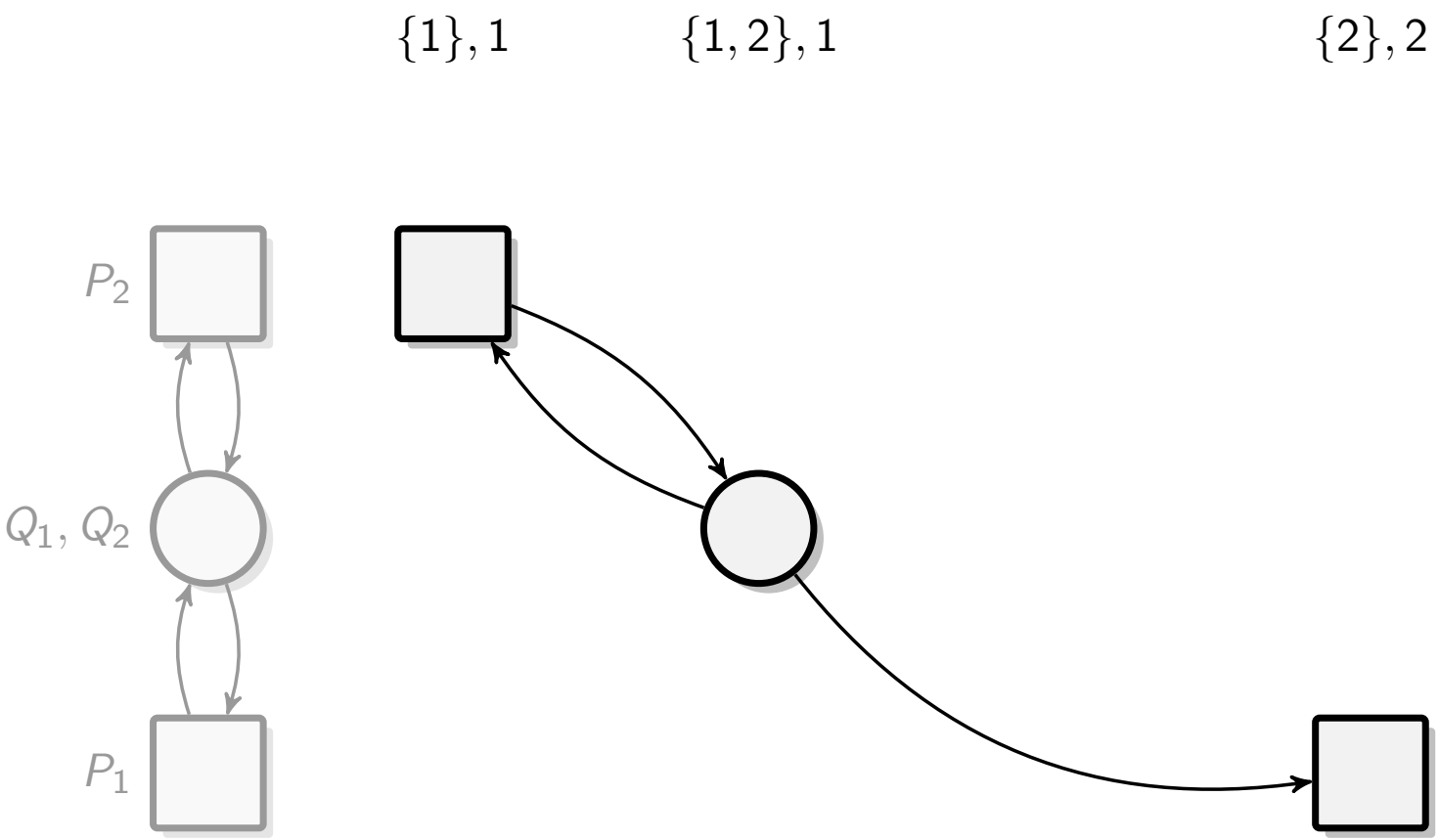


Back to the Example

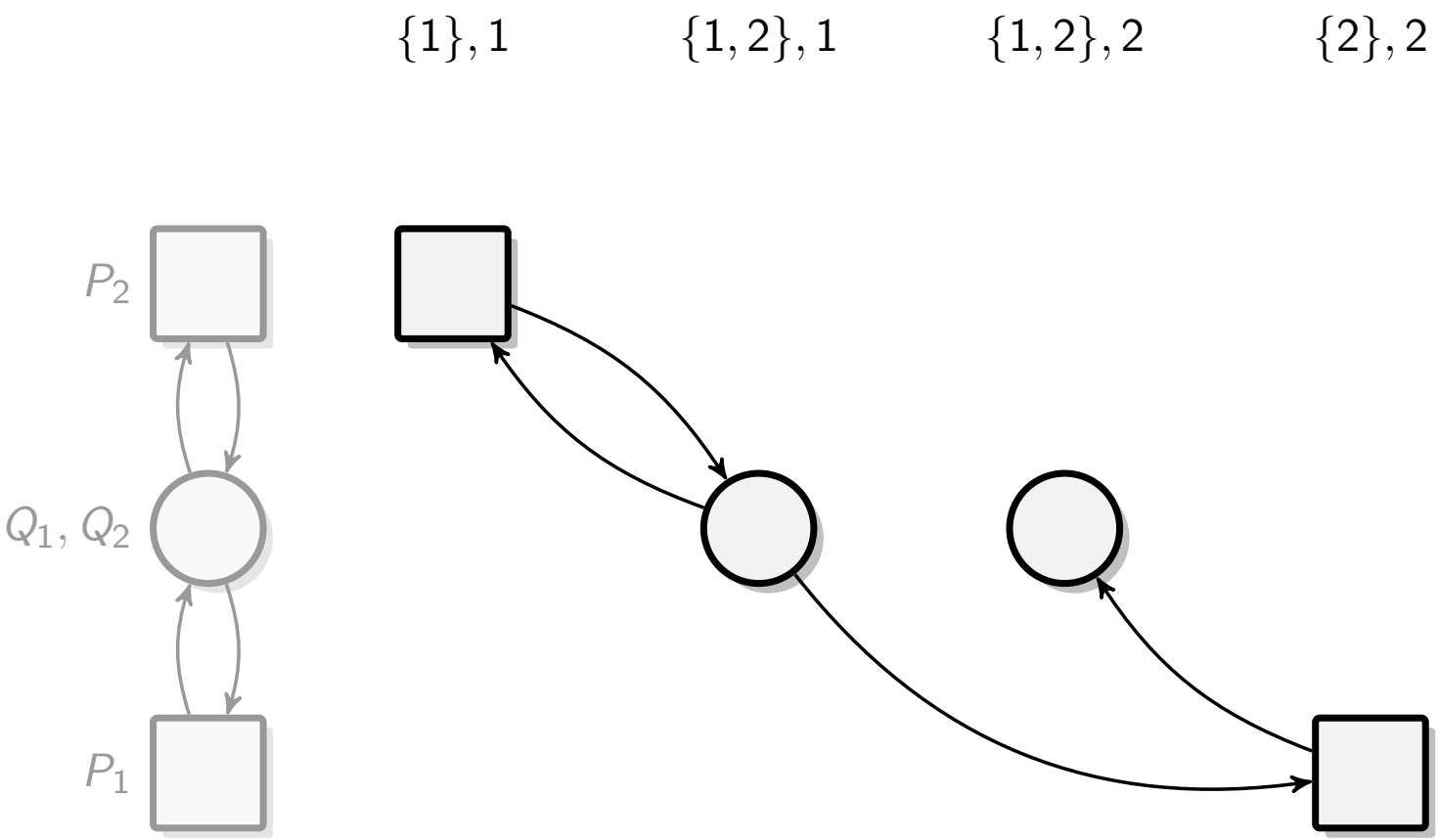
$\{1\}, 1$ $\{1, 2\}, 1$



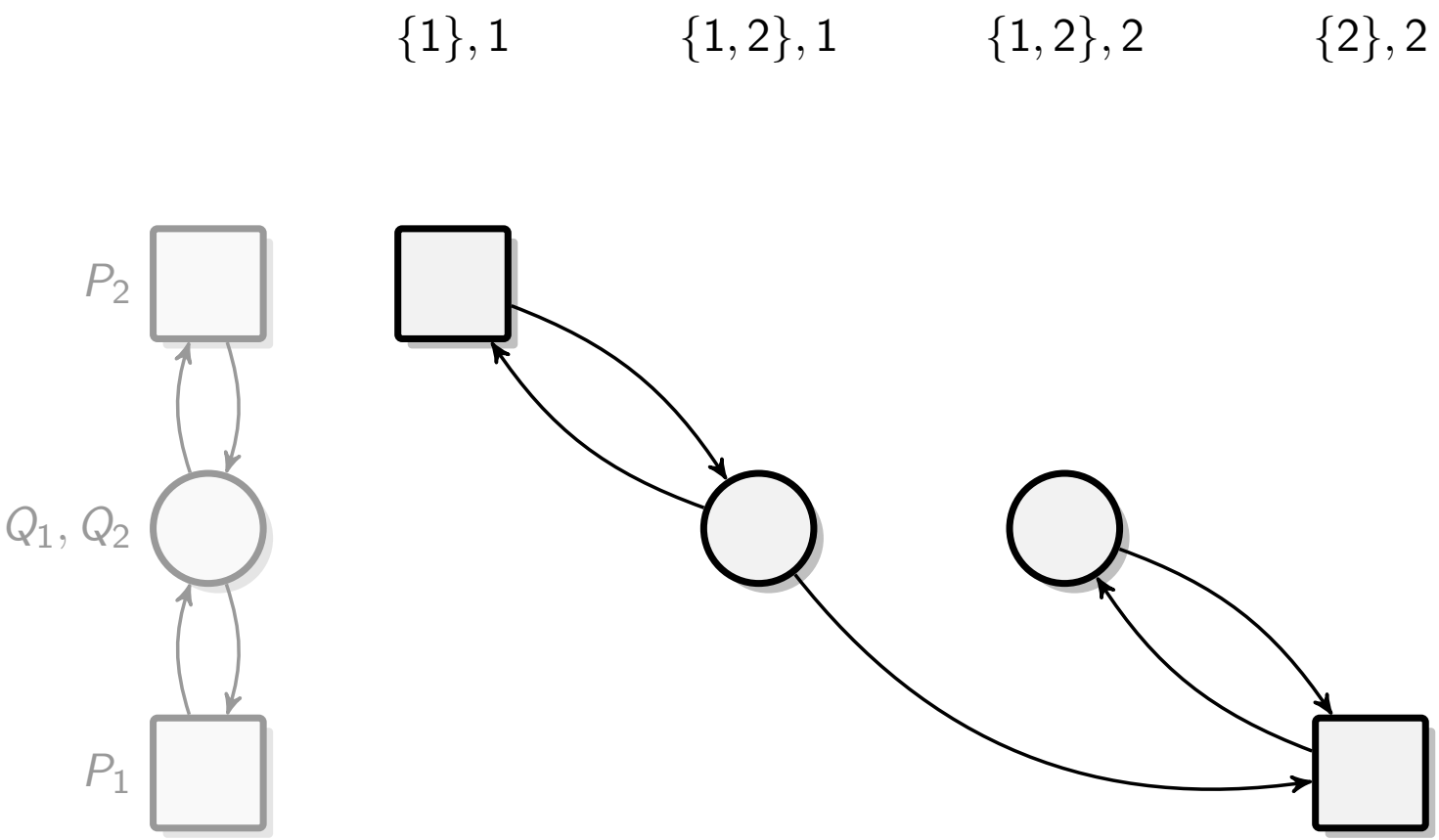
Back to the Example



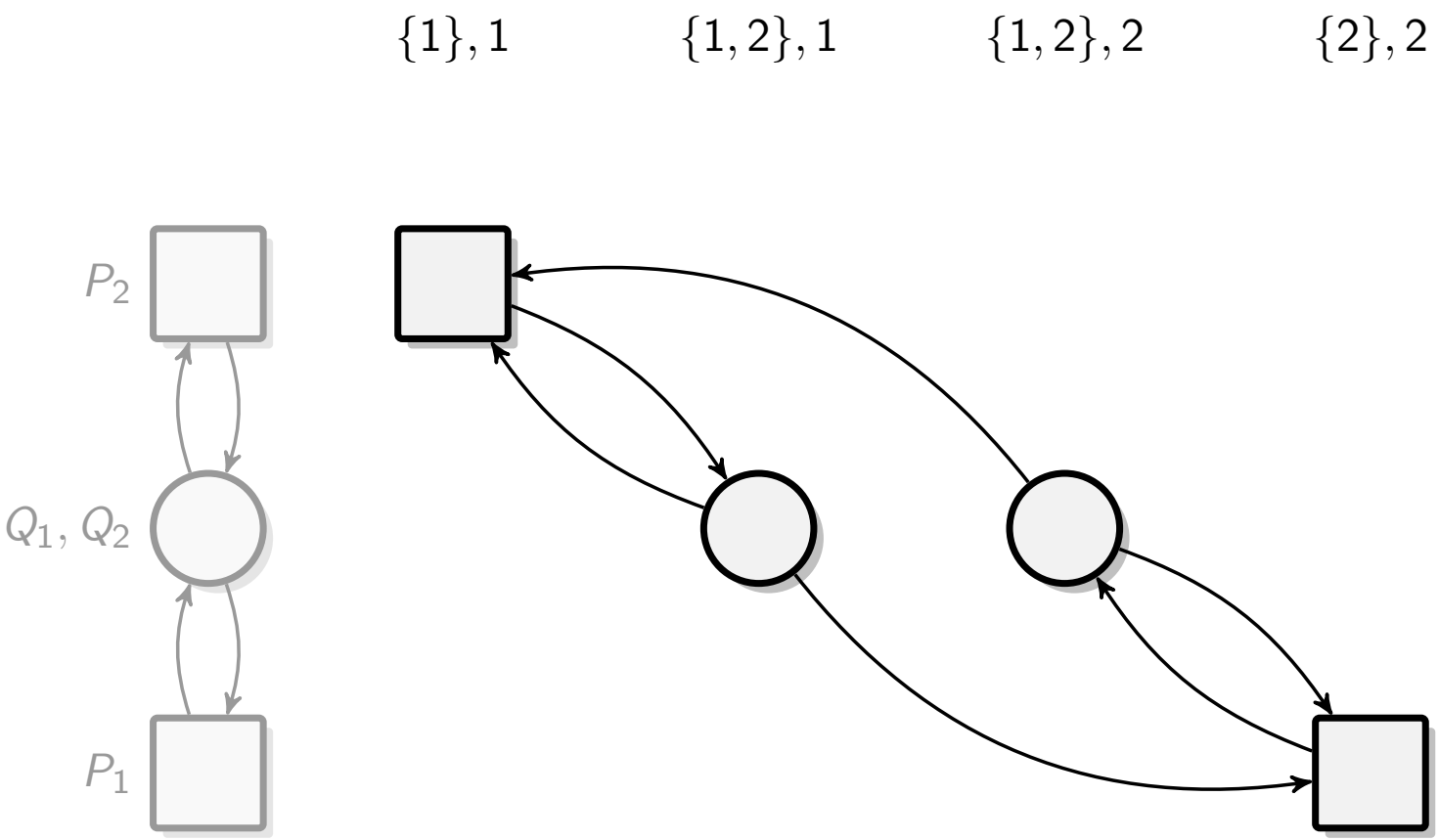
Back to the Example



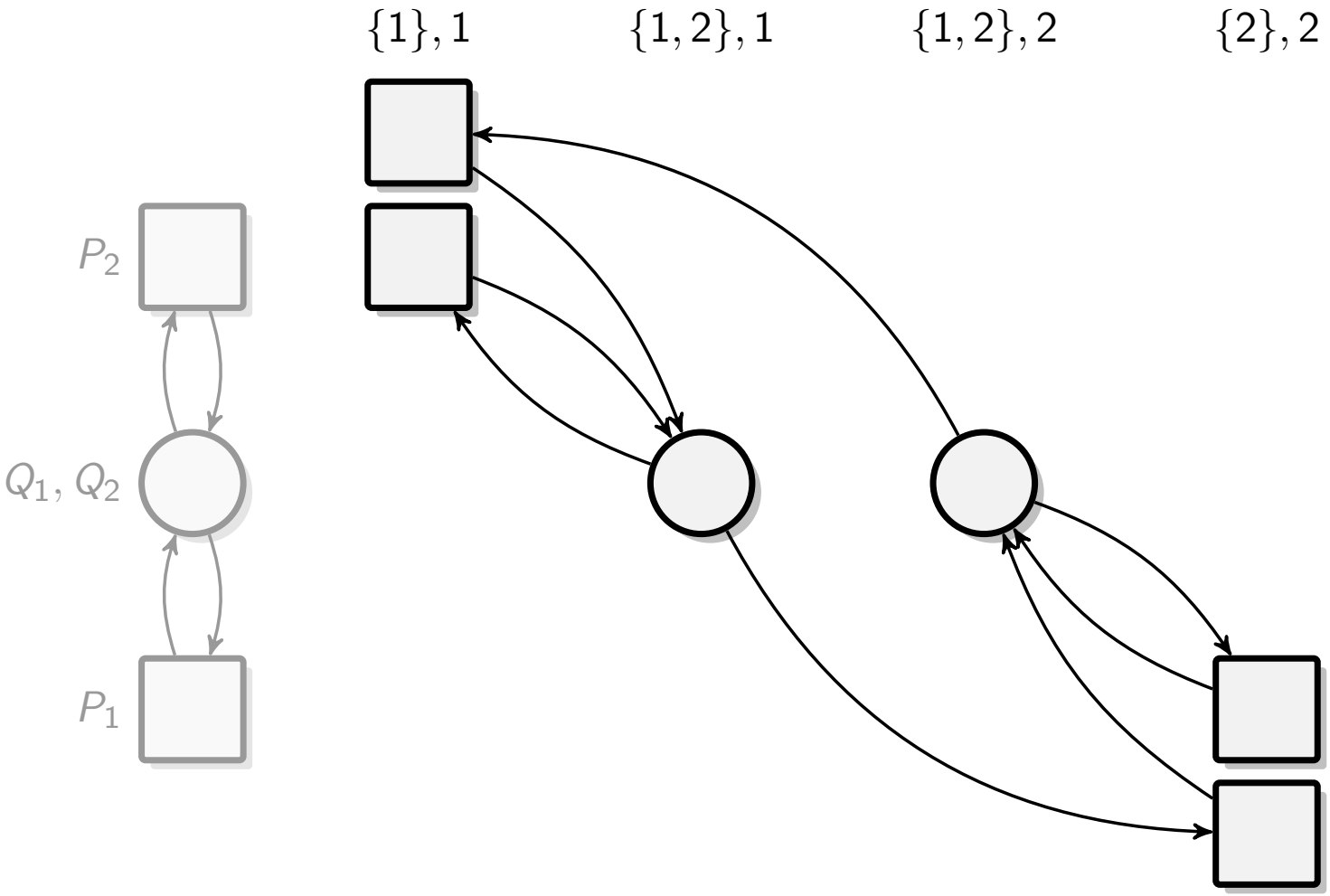
Back to the Example



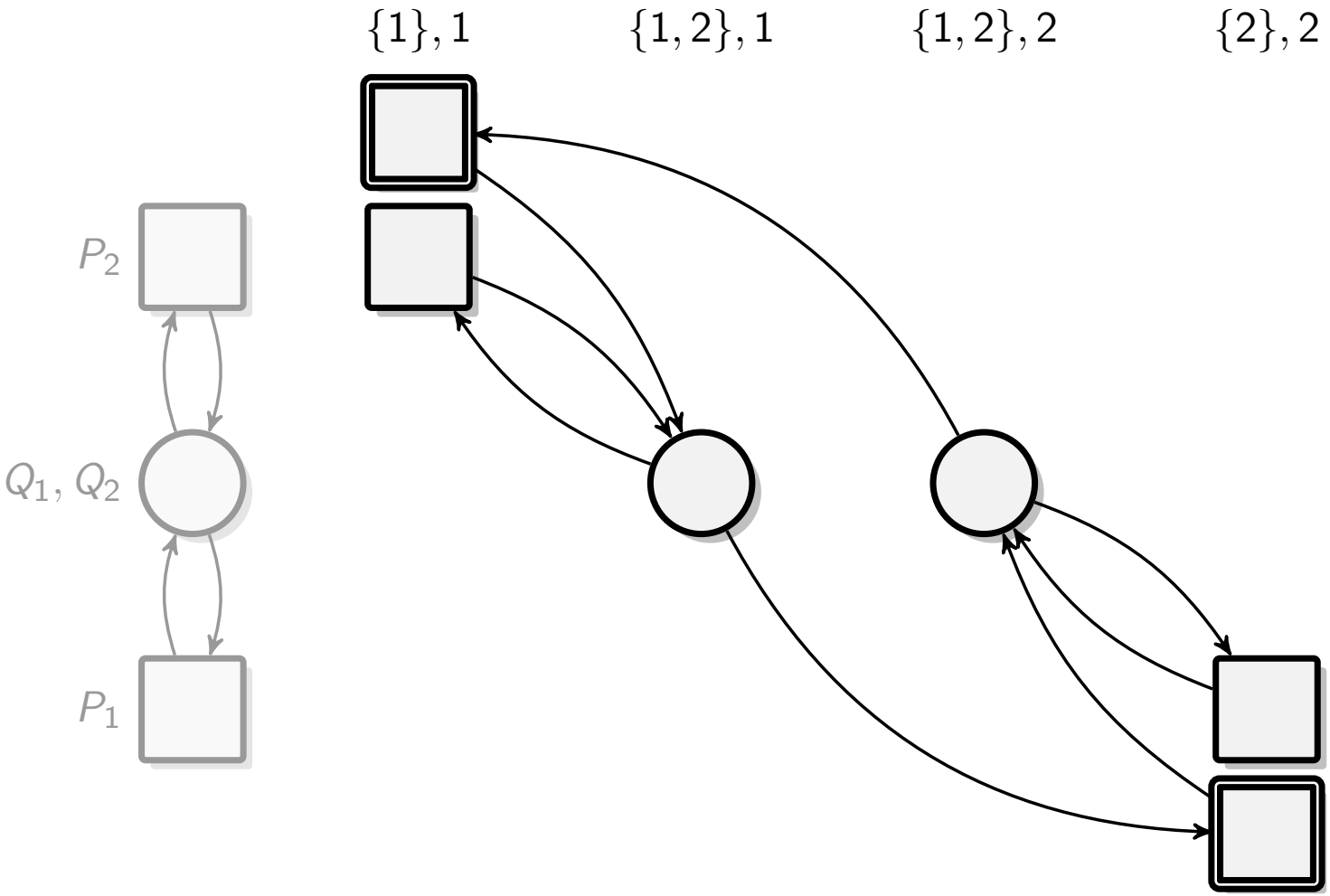
Back to the Example



Back to the Example



Back to the Example



The Büchi condition in the extended arena is satisfied if and only if the projected play satisfies the request-response condition.

Finite-state Strategies

Fix an arena $\mathcal{A} = (V, V_0, V_1, E)$.

- A memory structure $\mathcal{M} = (M, \mathrm{init}, \mathrm{upd})$ for $\mathcal{A}$ consists of
  - a finite set M of *memory states*,
  - a *initialization function* $\mathrm{init} \colon V \to M$, and
  - an *update function* $\mathrm{upd} \colon M \times V \to M$.

Finite-state Strategies

Fix an arena $\mathcal{A} = (V, V_0, V_1, E)$.

- A memory structure $\mathcal{M} = (M, \mathrm{init}, \mathrm{upd})$ for $\mathcal{A}$ consists of
  - a finite set M of *memory states*,
  - a *initialization function* $\mathrm{init} \colon V \to M$, and
  - an *update function* $\mathrm{upd} \colon M \times V \to M$.
- A *next-move function* for Player i is a map $\mathrm{nxt} \colon V \times M \to V$ such that $(v, \mathrm{nxt}(v, m)) \in E$ for all v and m.

Finite-state Strategies

Fix an arena $\mathcal{A} = (V, V_0, V_1, E)$.

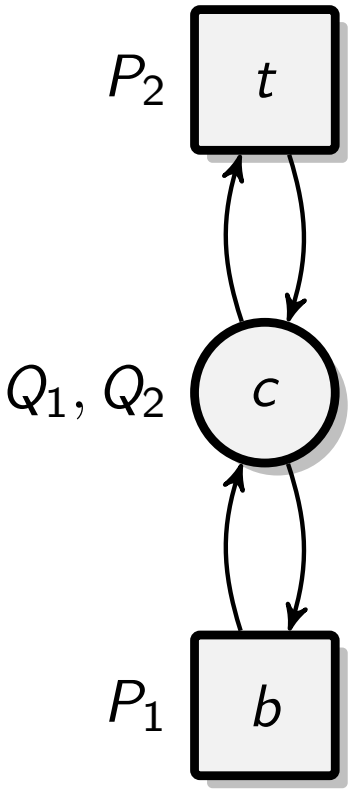
- A memory structure $\mathcal{M} = (M, \mathrm{init}, \mathrm{upd})$ for $\mathcal{A}$ consists of
  - a finite set M of *memory states*,
  - a *initialization function* $\mathrm{init} \colon V \to M$, and
  - an *update function* $\mathrm{upd} \colon M \times V \to M$.
- A *next-move function* for Player i is a map $\mathrm{nxt} \colon V \times M \to V$ such that $(v, \mathrm{nxt}(v, m)) \in E$ for all v and m.
- $\mathcal{M}$ and nxt implement a strategy σ for Player i defined via $\sigma(\rho_0 \cdots \rho_n) = \mathrm{nxt}(\rho_n, \mathrm{upd}^*(\rho_0 \cdots \rho_n))$ where
  - $\mathrm{upd}^*(\rho_0) = \mathrm{init}(\rho_0)$, and
  - $\mathrm{upd}^*(\rho_0 \cdots \rho_n \rho_{n+1}) = \mathrm{upd}(\mathrm{upd}^*(\rho_0 \cdots \rho_n), \rho_{n+1})$.

Finite-state Strategies

Fix an arena $\mathcal{A} = (V, V_0, V_1, E)$.

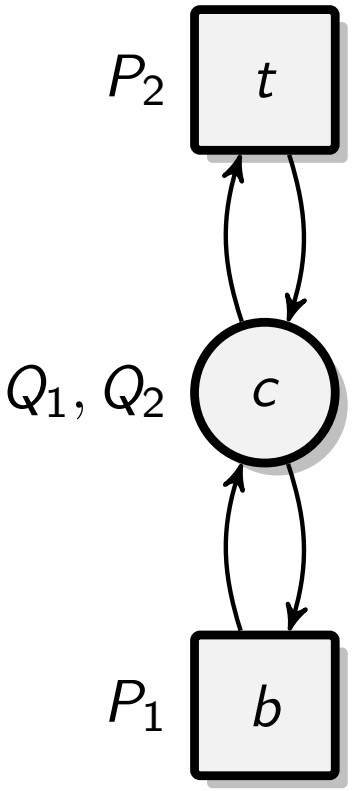
- A memory structure $\mathcal{M} = (M, \mathrm{init}, \mathrm{upd})$ for $\mathcal{A}$ consists of
  - a finite set M of *memory states*,
  - a *initialization function* $\mathrm{init}\colon V \to M$, and
  - an *update function* $\mathrm{upd}\colon M \times V \to M$.
- A *next-move function* for Player i is a map $\mathrm{nxt}\colon V \times M \to V$ such that $(v, \mathrm{nxt}(v, m)) \in E$ for all v and m.
- $\mathcal{M}$ and nxt implement a strategy σ for Player i defined via $\sigma(\rho_0 \cdots \rho_n) = \mathrm{nxt}(\rho_n, \mathrm{upd}^*(\rho_0 \cdots \rho_n))$ where
  - $\mathrm{upd}^*(\rho_0) = \mathrm{init}(\rho_0)$, and
  - $\mathrm{upd}^*(\rho_0 \cdots \rho_n \rho_{n+1}) = \mathrm{upd}(\mathrm{upd}^*(\rho_0 \cdots \rho_n), \rho_{n+1})$.
- A strategy is called *finite-state* if it is implementable by a memory structure and a next-move function.

Back to the Example



Example play

Back to the Example

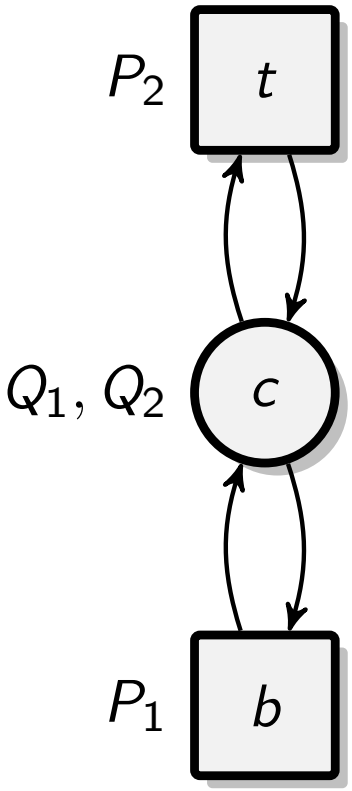


A winning strategy for Player 0:
alternate between top and bottom.

- $M = \{0, 1\}$
- $\mathrm{init}(v) = 0$ for all v
- $\mathrm{upd}(m, c) = 1 - m$ and $\mathrm{upd}(m, v) = m$ for $v \neq c$
- $\mathrm{nxt}(c, 0) = b$ and $\mathrm{nxt}(c, 1) = t$

Example play

Back to the Example



A winning strategy for Player 0:
alternate between top and bottom.

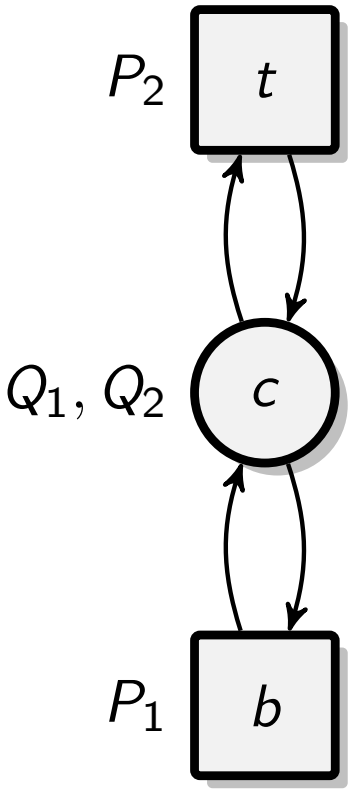
- $M = \{0, 1\}$
- $\mathrm{init}(v) = 0$ for all v
- $\mathrm{upd}(m, c) = 1 - m$ and $\mathrm{upd}(m, v) = m$ for $v \neq c$
- $\mathrm{nxt}(c, 0) = b$ and $\mathrm{nxt}(c, 1) = t$

Example play

Play: c

Memory:

Back to the Example



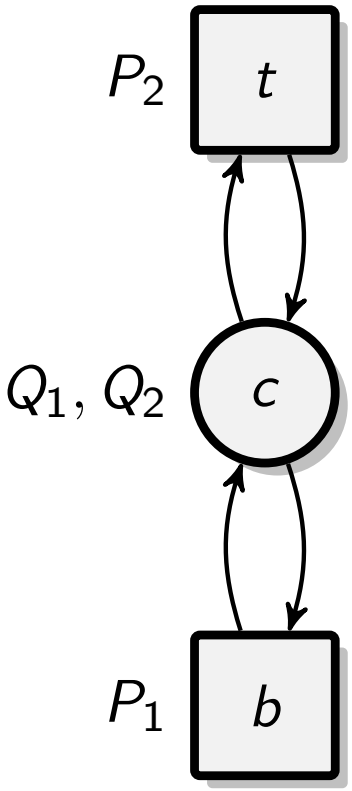
A winning strategy for Player 0:
alternate between top and bottom.

- $M = \{0, 1\}$
- $\mathrm{init}(v) = 0$ for all v
- $\mathrm{upd}(m, c) = 1 - m$ and $\mathrm{upd}(m, v) = m$ for $v \neq c$
- $\mathrm{nxt}(c, 0) = b$ and $\mathrm{nxt}(c, 1) = t$

Example play

Play:	c
Memory:	$\mathrm{init}(c)$

Back to the Example



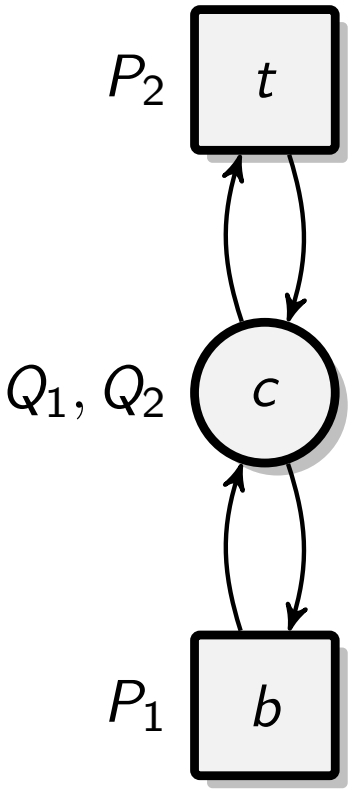
A winning strategy for Player 0:
alternate between top and bottom.

- $M = \{0, 1\}$
- $\mathrm{init}(v) = 0$ for all v
- $\mathrm{upd}(m, c) = 1 - m$ and $\mathrm{upd}(m, v) = m$ for $v \neq c$
- $\mathrm{nxt}(c, 0) = b$ and $\mathrm{nxt}(c, 1) = t$

Example play

Play:	c
Memory:	0

Back to the Example



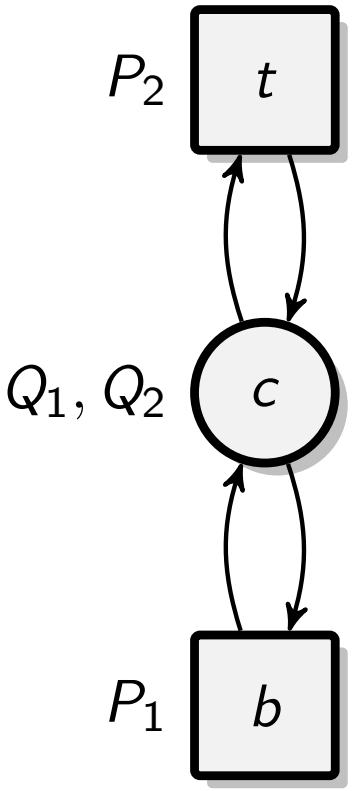
A winning strategy for Player 0:
alternate between top and bottom.

- $M = \{0, 1\}$
- $\mathrm{init}(v) = 0$ for all v
- $\mathrm{upd}(m, c) = 1 - m$ and $\mathrm{upd}(m, v) = m$ for $v \neq c$
- $\mathrm{nxt}(c, 0) = b$ and $\mathrm{nxt}(c, 1) = t$

Example play

Play:	c	$\mathrm{nxt}(c, 0)$
Memory:	0	

Back to the Example



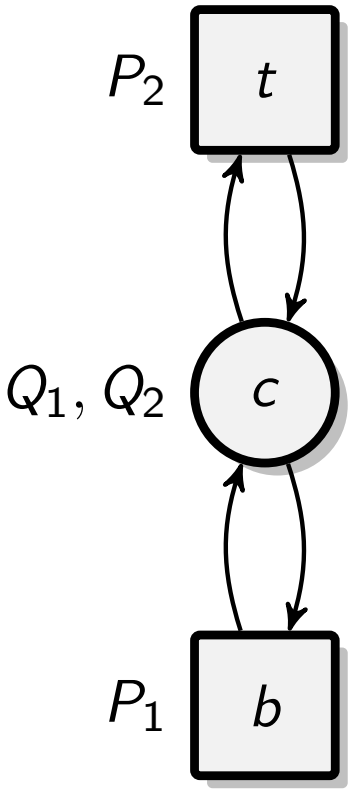
A winning strategy for Player 0:
alternate between top and bottom.

- $M = \{0, 1\}$
- $\mathrm{init}(v) = 0$ for all v
- $\mathrm{upd}(m, c) = 1 - m$ and $\mathrm{upd}(m, v) = m$ for $v \neq c$
- $\mathrm{nxt}(c, 0) = b$ and $\mathrm{nxt}(c, 1) = t$

Example play

Play:	c	b
Memory:	0	

Back to the Example



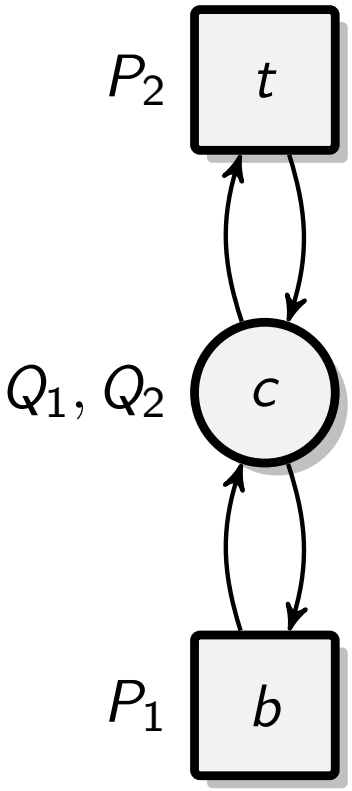
A winning strategy for Player 0:
alternate between top and bottom.

- $M = \{0, 1\}$
- $\mathrm{init}(v) = 0$ for all v
- $\mathrm{upd}(m, c) = 1 - m$ and $\mathrm{upd}(m, v) = m$ for $v \neq c$
- $\mathrm{nxt}(c, 0) = b$ and $\mathrm{nxt}(c, 1) = t$

Example play

Play:	c	b
Memory:	0	$\mathrm{upd}(0, b)$

Back to the Example



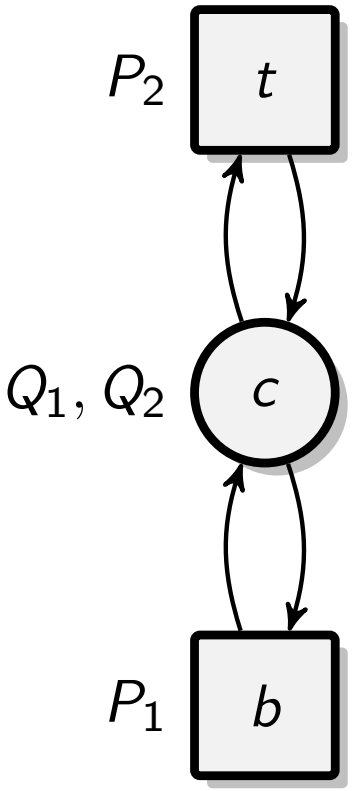
A winning strategy for Player 0:
alternate between top and bottom.

- $M = \{0, 1\}$
- $\mathrm{init}(v) = 0$ for all v
- $\mathrm{upd}(m, c) = 1 - m$ and $\mathrm{upd}(m, v) = m$ for $v \neq c$
- $\mathrm{nxt}(c, 0) = b$ and $\mathrm{nxt}(c, 1) = t$

Example play

Play:	c	b
Memory:	0	0

Back to the Example



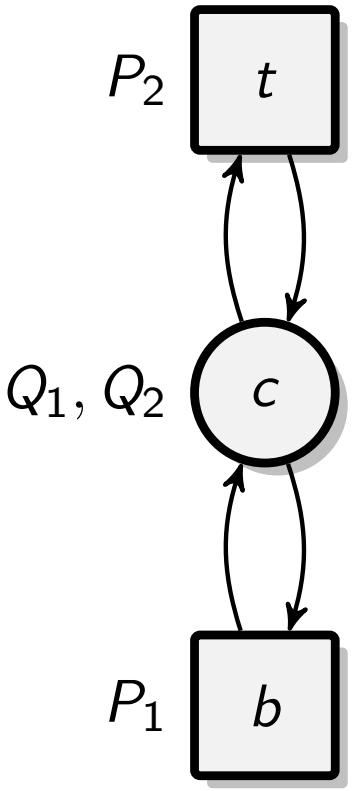
A winning strategy for Player 0:
alternate between top and bottom.

- $M = \{0, 1\}$
- $\mathrm{init}(v) = 0$ for all v
- $\mathrm{upd}(m, c) = 1 - m$ and $\mathrm{upd}(m, v) = m$ for $v \neq c$
- $\mathrm{nxt}(c, 0) = b$ and $\mathrm{nxt}(c, 1) = t$

Example play

Play:	c	b	c
Memory:	0	0	

Back to the Example



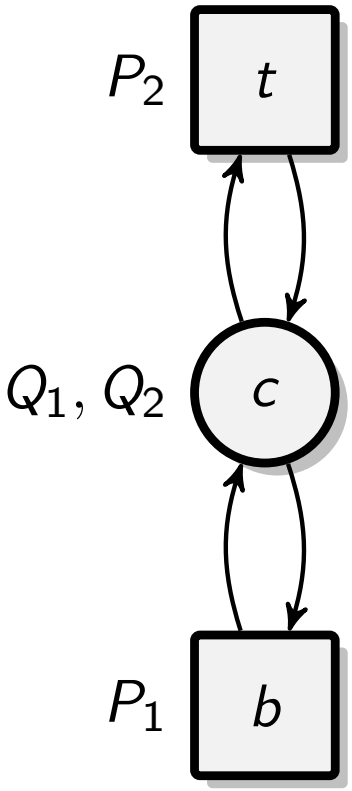
A winning strategy for Player 0:
alternate between top and bottom.

- $M = \{0, 1\}$
- $\mathrm{init}(v) = 0$ for all v
- $\mathrm{upd}(m, c) = 1 - m$ and $\mathrm{upd}(m, v) = m$ for $v \neq c$
- $\mathrm{nxt}(c, 0) = b$ and $\mathrm{nxt}(c, 1) = t$

Example play

Play:	c	b	c
Memory:	0	0	$\mathrm{upd}(0, c)$

Back to the Example



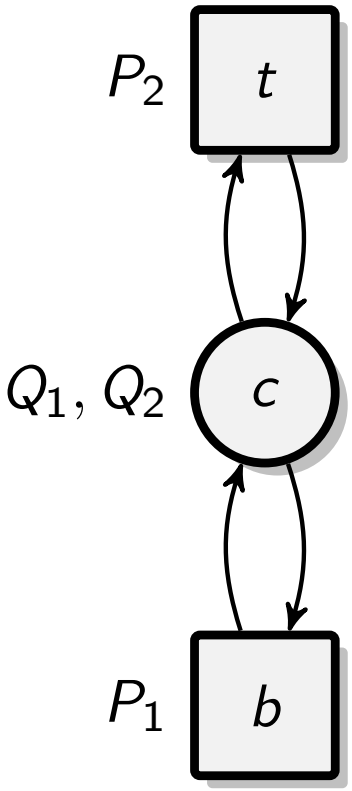
A winning strategy for Player 0:
alternate between top and bottom.

- $M = \{0, 1\}$
- $\mathrm{init}(v) = 0$ for all v
- $\mathrm{upd}(m, c) = 1 - m$ and $\mathrm{upd}(m, v) = m$ for $v \neq c$
- $\mathrm{nxt}(c, 0) = b$ and $\mathrm{nxt}(c, 1) = t$

Example play

Play:	c	b	c
Memory:	0	0	1

Back to the Example



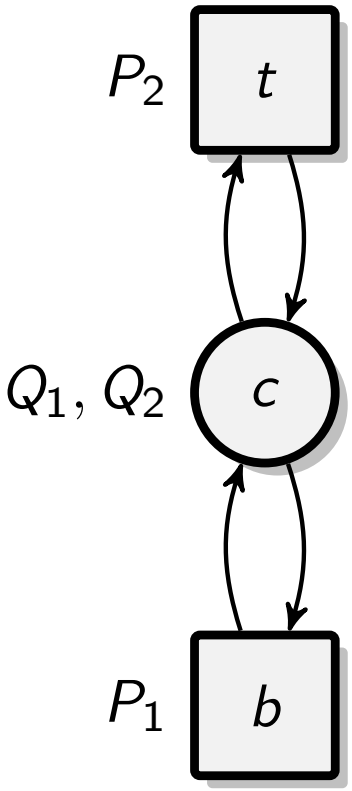
A winning strategy for Player 0:
alternate between top and bottom.

- $M = \{0, 1\}$
- $\mathrm{init}(v) = 0$ for all v
- $\mathrm{upd}(m, c) = 1 - m$ and $\mathrm{upd}(m, v) = m$ for $v \neq c$
- $\mathrm{nxt}(c, 0) = b$ and $\mathrm{nxt}(c, 1) = t$

Example play

Play:	c	b	c	$\mathrm{nxt}(c, 1)$
Memory:	0	0	1	

Back to the Example



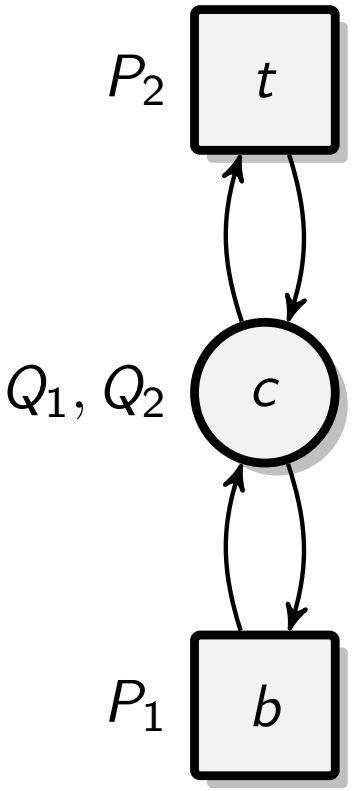
A winning strategy for Player 0:
alternate between top and bottom.

- $M = \{0, 1\}$
- $\mathrm{init}(v) = 0$ for all v
- $\mathrm{upd}(m, c) = 1 - m$ and $\mathrm{upd}(m, v) = m$ for $v \neq c$
- $\mathrm{nxt}(c, 0) = b$ and $\mathrm{nxt}(c, 1) = t$

Example play

Play:	c	b	c	t
Memory:	0	0	1	

Back to the Example



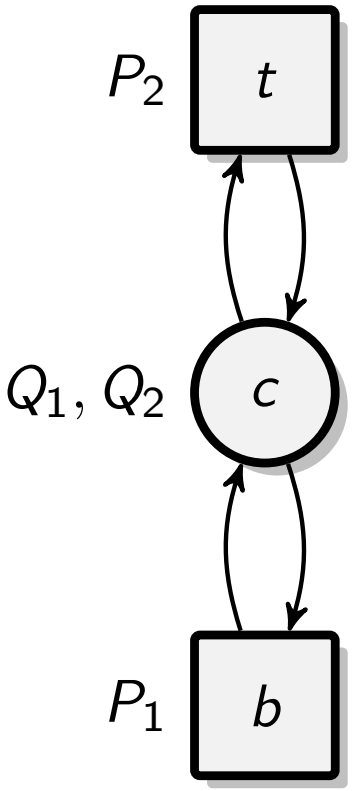
A winning strategy for Player 0:
alternate between top and bottom.

- $M = \{0, 1\}$
- $\mathrm{init}(v) = 0$ for all v
- $\mathrm{upd}(m, c) = 1 - m$ and $\mathrm{upd}(m, v) = m$ for $v \neq c$
- $\mathrm{nxt}(c, 0) = b$ and $\mathrm{nxt}(c, 1) = t$

Example play

Play:	c	b	c	t
Memory:	0	0	1	$\mathrm{upd}(1, t)$

Back to the Example



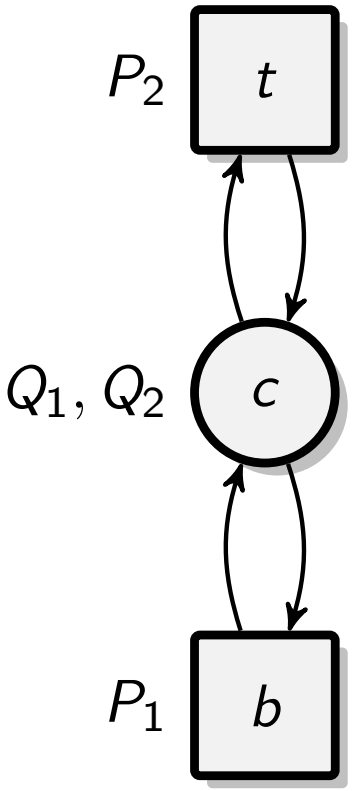
A winning strategy for Player 0:
alternate between top and bottom.

- $M = \{0, 1\}$
- $\mathrm{init}(v) = 0$ for all v
- $\mathrm{upd}(m, c) = 1 - m$ and $\mathrm{upd}(m, v) = m$ for $v \neq c$
- $\mathrm{nxt}(c, 0) = b$ and $\mathrm{nxt}(c, 1) = t$

Example play

Play:	c	b	c	t
Memory:	0	0	1	1

Back to the Example



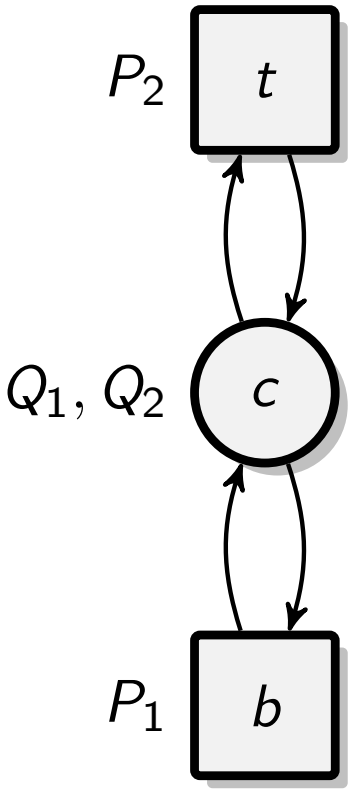
A winning strategy for Player 0:
alternate between top and bottom.

- $M = \{0, 1\}$
- $\mathrm{init}(v) = 0$ for all v
- $\mathrm{upd}(m, c) = 1 - m$ and $\mathrm{upd}(m, v) = m$ for $v \neq c$
- $\mathrm{nxt}(c, 0) = b$ and $\mathrm{nxt}(c, 1) = t$

Example play

Play:	c	b	c	t	c
Memory:	0	0	1	1	

Back to the Example



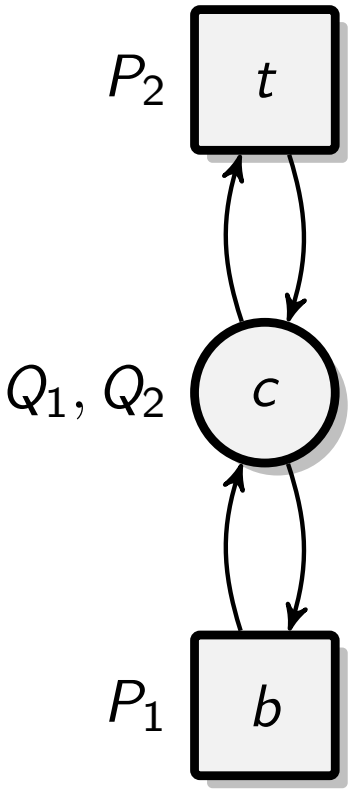
A winning strategy for Player 0:
alternate between top and bottom.

- $M = \{0, 1\}$
- $\mathrm{init}(v) = 0$ for all v
- $\mathrm{upd}(m, c) = 1 - m$ and $\mathrm{upd}(m, v) = m$ for $v \neq c$
- $\mathrm{nxt}(c, 0) = b$ and $\mathrm{nxt}(c, 1) = t$

Example play

Play:	c	b	c	t	c
Memory:	0	0	1	1	$\mathrm{upd}(1, c)$

Back to the Example



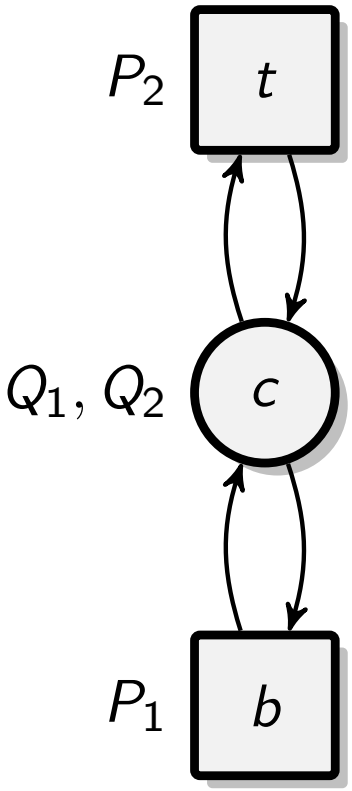
A winning strategy for Player 0:
alternate between top and bottom.

- $M = \{0, 1\}$
- $\mathrm{init}(v) = 0$ for all v
- $\mathrm{upd}(m, c) = 1 - m$ and $\mathrm{upd}(m, v) = m$ for $v \neq c$
- $\mathrm{nxt}(c, 0) = b$ and $\mathrm{nxt}(c, 1) = t$

Example play

Play:	c	b	c	t	c
Memory:	0	0	1	1	0

Back to the Example



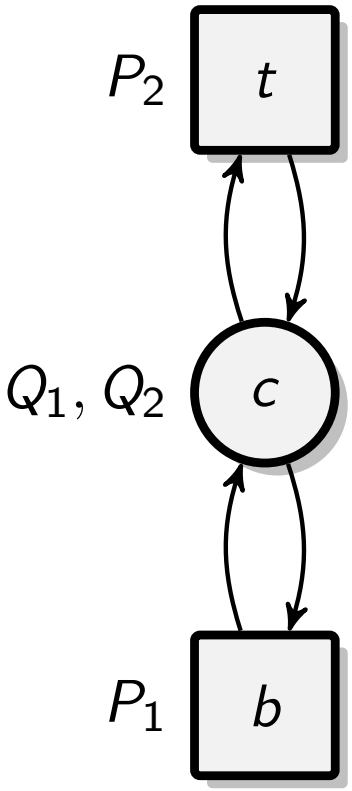
A winning strategy for Player 0:
alternate between top and bottom.

- $M = \{0, 1\}$
- $\mathrm{init}(v) = 0$ for all v
- $\mathrm{upd}(m, c) = 1 - m$ and $\mathrm{upd}(m, v) = m$ for $v \neq c$
- $\mathrm{nxt}(c, 0) = b$ and $\mathrm{nxt}(c, 1) = t$

Example play

Play:	c	b	c	t	c	$\mathrm{nxt}(c, 0)$
Memory:	0	0	1	1	0	

Back to the Example



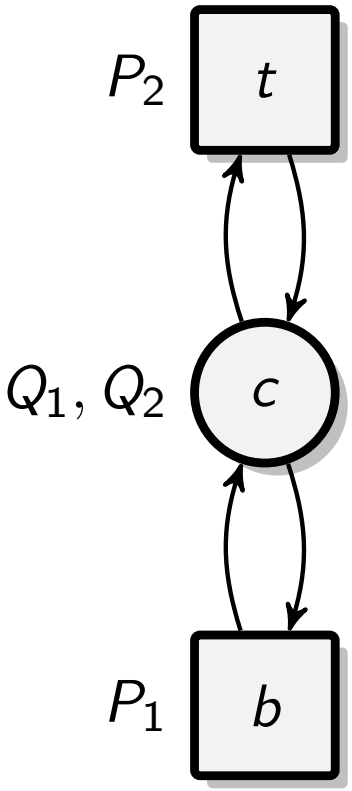
A winning strategy for Player 0:
alternate between top and bottom.

- $M = \{0, 1\}$
- $\mathrm{init}(v) = 0$ for all v
- $\mathrm{upd}(m, c) = 1 - m$ and $\mathrm{upd}(m, v) = m$ for $v \neq c$
- $\mathrm{nxt}(c, 0) = b$ and $\mathrm{nxt}(c, 1) = t$

Example play

Play:	c	b	c	t	c	b
Memory:	0	0	1	1	0	

Back to the Example



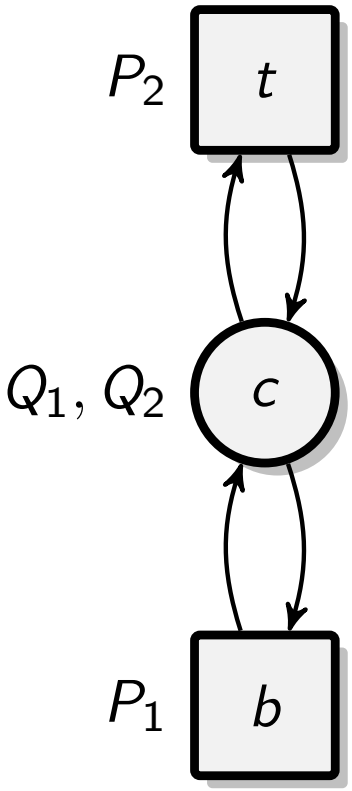
A winning strategy for Player 0:
alternate between top and bottom.

- $M = \{0, 1\}$
- $\mathrm{init}(v) = 0$ for all v
- $\mathrm{upd}(m, c) = 1 - m$ and $\mathrm{upd}(m, v) = m$ for $v \neq c$
- $\mathrm{nxt}(c, 0) = b$ and $\mathrm{nxt}(c, 1) = t$

Example play

Play:	c	b	c	t	c	b
Memory:	0	0	1	1	0	$\mathrm{upd}(0, b)$

Back to the Example



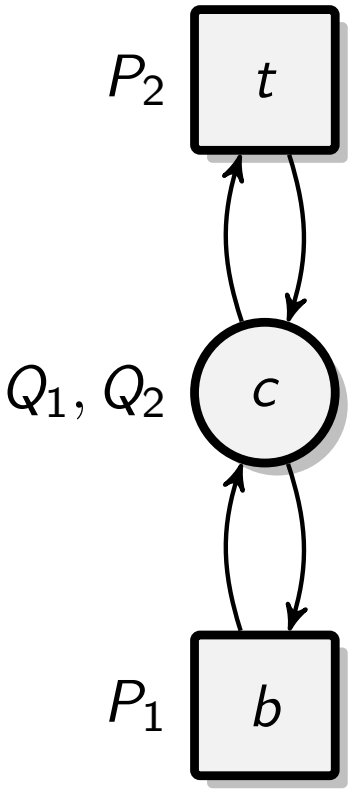
A winning strategy for Player 0:
alternate between top and bottom.

- $M = \{0, 1\}$
- $\mathrm{init}(v) = 0$ for all v
- $\mathrm{upd}(m, c) = 1 - m$ and $\mathrm{upd}(m, v) = m$ for $v \neq c$
- $\mathrm{nxt}(c, 0) = b$ and $\mathrm{nxt}(c, 1) = t$

Example play

Play:	c	b	c	t	c	b
Memory:	0	0	1	1	0	0

Back to the Example



A winning strategy for Player 0:
alternate between top and bottom.

- $M = \{0, 1\}$
- $\mathrm{init}(v) = 0$ for all v
- $\mathrm{upd}(m, c) = 1 - m$ and $\mathrm{upd}(m, v) = m$ for $v \neq c$
- $\mathrm{nxt}(c, 0) = b$ and $\mathrm{nxt}(c, 1) = t$

Example play

Play:	c	b	c	t	c	b	$\cdots$
Memory:	0	0	1	1	0	0	$\cdots$

Game Reductions

- The product arena $\mathcal{A} \times \mathcal{M} = (V', V_0', V_1', E')$ is defined as
  - $V' = V \times M$,
  - $V_i' = V_i \times M$, and
  - $((v, m), (v', m')) \in E'$ if and only if $(v, v') \in E$ and $\mathrm{upd}(m, v') = m'$.

Game Reductions

- The product arena $\mathcal{A} \times \mathcal{M} = (V', V_0', V_1', E')$ is defined as
  - $V' = V \times M$,
  - $V_i' = V_i \times M$, and
  - $((v, m), (v', m')) \in E'$ if and only if $(v, v') \in E$ and $\mathrm{upd}(m, v') = m'$.

- For every play $\rho = \rho_0 \rho_1 \rho_2 \cdots$ in $\mathcal{A}$ there is an extended play $\mathrm{ext}(\rho) = (\rho_0, m_0)(\rho_1, m_1)(\rho_2, m_2) \cdots$ in $\mathcal{A} \times \mathcal{M}$ with $m_n = \mathrm{upd}^*(\rho_0 \cdots \rho_n)$.

Game Reductions

- The product arena $\mathcal{A} \times \mathcal{M} = (V', V_0', V_1', E')$ is defined as
  - $V' = V \times M$,
  - $V_i' = V_i \times M$, and
  - $((v, m), (v', m')) \in E'$ if and only if $(v, v') \in E$ and $\mathrm{upd}(m, v') = m'$.

- For every play $\rho = \rho_0 \rho_1 \rho_2 \cdots$ in $\mathcal{A}$ there is an extended play $\mathrm{ext}(\rho) = (\rho_0, m_0)(\rho_1, m_1)(\rho_2, m_2) \cdots$ in $\mathcal{A} \times \mathcal{M}$ with $m_n = \mathrm{upd}^*(\rho_0 \cdots \rho_n)$.

- A game $\mathcal{G} = (\mathcal{A}, \mathrm{Win})$ is reducible to a game $\mathcal{G}' = (\mathcal{A}', \mathrm{Win}')$ via $\mathcal{M}$, written $\mathcal{G} \leq_{\mathcal{M}} \mathcal{G}'$, if
  - $\mathcal{A}' = \mathcal{A} \times \mathcal{M}$, and
  - For each play ρ in $\mathcal{A}$: $\rho \in \mathrm{Win}$ if and only if $\mathrm{ext}(\rho) \in \mathrm{Win}'$.

The Reduction Lemma

Lemma

Let $\mathcal{G}$ be a game with vertex set V and $W \subseteq V$. If $\mathcal{G} \leq_{\mathcal{M}} \mathcal{G}'$ and Player i has a positional winning strategy for $\mathcal{G}'$ from every vertex $(v, \mathrm{init}(v))$ with $v \in W$, then she has a winning strategy with memory $\mathcal{M}$ for $\mathcal{G}$ from every $v \in W$.

Proof.

See lecture notes. □

The Reduction Lemma

Lemma

Let $\mathcal{G}$ be a game with vertex set V and $W \subseteq V$. If $\mathcal{G} \leq_{\mathcal{M}} \mathcal{G}'$ and Player i has a positional winning strategy for $\mathcal{G}'$ from every vertex $(v, \mathrm{init}(v))$ with $v \in W$, then she has a winning strategy with memory $\mathcal{M}$ for $\mathcal{G}$ from every $v \in W$.

Proof.

See lecture notes. □

Recall (Problem set 1)

A winning strategy for Player i is *uniform*, if it is winning from every vertex in Player i's winning region.

The Reduction Lemma

Lemma

Let $\mathcal{G}$ be a game with vertex set V and $W \subseteq V$. If $\mathcal{G} \leq_{\mathcal{M}} \mathcal{G}'$ and Player i has a positional winning strategy for $\mathcal{G}'$ from every vertex $(v, \mathrm{init}(v))$ with $v \in W$, then she has a winning strategy with memory $\mathcal{M}$ for $\mathcal{G}$ from every $v \in W$.

Proof.

See lecture notes. □

Recall (Problem set 1)

A winning strategy for Player i is *uniform*, if it is winning from every vertex in Player i's winning region.

Corollary

If Player i has a uniform positional winning strategy for $\mathcal{G}'$ then she has a uniform finite-state winning strategy with memory $\mathcal{M}$ for $\mathcal{G}$.

Back to Request-response Games

Fix a request-response game

$$\mathcal{G} = ((V, V_0, V_1, E), \textsc{ReqRes}((Q_j, P_j)_{j=1,\dots,k}))$$

with (w.l.o.g.) $k \geq 2$ and define $\mathcal{M} = (M, \mathrm{init}, \mathrm{upd})$ with

- $M = 2^J \times J \times \{0, 1\}$ where $J = \{1, \dots, k\}$,
- $\mathrm{init}(v) = (\{j \in I \mid v \in Q_j\}, 1, 0)$, and
- $\mathrm{upd}((R, c, b), v) = (R', c', b')$ with
  - $R' = (R \cup \{j \in I \mid v \in Q_j\}) \setminus \{j \in I \mid v \in P_j\}$,
  - $c' = \begin{cases} c & \text{if } c \in R \cap R' \\ (c \bmod k) + 1 & \text{otherwise.} \end{cases}$, and
  - $b' = \begin{cases} 0 & \text{if } c = c' \\ 1 & \text{otherwise} \end{cases}$.

Back to Request-response Games

Fix a request-response game

$$\mathcal{G} = ((V, V_0, V_1, E), \textsc{ReqRes}((Q_j, P_j)_{j=1,\dots,k}))$$

with (w.l.o.g.) $k \geq 2$ and define $\mathcal{M} = (M, \mathrm{init}, \mathrm{upd})$ with

- $M = 2^J \times J \times \{0, 1\}$ where $J = \{1, \dots, k\}$,
- $\mathrm{init}(v) = (\{j \in I \mid v \in Q_j\}, 1, 0)$, and
- $\mathrm{upd}((R, c, b), v) = (R', c', b')$ with
  - $R' = (R \cup \{j \in I \mid v \in Q_j\}) \setminus \{j \in I \mid v \in P_j\}$,
  - $c' = \begin{cases} c & \text{if } c \in R \cap R' \\ (c \bmod k) + 1 & \text{otherwise.} \end{cases}$, and
  - $b' = \begin{cases} 0 & \text{if } c = c' \\ 1 & \text{otherwise} \end{cases}$.

Now, define $\mathcal{G}' = (\mathcal{A} \times \mathcal{M}, \textsc{Büchi}(F))$ with $F = 2^J \times J \times \{1\}$.

Results

Lemma

$\mathcal{G} \leq_{\mathcal{M}} \mathcal{G}'$.

Results

Lemma
$\mathcal{G} \leq_{\mathcal{M}} \mathcal{G}'$.

Proof Sketch.
Show

$$\rho \in \textsc{ReqRes}((Q_j, P_j)_{j=1,\ldots,k}) \Leftrightarrow \mathrm{ext}(\rho) \in \textsc{Büchi}(F).$$

by proving the equivalence of the following two statements:

1. Every request in ρ is eventually answered.
2. The cyclic counter is updated infinitely often. □

Results

Lemma

$\mathcal{G} \leq_{\mathcal{M}} \mathcal{G}'$.

Proof Sketch.

Show

$$\rho \in \textsc{ReqRes}((Q_j, P_j)_{j=1,\ldots,k}) \Leftrightarrow \mathrm{ext}(\rho) \in \textsc{Büchi}(F).$$

by proving the equivalence of the following two statements:

1. Every request in ρ is eventually answered.
2. The cyclic counter is updated infinitely often. □

Corollary

Request-response games are determined with finite-state strategies of exponential size and can be solved in exponential time.

Results

Lemma

$\mathcal{G} \leq_{\mathcal{M}} \mathcal{G}'$.

Proof Sketch.

Show

$$\rho \in \textsc{ReqRes}((Q_j, P_j)_{j=1,\ldots,k}) \Leftrightarrow \mathrm{ext}(\rho) \in \textsc{Büchi}(F).$$

by proving the equivalence of the following two statements:

1. Every request in ρ is eventually answered.
2. The cyclic counter is updated infinitely often. □

Corollary

Request-response games are determined with finite-state strategies of exponential size and can be solved in exponential time.

Upper bounds on memory and complexity are tight.