Good-for-Games automata

Denis Kuperberg

CNRS, LIP, ENS Lyon

Journées SDA2, Saint-Etienne
04-07-2018

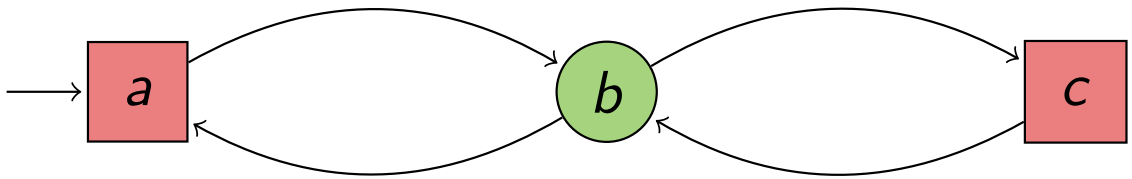
Games

Two Players:

Games

Two Players: Eve Adam

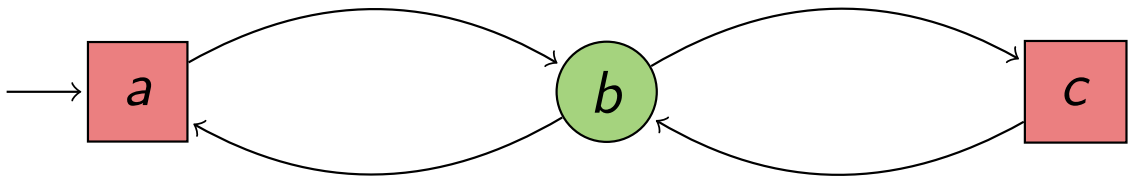
Arena: finite graph $G = (V, E)$, with $V = V_{\bigcirc} \uplus V_{\square}$.



Games

Two Players:

Arena: finite graph $G = (V, E)$, with $V = V_{\bigcirc} \uplus V_{\square}$.

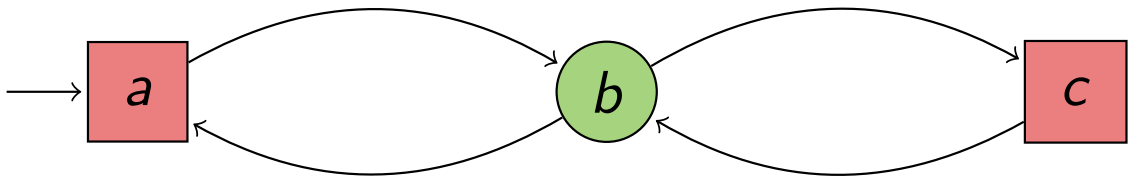


Initial vertex: $v_0 \in V$.

Games

Two Players: 

Arena: finite graph $G = (V, E)$, with $V = V_{\bigcirc} \uplus V_{\square}$.



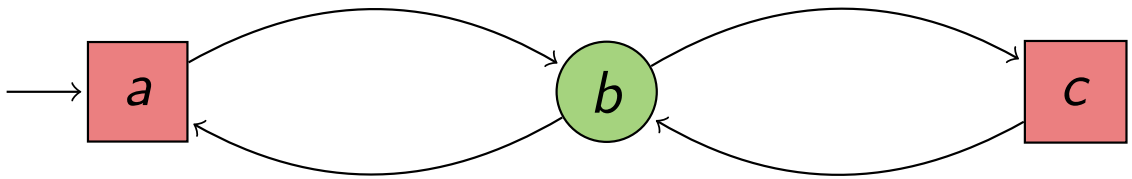
Initial vertex: $v_0 \in V$.

Play: Infinite path: $v_0 v_1 v_2 \cdots \in V^\omega$

Games

Two Players:

Arena: finite graph $G = (V, E)$, with $V = V_{\bigcirc} \uplus V_{\square}$.



Initial vertex: $v_0 \in V$.

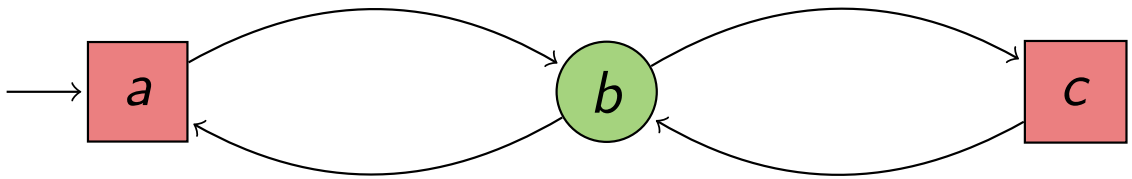
Play: Infinite path: $v_0 v_1 v_2 \cdots \in V^\omega$

Winning Condition: $W \subseteq V^\omega$.

Games

Two Players: Eve Adam

Arena: finite graph $G = (V, E)$, with $V = V_{\bigcirc} \uplus V_{\square}$.



Initial vertex: $v_0 \in V$.

Play: Infinite path: $v_0 v_1 v_2 \cdots \in V^\omega$

Winning Condition: $W \subseteq V^\omega$.

Eve wins a play π if $\pi \in W$

Strategies

Strategy for Eve: $\sigma_E : V^* V_{\bigcirc} \to V$,
$\sigma_E(v_0 v_1 \dots v_n)$ chooses what to play if $v_0 v_1 \dots v_n$ is the history of the play seen so far..

Strategies

Strategy for Eve: $\sigma_E : V^* V_{\bigcirc} \to V$,
$\sigma_E(v_0 v_1 \dots v_n)$ chooses what to play if $v_0 v_1 \dots v_n$ is the history of the play seen so far..

For Adam: $\sigma_A : V^* V_{\square} \to V$.

Strategies

Strategy for Eve: $\sigma_E : V^* V_{\bigcirc} \to V$,
$\sigma_E(v_0 v_1 \dots v_n)$ chooses what to play if $v_0 v_1 \dots v_n$ is the history of the play seen so far..

For Adam: $\sigma_A : V^* V_{\square} \to V$.

Two strategies σ_E, σ_A yield a unique play $\pi(\sigma_E, \sigma_A) = v_0 v_1 v_2 \dots$

- v_0 is the initial vertex,
- if $v_i \in V_{\bigcirc}$, then $v_{i+1} = \sigma_E(v_0 v_1 \dots v_i)$,
- if $v_i \in V_{\square}$, then $v_{i+1} = \sigma_A(v_0 v_1 \dots v_i)$,

Strategies

Strategy for Eve: $\sigma_E : V^* V_{\bigcirc} \to V$,
$\sigma_E(v_0 v_1 \dots v_n)$ chooses what to play if $v_0 v_1 \dots v_n$ is the history of the play seen so far..

For Adam: $\sigma_A : V^* V_{\square} \to V$.

Two strategies σ_E, σ_A yield a unique play $\pi(\sigma_E, \sigma_A) = v_0 v_1 v_2 \dots$

- v_0 is the initial vertex,
- if $v_i \in V_{\bigcirc}$, then $v_{i+1} = \sigma_E(v_0 v_1 \dots v_i)$,
- if $v_i \in V_{\square}$, then $v_{i+1} = \sigma_A(v_0 v_1 \dots v_i)$,

A strategy σ_E for Eve is winning if for all σ_A, $\pi(\sigma_E, \sigma_A) \in W$.

Strategies

Strategy for Eve: $\sigma_E : V^* V_{\bigcirc} \to V$,
$\sigma_E(v_0 v_1 \dots v_n)$ chooses what to play if $v_0 v_1 \dots v_n$ is the history of the play seen so far..

For Adam: $\sigma_A : V^* V_{\square} \to V$.

Two strategies σ_E, σ_A yield a unique play $\pi(\sigma_E, \sigma_A) = v_0 v_1 v_2 \dots$

- v_0 is the initial vertex,
- if $v_i \in V_{\bigcirc}$, then $v_{i+1} = \sigma_E(v_0 v_1 \dots v_i)$,
- if $v_i \in V_{\square}$, then $v_{i+1} = \sigma_A(v_0 v_1 \dots v_i)$,

A strategy σ_A for Adam is winning if for all σ_E, $\pi(\sigma_E, \sigma_A) \notin W$.

Strategies

Strategy for Eve: $\sigma_E : V^* V_{\bigcirc} \to V$,
$\sigma_E(v_0 v_1 \dots v_n)$ chooses what to play if $v_0 v_1 \dots v_n$ is the history of the play seen so far..

For Adam: $\sigma_A : V^* V_{\square} \to V$.

Two strategies σ_E, σ_A yield a unique play $\pi(\sigma_E, \sigma_A) = v_0 v_1 v_2 \dots$

- v_0 is the initial vertex,
- if $v_i \in V_{\bigcirc}$, then $v_{i+1} = \sigma_E(v_0 v_1 \dots v_i)$,
- if $v_i \in V_{\square}$, then $v_{i+1} = \sigma_A(v_0 v_1 \dots v_i)$,

A strategy σ_A for Adam is winning if for all σ_E, $\pi(\sigma_E, \sigma_A) \notin W$.
The game is determined if a player has a winning strategy.

Strategies

Strategy for Eve: $\sigma_E : V^* V_{\bigcirc} \to V$,
$\sigma_E(v_0 v_1 \dots v_n)$ chooses what to play if $v_0 v_1 \dots v_n$ is the history of the play seen so far..

For Adam: $\sigma_A : V^* V_{\square} \to V$.

Two strategies σ_E, σ_A yield a unique play $\pi(\sigma_E, \sigma_A) = v_0 v_1 v_2 \dots$

- v_0 is the initial vertex,
- if $v_i \in V_{\bigcirc}$, then $v_{i+1} = \sigma_E(v_0 v_1 \dots v_i)$,
- if $v_i \in V_{\square}$, then $v_{i+1} = \sigma_A(v_0 v_1 \dots v_i)$,

A strategy σ_A for Adam is winning if for all σ_E, $\pi(\sigma_E, \sigma_A) \notin W$.
The game is determined if a player has a winning strategy.

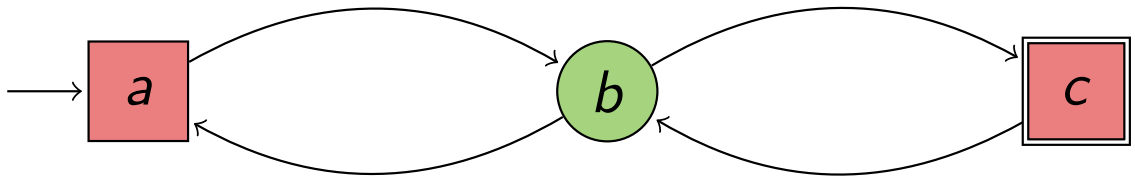
Positional strategy: $\sigma_X : V_X \to V$, for $X \in \{E, A\}$.

Winning conditions

Reachability Condition

Goal for Eve: reach a subset $F \subseteq V$. I.e. $W = V^* F V^\omega$.

Example with $F = \{c\}$:

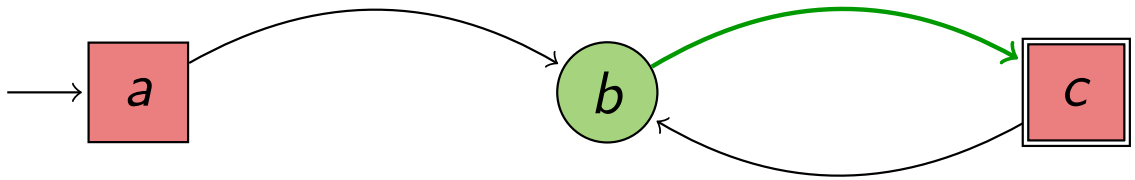


Winning conditions

Reachability Condition

Goal for Eve: reach a subset $F \subseteq V$. I.e. $W = V^* F V^\omega$.

Example with $F = \{c\}$:



Here: Eve wins with positional strategy.

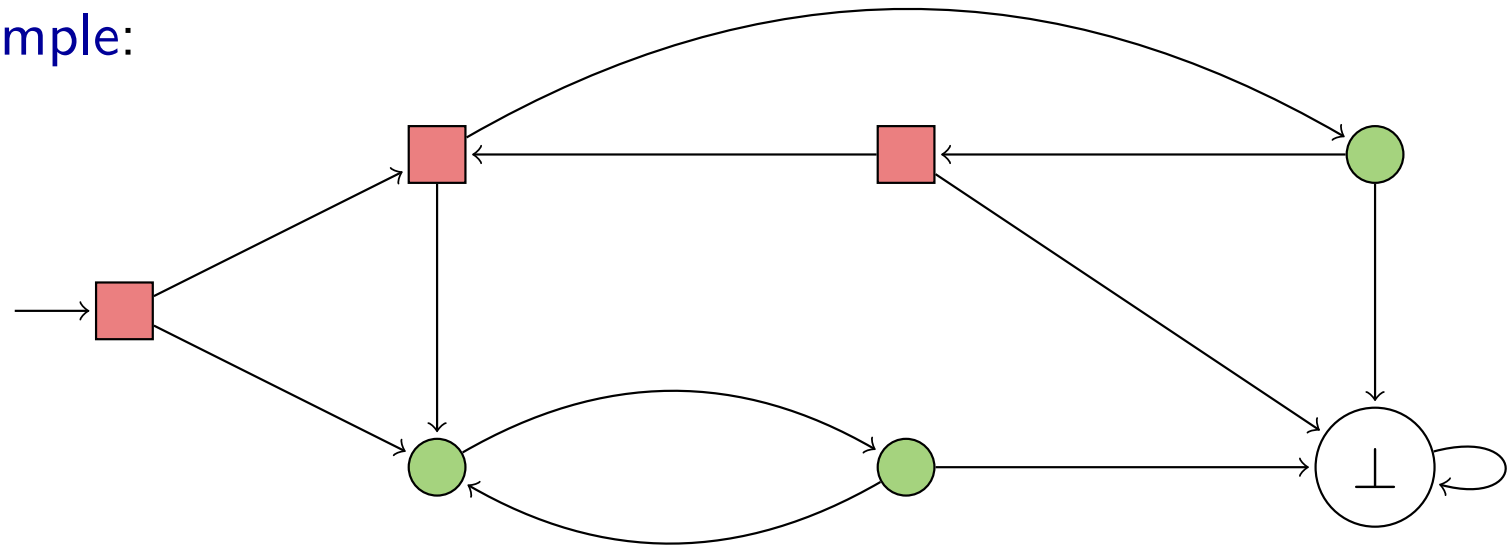
Winning conditions

Safety Condition

Goal for Eve: stay in a subset $F \subseteq V$. I.e. $W = F^\omega$.

Dual of Reachability

Example:



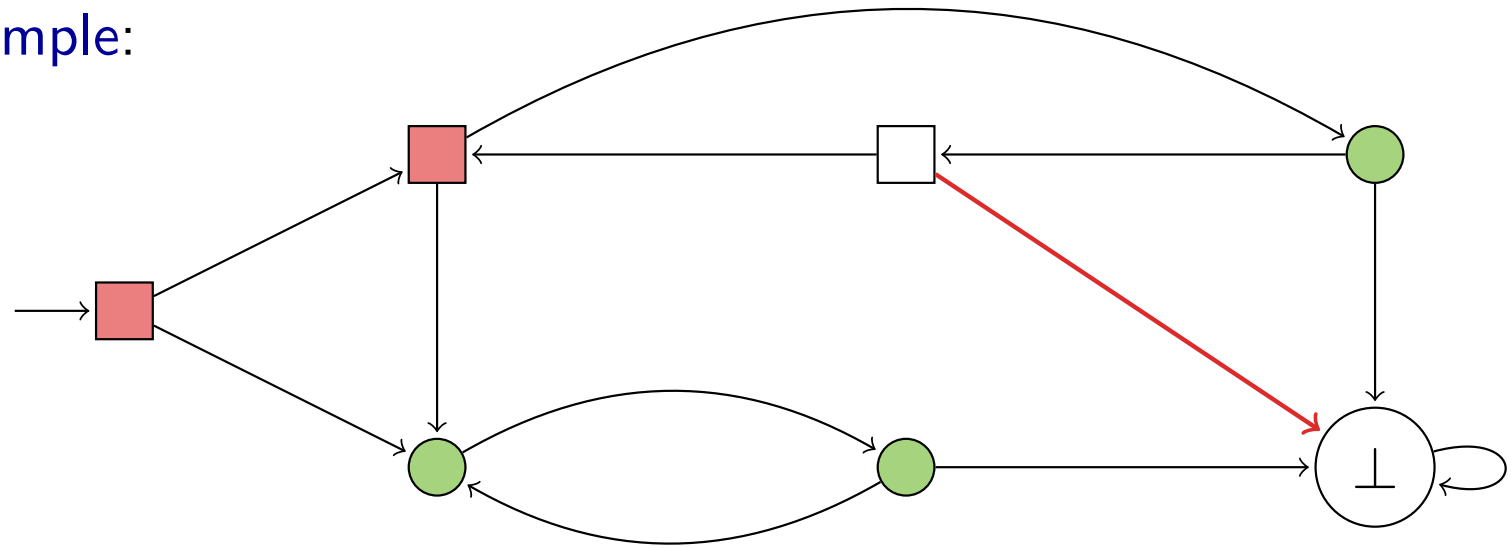
Winning conditions

Safety Condition

Goal for Eve: stay in a subset $F \subseteq V$. I.e. $W = F^\omega$.

Dual of Reachability

Example:



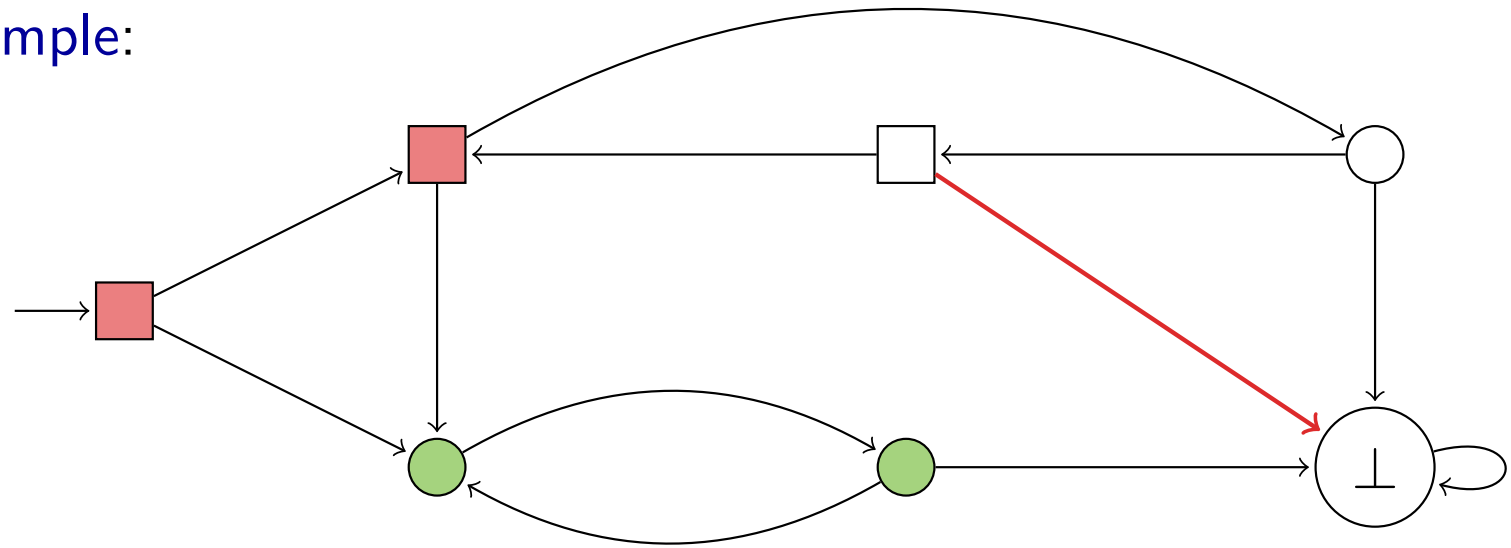
Winning conditions

Safety Condition

Goal for Eve: stay in a subset $F \subseteq V$. I.e. $W = F^\omega$.

Dual of Reachability

Example:



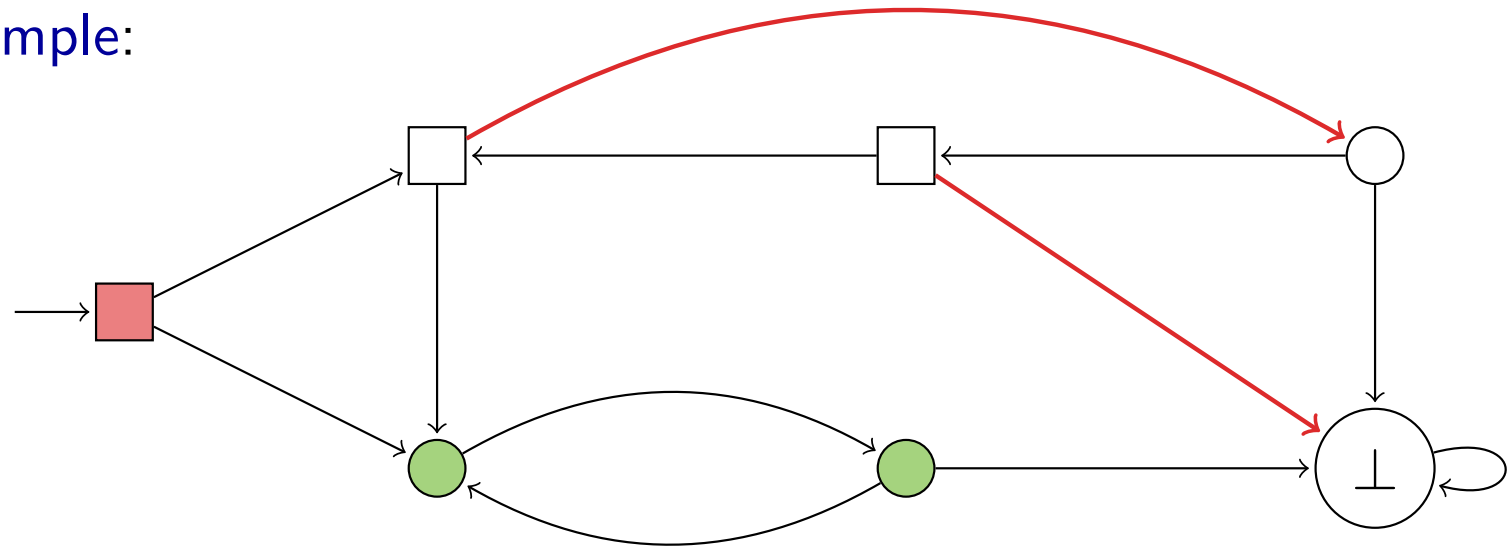
Winning conditions

Safety Condition

Goal for Eve: stay in a subset $F \subseteq V$. I.e. $W = F^\omega$.

Dual of Reachability

Example:



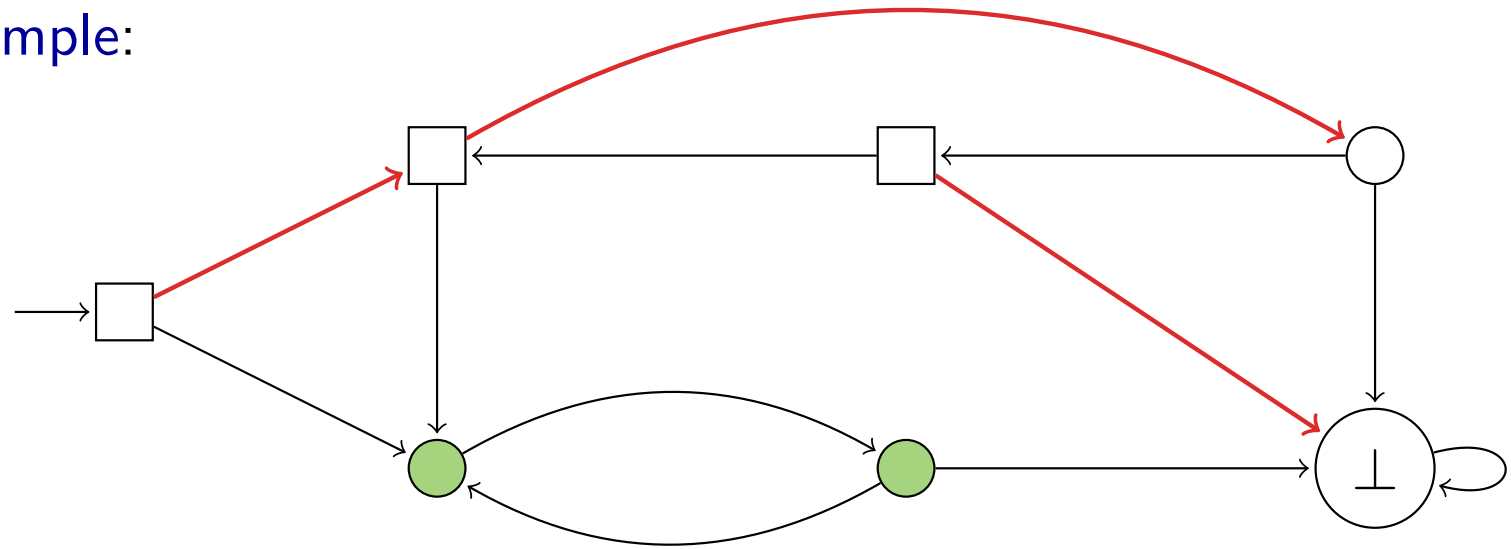
Winning conditions

Safety Condition

Goal for Eve: stay in a subset $F \subseteq V$. I.e. $W = F^\omega$.

Dual of Reachability

Example:



$\rightarrow$ Algorithm for solving Reachability/Safety Games in P.

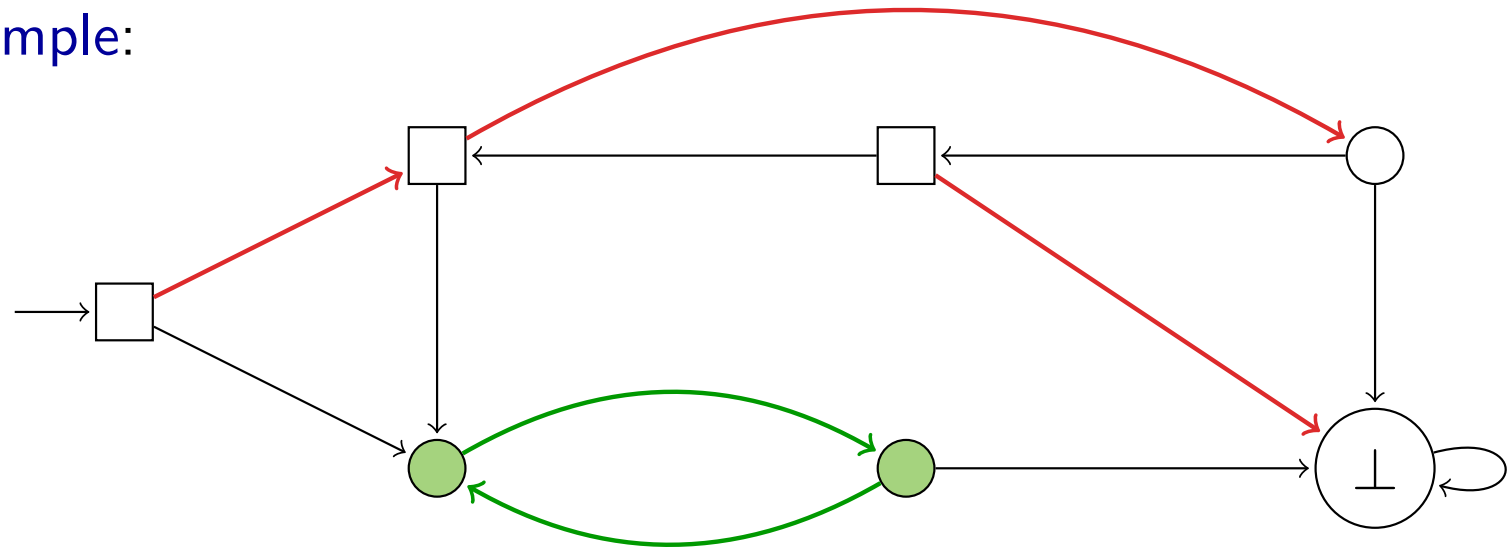
Winning conditions

Safety Condition

Goal for Eve: stay in a subset $F \subseteq V$. I.e. $W = F^\omega$.

Dual of Reachability

Example:



$\rightarrow$ Algorithm for solving Reachability/Safety Games in P.

Corollary: Reachability and Safety games are positionally determined.

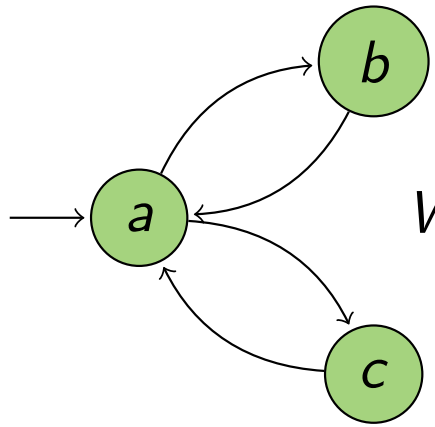
Arbitrary conditions

What about arbitrary $W \subseteq V^\omega$? Are they determined ?
Positionally determined ?

Arbitrary conditions

What about arbitrary $W \subseteq V^\omega$? Are they determined ? Positionally determined ?

Non-example for positional determinacy:

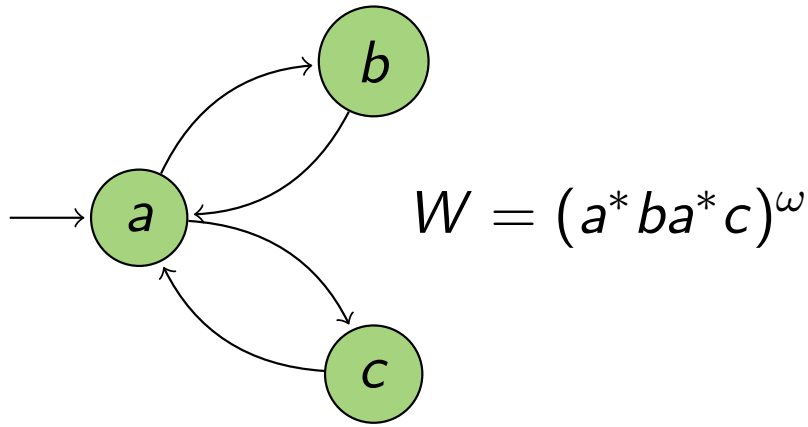


$W = (a^* b a^* c)^\omega$

Arbitrary conditions

What about arbitrary $W \subseteq V^\omega$? Are they determined ? Positionally determined ?

Non-example for positional determinacy:



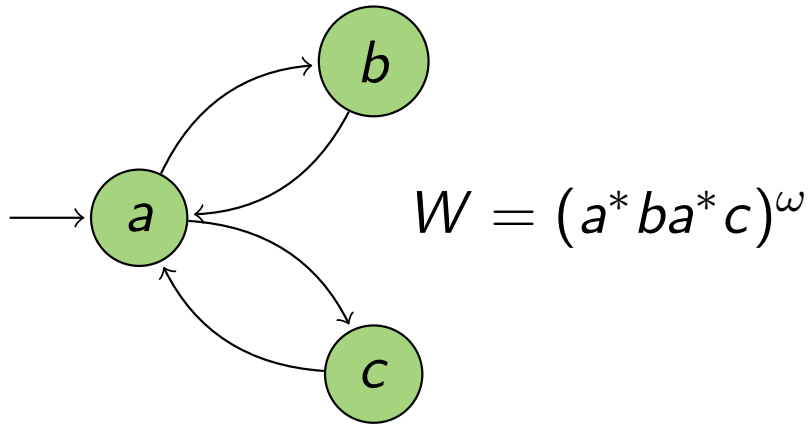
Theorem (Martin 1975)

*Any game with a **Borel** acceptance condition is determined.*

Arbitrary conditions

What about arbitrary $W \subseteq V^\omega$? Are they determined ?
Positionally determined ?

Non-example for positional determinacy:



Theorem (Martin 1975)

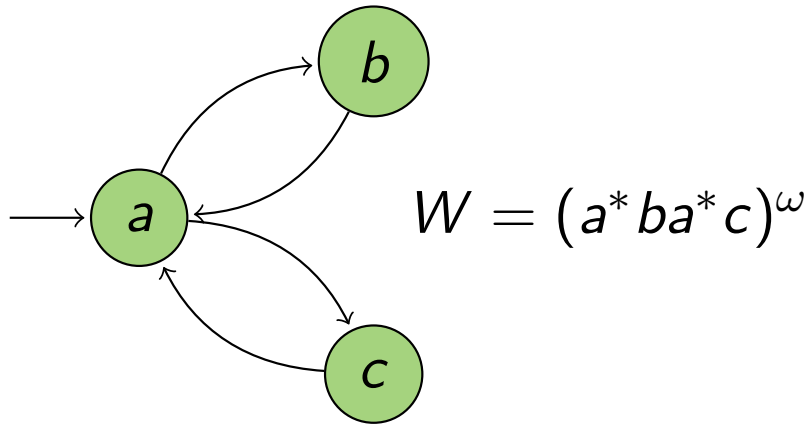
*Any game with a **Borel** acceptance condition is determined.*

Cantor Topology Reachability: **Open** Safety: **Closed**

Arbitrary conditions

What about arbitrary $W \subseteq V^\omega$? Are they determined ?
Positionally determined ?

Non-example for positional determinacy:



Theorem (Martin 1975)

*Any game with a **Borel** acceptance condition is determined.*

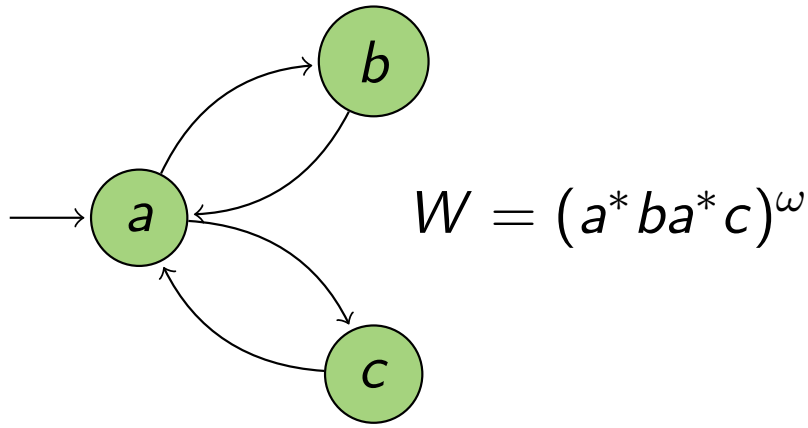
Cantor Topology Reachability: **Open** Safety: **Closed**

And in general ? Could all games be determined ?

Arbitrary conditions

What about arbitrary $W \subseteq V^\omega$? Are they determined ?
Positionally determined ?

Non-example for positional determinacy:



Theorem (Martin 1975)

*Any game with a **Borel** acceptance condition is determined.*

Cantor Topology Reachability: **Open** Safety: **Closed**

And in general ? Could all games be determined ?

Theorem

*The **Axiom of determinacy** is uncompatible with ZFC.*
In ZF+DC, it is equivalent to some large cardinals axioms.

Parity conditions

- Büchi condition: $W = (V^*F)^\omega$, i.e. "∞ many F"

Parity conditions

- Büchi condition: $W = (V^*F)^\omega$, i.e. "∞ many F"
- coBüchi condition: $W = V^*F^\omega$, i.e. "finitely many $\overline{F}$"

.

Parity conditions

- Büchi condition: $W = (V^* F)^\omega$, i.e. "∞ many F"
- coBüchi condition: $W = V^* F^\omega$, i.e. "finitely many $\overline{F}$"
- $[i,j]$-Parity condition: $\mathrm{rank} : V \to \{i, i+1, \ldots, j\}$
  $v_0 v_1 v_2 \cdots \in W$ iff "maximal rank appearing ∞ often is even"
  $\max\{r \mid \exists\infty \text{ many } i \text{ such that } \mathrm{rank}(v_i) = r\}$ is even.

.

Parity conditions

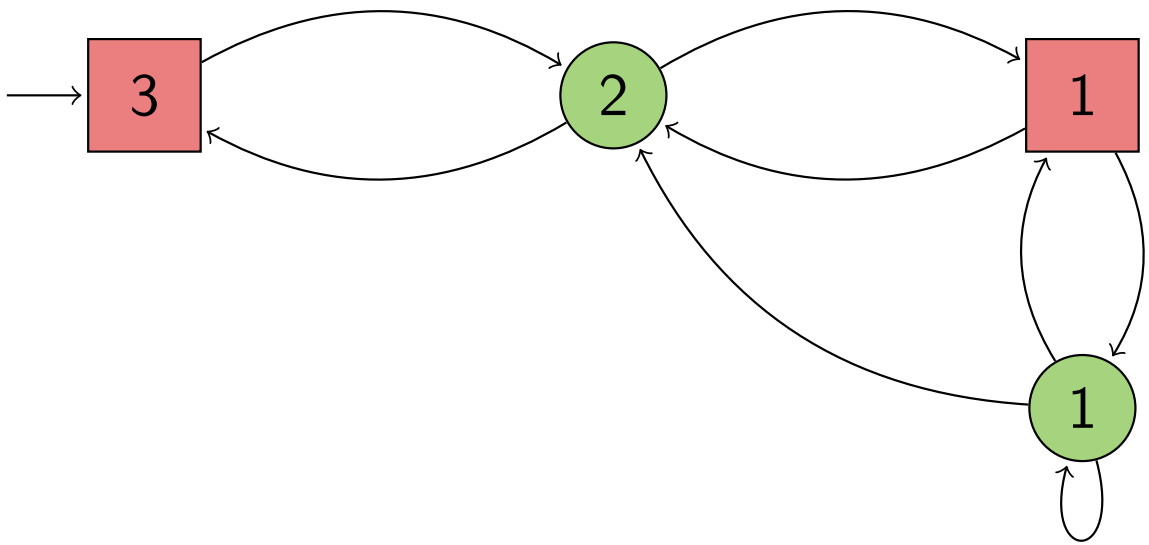
- $[1,2]$-parity condition: $W = (V^*F)^\omega$, i.e. "∞ many F"
- $[0,1]$-parity condition: $W = V^*F^\omega$, i.e. "finitely many $\overline{F}$"
- $[i,j]$-Parity condition: $\mathrm{rank} : V \to \{i, i+1, \ldots, j\}$
  $v_0 v_1 v_2 \cdots \in W$ iff "maximal rank appearing ∞ often is even"
  $\max\{r \mid \exists\infty \text{ many } i \text{ such that } \mathrm{rank}(v_i) = r\}$ is even.

.

Parity conditions

- $[1,2]$-parity condition: $W = (V^*F)^\omega$, i.e. "∞ many F"
- $[0,1]$-parity condition: $W = V^*F^\omega$, i.e. "finitely many $\overline{F}$"
- $[i,j]$-Parity condition: $\mathrm{rank} : V \to \{i, i+1, \ldots, j\}$
  $v_0 v_1 v_2 \cdots \in W$ iff "maximal rank appearing ∞ often is even"
  $\max\{r \mid \exists\infty \text{ many } i \text{ such that } \mathrm{rank}(v_i) = r\}$ is even.

Example:

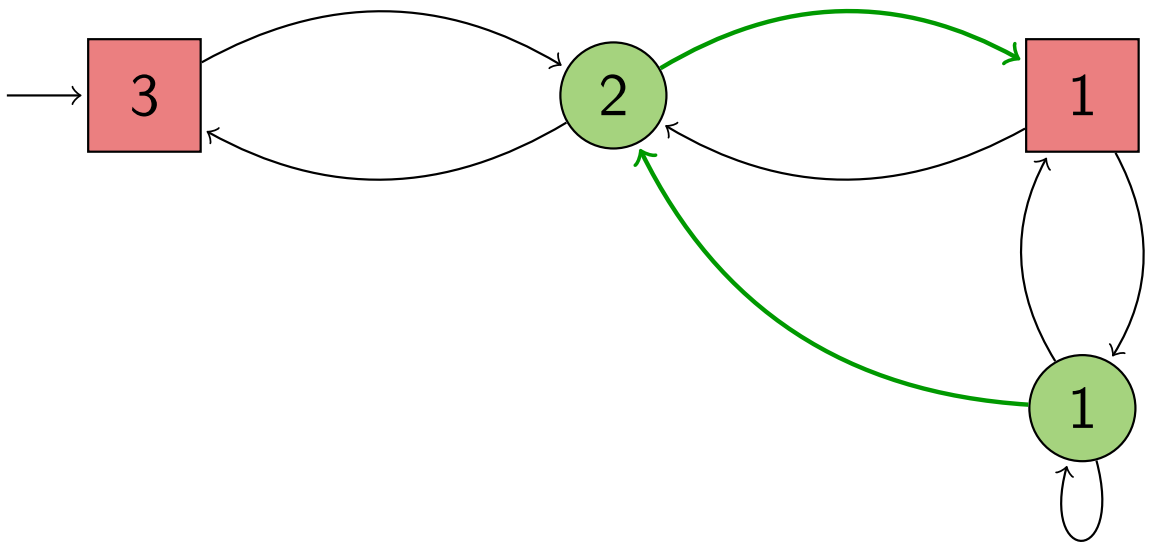


.

Parity conditions

- $[1,2]$-parity condition: $W = (V^*F)^\omega$, i.e. "∞ many F"
- $[0,1]$-parity condition: $W = V^*F^\omega$, i.e. "finitely many $\overline{F}$"
- $[i,j]$-Parity condition: $\mathrm{rank} : V \to \{i, i+1, \ldots, j\}$
  $v_0 v_1 v_2 \cdots \in W$ iff "maximal rank appearing ∞ often is even"
  $\max\{r \mid \exists\infty \text{ many } i \text{ such that } \mathrm{rank}(v_i) = r\}$ is even.

Example:



Here: Eve wins positionally.

Results on parity games

Theorem (Rabin 1969)

*Parity games **positionally** determined.*

This is even true for infinite graphs.

Results on parity games

Theorem (Rabin 1969)

*Parity games **positionally** determined.*

This is even true for infinite graphs.

Complexity ?

Results on parity games

Theorem (Rabin 1969)

Parity games ***positionally*** *determined.*

This is even true for infinite graphs.

Complexity ?

Theorem (Zielonka 1998)

Parity games with d ranks can be solved in $O(n^d)$.

Idea: nesting of the previous algorithm.

Results on parity games

Theorem (Rabin 1969)

Parity games ***positionally*** *determined.*

This is even true for infinite graphs.

Complexity ?

Theorem (Zielonka 1998)

Parity games with d ranks can be solved in $O(n^d)$.

Idea: nesting of the previous algorithm.

Remark: From Rabin, Parity Games are in **NP**$\cap$**coNP**

Results on parity games

Theorem (Rabin 1969)

Parity games ***positionally*** *determined.*

This is even true for infinite graphs.

Complexity ?

Theorem (Zielonka 1998)

Parity games with d ranks can be solved in $O(n^d)$.

Idea: nesting of the previous algorithm.

Remark: From Rabin, Parity Games are in **NP**$\cap$**coNP**

Theorem (Calude, Jain, Khoussainov, Li, Stephan 2017)

Parity games can be solved in Quasi-Polynomial time: $O(n^{\log n+6})$.
For any fixed $[i,j]$, they can be solved in $O(n^5)$.

ω-regular languages

Parity automata on infinite words:

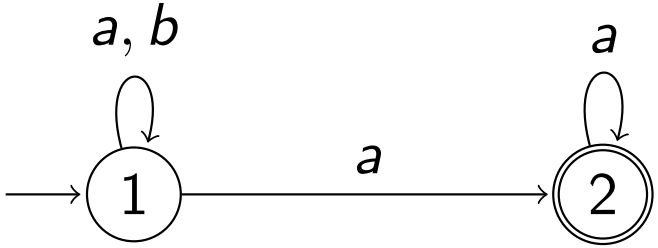
- can be deterministic (Det) or non-deterministic (ND)
- a run is a sequence of state $q_1 q_2 \cdots \in Q^\omega$
- labelling function $\mathrm{rk} : Q \to [i, j]$
- run accepting if parity condition satisfied
- $L(\mathcal{A}) = \{u \in V^\omega \mid \text{ there is an accepting run on } u\}$.

ω-regular languages

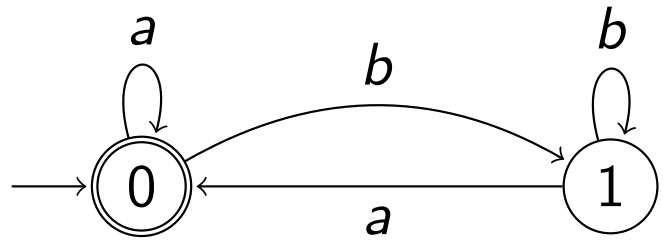
Parity automata on infinite words:

- can be deterministic (Det) or non-deterministic (ND)
- a run is a sequence of state $q_1 q_2 \cdots \in Q^\omega$
- labelling function $\mathrm{rk} : Q \to [i, j]$
- run accepting if parity condition satisfied
- $L(\mathcal{A}) = \{u \in V^\omega \mid$ there is an accepting run on $u\}$.

Example: Parity automata for $L = (a + b)^* a^\omega$.



ND Büchi



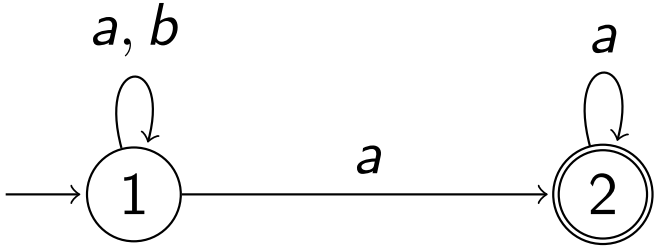
Det coBüchi

ω-regular languages

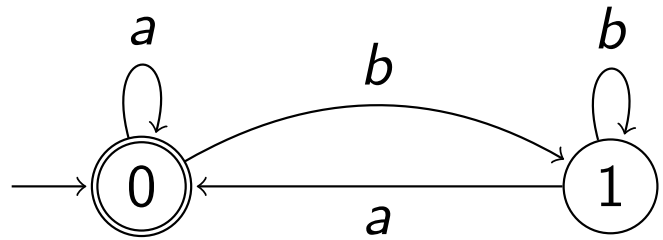
Parity automata on infinite words:

- can be deterministic (Det) or non-deterministic (ND)
- a run is a sequence of state $q_1 q_2 \cdots \in Q^\omega$
- labelling function $\mathrm{rk} : Q \to [i, j]$
- run accepting if parity condition satisfied
- $L(\mathcal{A}) = \{u \in V^\omega \mid$ there is an accepting run on $u\}$.

Example: Parity automata for $L = (a + b)^* a^\omega$.



ND Büchi



Det coBüchi

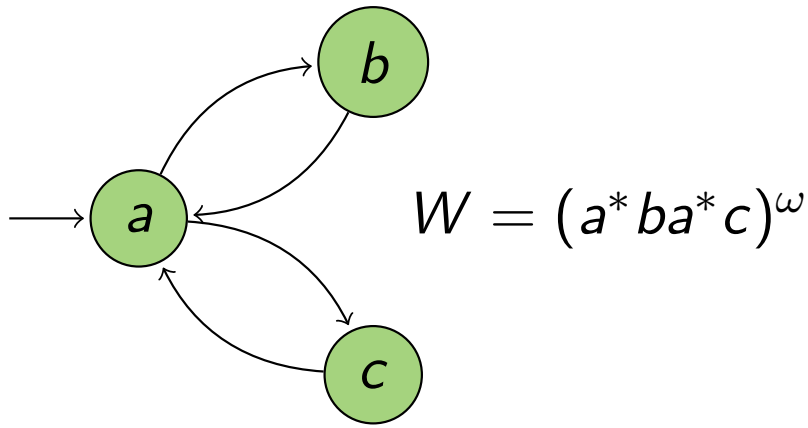
Theorem (McNaughton 1966)

ND Büchi $\Leftrightarrow$ *Det Parity* $\Leftrightarrow$ *ND Parity* ($\Leftrightarrow \ldots$)

Languages as winning conditions

What if the winning W condition of G can be any ω-regular language ?

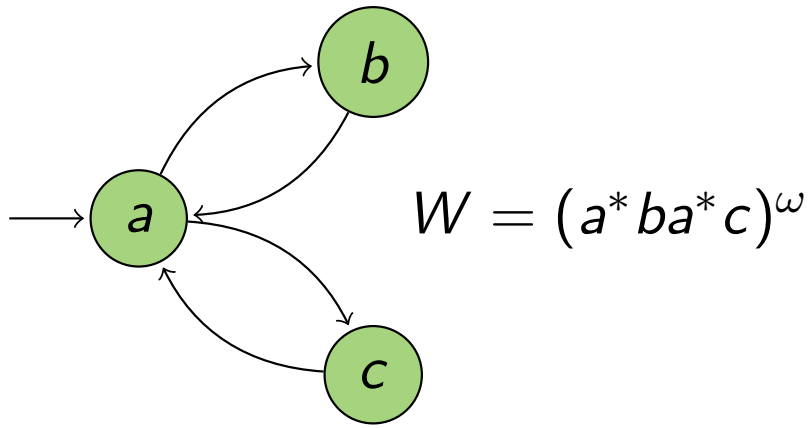
Example:



Languages as winning conditions

What if the winning W condition of G can be any ω-regular language ?

Example:

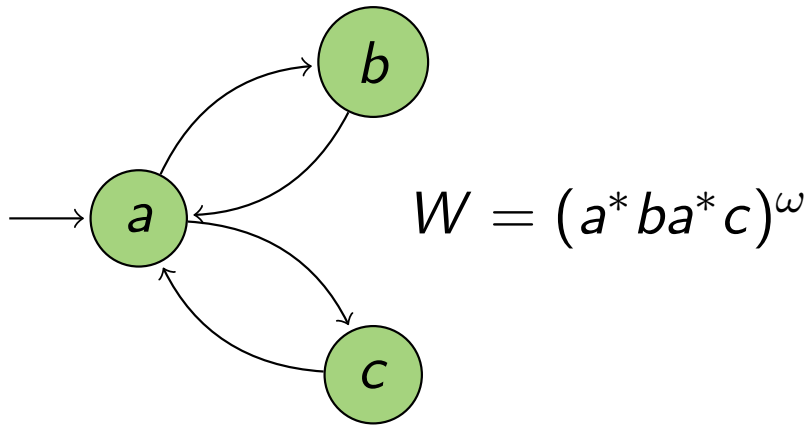


Why study these games ?

Languages as winning conditions

What if the winning W condition of G can be any ω-regular language ?

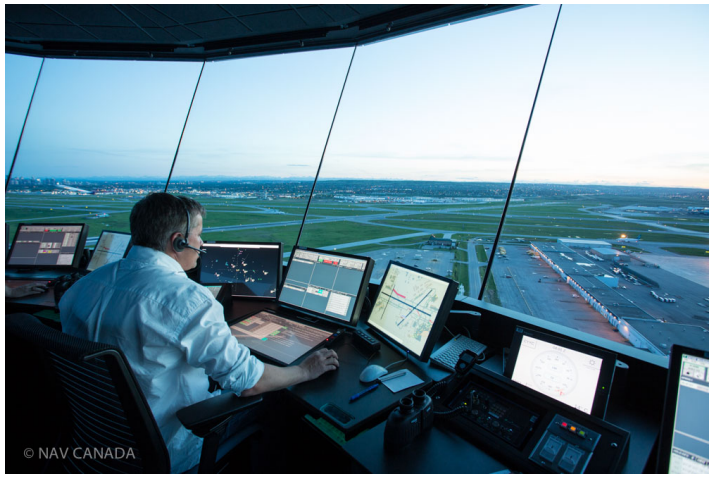
Example:



Why study these games ?

Applications in Logic, Verification, Proof theory, Complexity, ...

Church's synthesis problem (1962)



$\longleftarrow I_1$

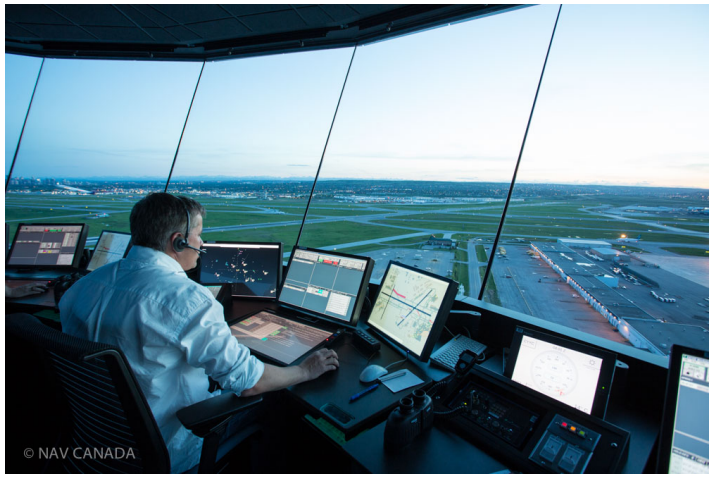
$\Longrightarrow O_1$

$\longleftarrow I_2$

$\Longrightarrow O_2$

Desired algorithm: Specification $\varphi \rightsquigarrow$ Controller C
such that $\forall I_1 I_2 \ldots$, we have $I_1 O_1 I_2 O_2 \cdots \in L(\varphi)$.

Church's synthesis problem (1962)



$\longleftarrow I_1$

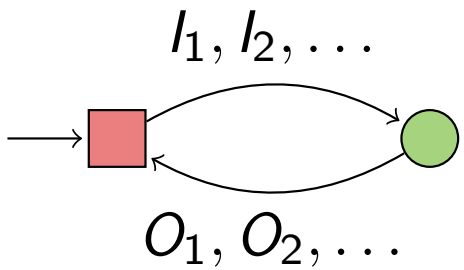
$\Longrightarrow O_1$

$\longleftarrow I_2$

$\Longrightarrow O_2$

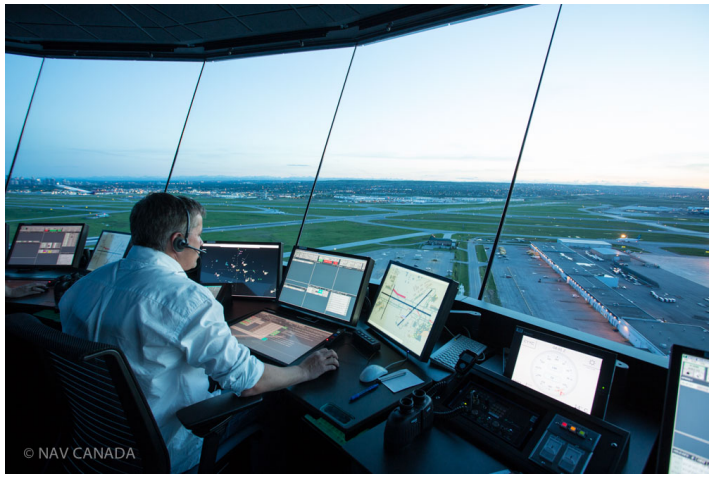
Desired algorithm: Specification $\varphi \rightsquigarrow$ Controller C
such that $\forall I_1 I_2 \ldots$, we have $I_1 O_1 I_2 O_2 \cdots \in L(\varphi)$.

Solution: Solve the following game:



$W = L(\varphi)$

Church's synthesis problem (1962)



$\longleftarrow I_1$

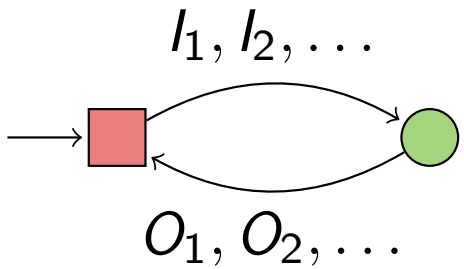
$\Longrightarrow O_1$

$\longleftarrow I_2$

$\Longrightarrow O_2$

Desired algorithm: Specification $\varphi \rightsquigarrow$ Controller C
such that $\forall I_1 I_2 \ldots$, we have $I_1 O_1 I_2 O_2 \cdots \in L(\varphi)$.

Solution: Solve the following game:



$W = L(\varphi)$

Strategy $\rightsquigarrow$ Controller

Solving an ω-regular game

Input: G game with ω-regular winning condition $W \subseteq V^\omega$.
Question: Who wins G ? How ?

Solving an ω-regular game

Input: G game with ω-regular winning condition $W \subseteq V^\omega$.
Question: Who wins G ? How ?
Solution: Turn G into a game with more states but simpler acceptance condition.

Solving an ω-regular game

Input: G game with ω-regular winning condition $W \subseteq V^\omega$.
Question: Who wins G ? How ?
Solution: Turn G into a game with more states but simpler acceptance condition.

1. Build Det Parity automaton $\mathcal{A}_{\mathrm{Det}}$ for W,
2. Solve the parity game $G' = G \circ \mathcal{A}_{\mathrm{Det}}$.

Solving an ω-regular game

Input: G game with ω-regular winning condition $W \subseteq V^\omega$.
Question: Who wins G ? How ?
Solution: Turn G into a game with more states but simpler acceptance condition.

1. Build Det Parity automaton $\mathcal{A}_{\mathrm{Det}}$ for W,
2. Solve the parity game $G' = G \circ \mathcal{A}_{\mathrm{Det}}$.

What is $G \circ \mathcal{A}_{\mathrm{Det}}$?

- States: $V' = V \times Q_{\mathrm{Det}}$,
- Initial state: $v'_0 = (v_0, \delta(q_0, v_0))$,
- Edges: $E' = \{(v, q) \to (v', \delta(q, v') \mid v \to v' \in E,\ q \in Q_{\mathrm{Det}}\}$
- Winning condition: inherited from $\mathcal{A}_{\mathrm{Det}}$.

Solving an ω-regular game

Input: G game with ω-regular winning condition $W \subseteq V^\omega$.
Question: Who wins G ? How ?
Solution: Turn G into a game with more states but simpler acceptance condition.

1. Build Det Parity automaton $\mathcal{A}_{\mathrm{Det}}$ for W,
2. Solve the parity game $G' = G \circ \mathcal{A}_{\mathrm{Det}}$.

What is $G \circ \mathcal{A}_{\mathrm{Det}}$?

- States: $V' = V \times Q_{\mathrm{Det}}$,
- Initial state: $v'_0 = (v_0, \delta(q_0, v_0))$,
- Edges: $E' = \{(v, q) \to (v', \delta(q, v') \mid v \to v' \in E,\ q \in Q_{\mathrm{Det}}\}$
- Winning condition: inherited from $\mathcal{A}_{\mathrm{Det}}$.

Theorem
$G \circ \mathcal{A}_{\mathrm{Det}}$ has same winner as G.

Do we need to determinize ?

Theorem

When $\mathcal{A}$ is deterministic, $G \circ \mathcal{A}$ has same winner as G.

Problem: Determinization is expensive. Maybe too strong ?

Do we need to determinize ?

Theorem

When $\mathcal{A}$ is deterministic, $G \circ \mathcal{A}$ has same winner as G.

Problem: Determinization is expensive. Maybe too strong ?

Definition (Henzinger, Piterman 2006)

$\mathcal{A}$ is Good-for-Games (GFG) if $G \circ \mathcal{A}$ has same winner as $\mathcal{A}$, for any game G.

Do we need to determinize ?

Theorem

When $\mathcal{A}$ is deterministic, $G \circ \mathcal{A}$ has same winner as G.

Problem: Determinization is expensive. Maybe too strong ?

Definition (Henzinger, Piterman 2006)

$\mathcal{A}$ is Good-for-Games (GFG) if $G \circ \mathcal{A}$ has same winner as $\mathcal{A}$, for any game G.

- Weakening of determinism,
- Sufficient to solve Church's synthesis, ω-regular games
  New Algorithm: $\varphi \to \mathcal{A}_{\mathrm{GFG}}$, then solve $G \circ \mathcal{A}_{\mathrm{GFG}}$.
- What are these automata ?

Do we need to determinize ?

Theorem

When $\mathcal{A}$ is deterministic, $G \circ \mathcal{A}$ has same winner as G.

Problem: Determinization is expensive. Maybe too strong ?

Definition (Henzinger, Piterman 2006)

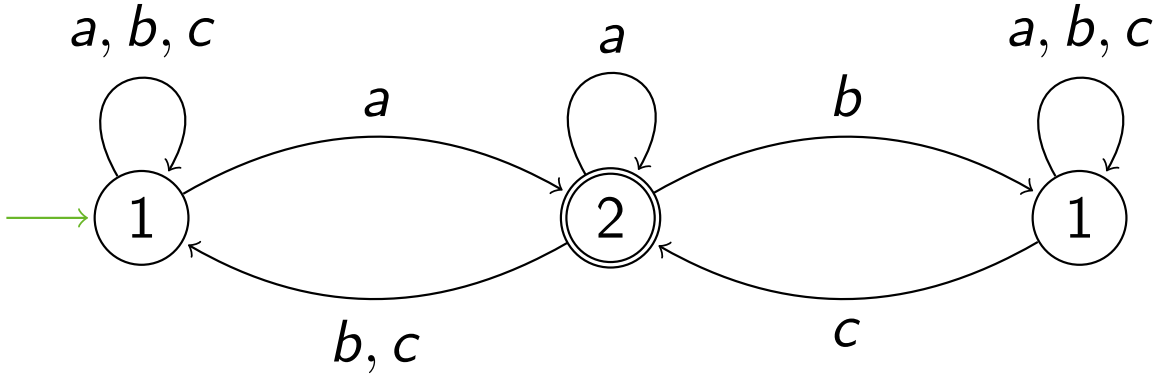
$\mathcal{A}$ is Good-for-Games (GFG) if $G \circ \mathcal{A}$ has same winner as $\mathcal{A}$, for any game G.

- Weakening of determinism, **Really ?**
- Sufficient to solve Church's synthesis, ω-regular games
  New Algorithm: $\varphi \to \mathcal{A}_{\mathrm{GFG}}$, then solve $G \circ \mathcal{A}_{\mathrm{GFG}}$.
- What are these automata ?

Definition of GFG via a game

$\mathcal{A}$ Parity automaton, we associate to it a **GFG game**:
Adam plays letters:

Eve: resolves non-deterministic choices for transitions

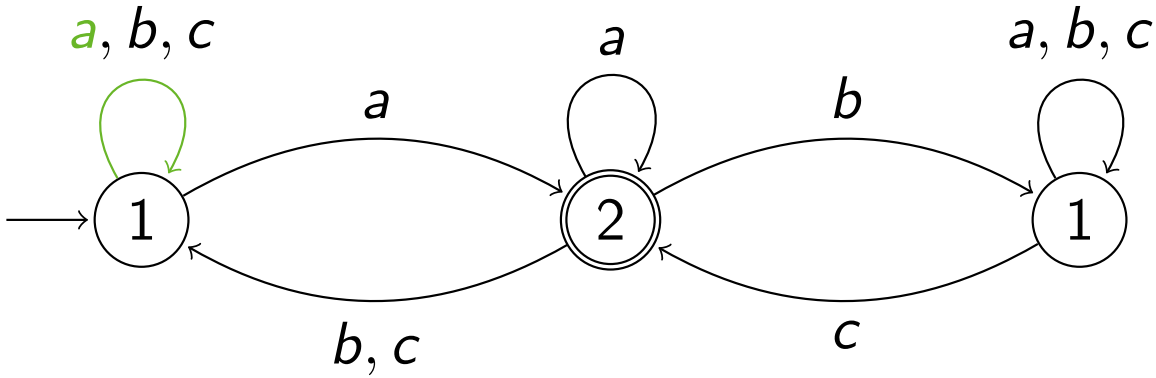


Definition of GFG via a game

$\mathcal{A}$ Parity automaton, we associate to it a **GFG game**:

Adam plays letters: a

Eve: resolves non-deterministic choices for transitions

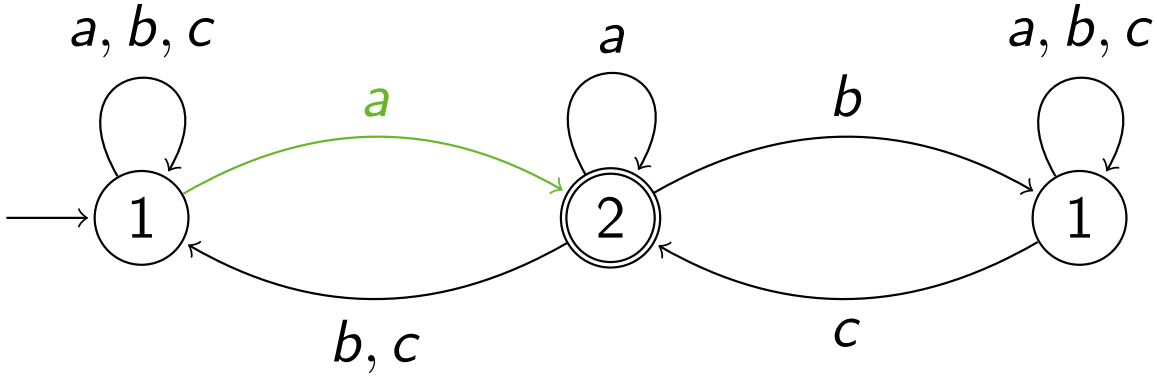


Definition of GFG via a game

$\mathcal{A}$ Parity automaton, we associate to it a **GFG game**:

Adam plays letters: a a

Eve: resolves non-deterministic choices for transitions

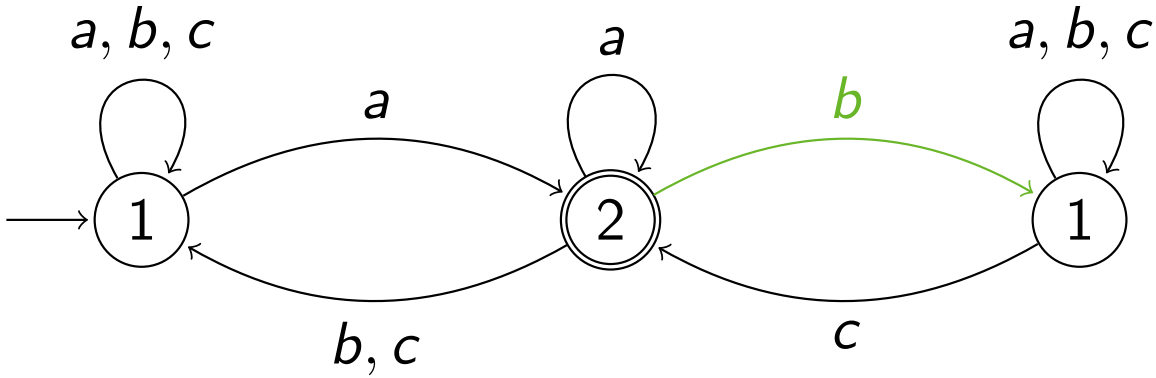


Definition of GFG via a game

$\mathcal{A}$ Parity automaton, we associate to it a **GFG game**:

Adam plays letters: $a \quad a \quad b$

Eve: resolves non-deterministic choices for transitions

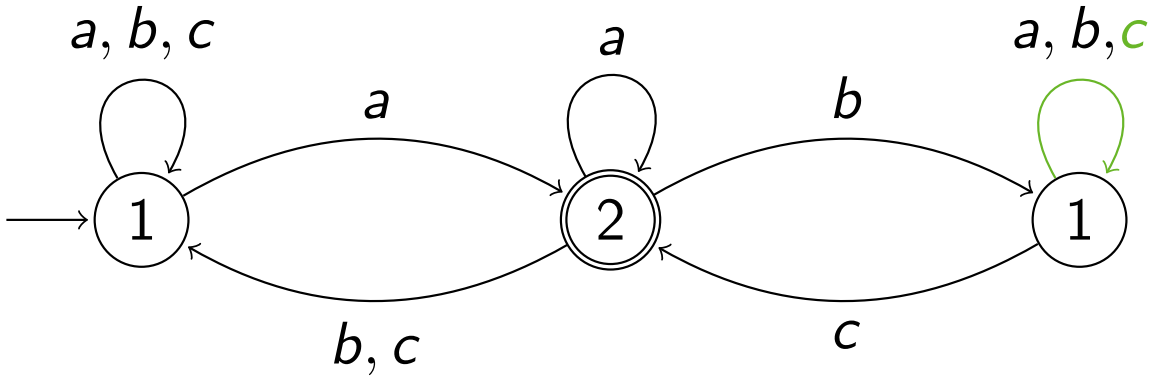


Definition of GFG via a game

$\mathcal{A}$ Parity automaton, we associate to it a **GFG game**:

Adam plays letters: $a \quad a \quad b \quad c$

Eve: resolves non-deterministic choices for transitions

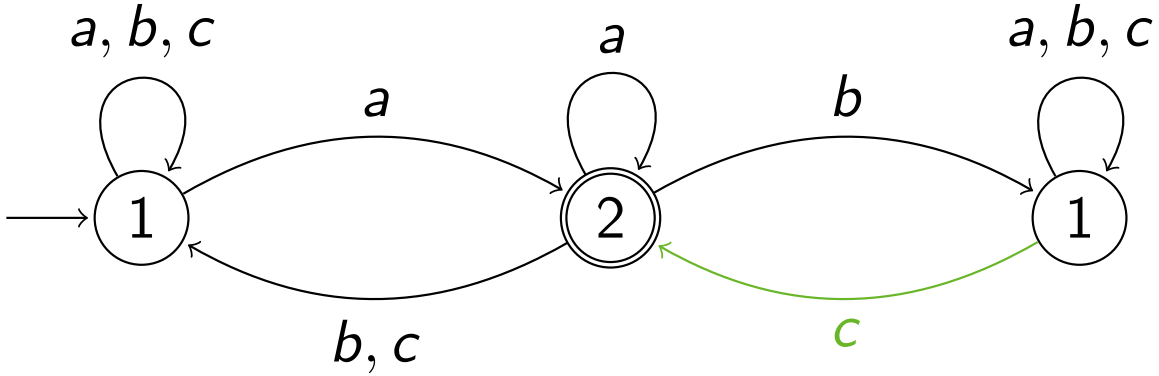


Definition of GFG via a game

$\mathcal{A}$ Parity automaton, we associate to it a **GFG game**:

Adam plays letters: $a \quad a \quad b \quad c \quad c$

Eve: resolves non-deterministic choices for transitions

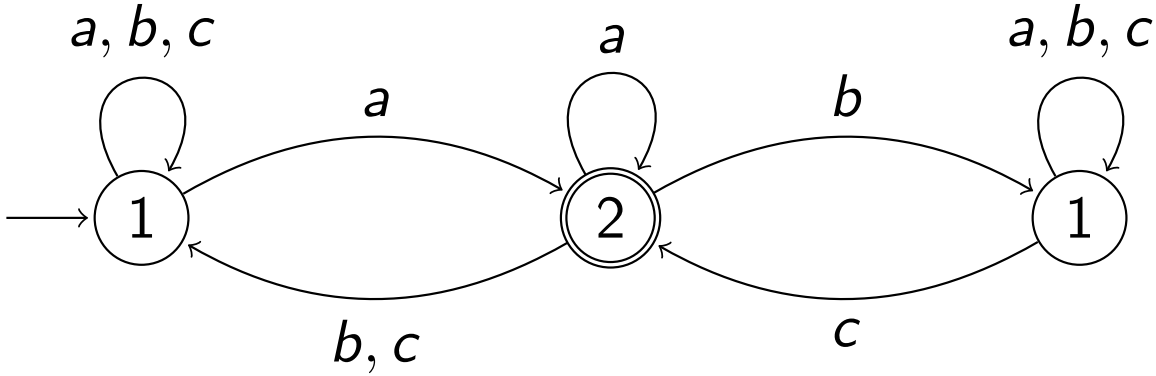


Definition of GFG via a game

$\mathcal{A}$ Parity automaton, we associate to it a **GFG game**:

Adam plays letters: $a \quad a \quad b \quad c \quad c \quad \ldots \quad = \quad w$

Eve: resolves non-deterministic choices for transitions

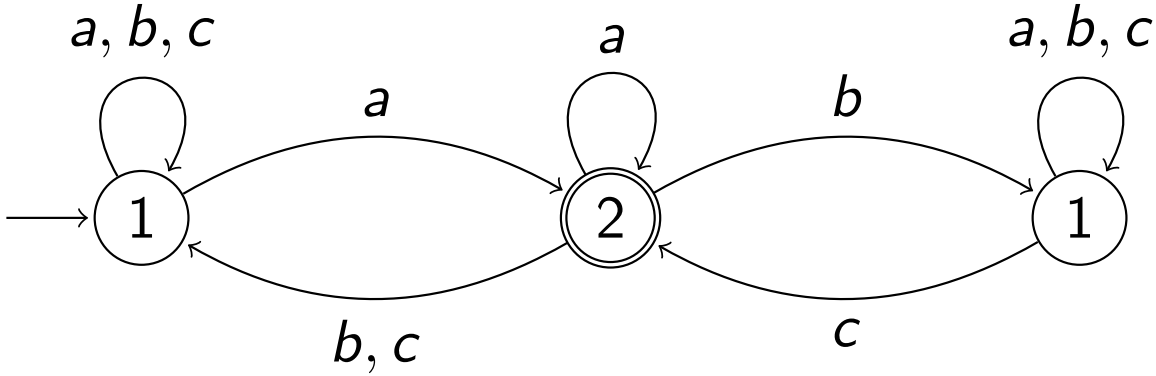


Eve wins if: $w \in L \Rightarrow$ Run accepting.

Definition of GFG via a game

$\mathcal{A}$ Parity automaton, we associate to it a **GFG game**:
Adam plays letters: $a \quad a \quad b \quad c \quad c \quad \ldots \quad = \quad w$

Eve: resolves non-deterministic choices for transitions



Eve wins if: $w \in L \Rightarrow$ Run accepting.

$\mathcal{A}$ **GFG** means that there is a strategy $\sigma_{\mathrm{GFG}} : A^* \to Q$, for accepting all words of $L(\mathcal{A})$.

A few facts on GFG automata

Fact

Every deterministic automaton is **GFG**.

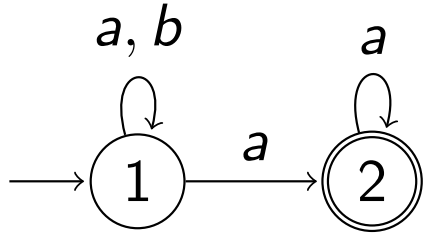
A few facts on GFG automata

Fact

Every deterministic automaton is **GFG**.

Fact

*There are some non-*__GFG__ *automata:*



$(a+b)^* a^\omega$

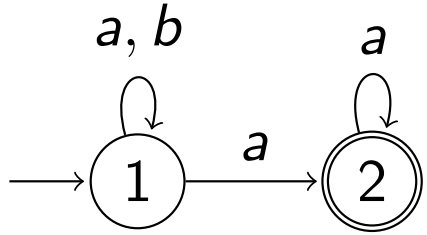
A few facts on GFG automata

Fact

Every deterministic automaton is **GFG**.

Fact

*There are some non-***GFG** *automata:*



$(a+b)^* a^\omega$

Adam plays $aaa\ldots$ until Eve goes to 2, then ba^ω.

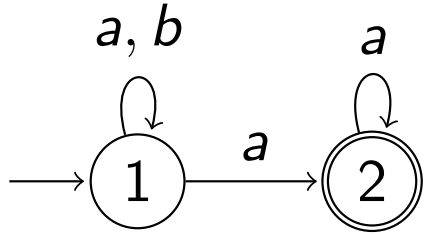
A few facts on GFG automata

Fact

Every deterministic automaton is **GFG**.

Fact

*There are some non-***GFG** *automata:*



$(a+b)^* a^\omega$

Adam plays $aaa\ldots$ until Eve goes to 2, then ba^ω.

Fact

GFG *automaton* $+ \sigma_{\mathrm{GFG}} =$ *Det automaton.*

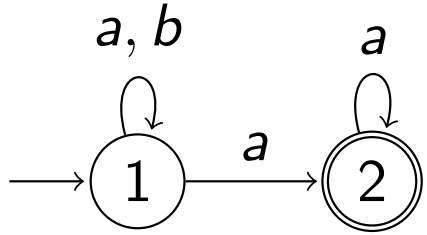
A few facts on GFG automata

Fact

Every deterministic automaton is **GFG**.

Fact

*There are some non-*__GFG__ *automata:*



$(a+b)^* a^\omega$

Adam plays $aaa\ldots$ until Eve goes to 2, then ba^ω.

Fact

GFG *automaton* $+ \sigma_{\mathrm{GFG}} =$ *Det automaton.*

Fact

If $\mathcal{A}$ *is* **GFG**, *no need to know its strategy* σ_{GFG} *to solve* $G \circ \mathcal{A}$ *!*

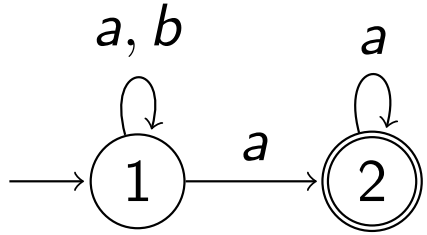
A few facts on GFG automata

Fact

Every deterministic automaton is **GFG**.

Fact

*There are some non-*GFG *automata:*



$(a+b)^* a^\omega$

Adam plays $aaa\ldots$ until Eve goes to 2, then ba^ω.

Fact

GFG *automaton* $+ \sigma_{\mathrm{GFG}} =$ *Det automaton.*

Fact

If $\mathcal{A}$ *is* **GFG**, *no need to know its strategy* σ_{GFG} *to solve* $G \circ \mathcal{A}$ *!*

$\rightarrow$ we can ignore the strategy in algorithms like synthesis.

GFG versus Deterministic

Lemma: For Reachability/Safety conditions, **GFG** $\approx$ Det.

GFG versus Deterministic

Lemma: For Reachability/Safety conditions, **GFG** $\approx$ Det.

Theorem (K., Skrzypczak 2015)

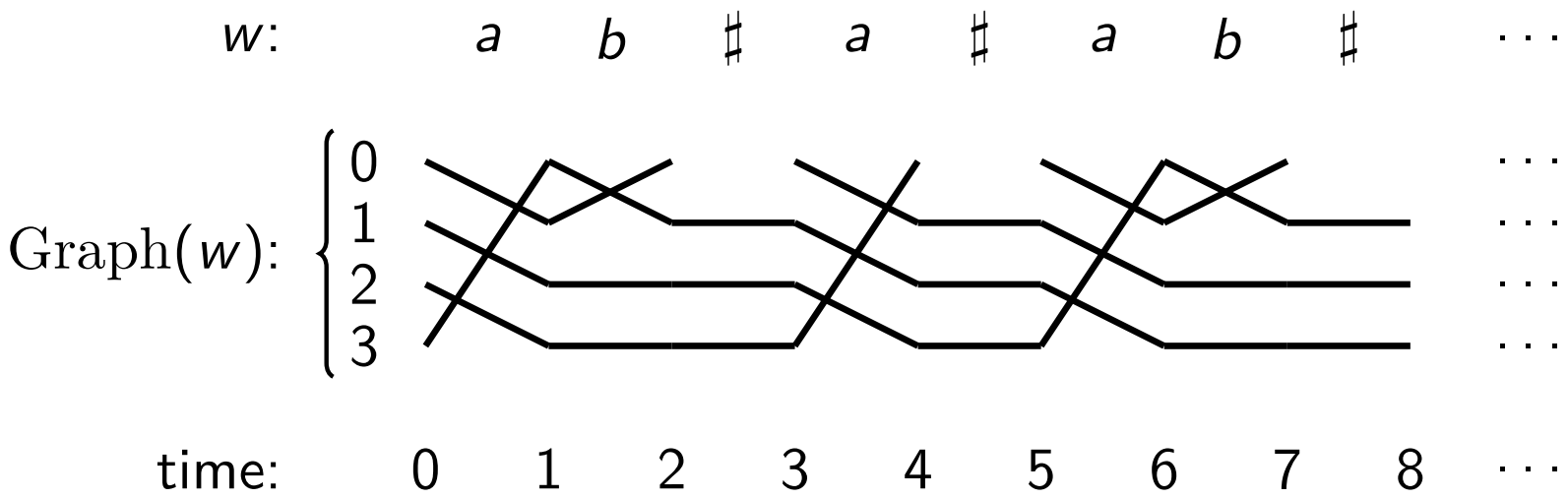
GFG *parity automata can be exponentially more succinct than deterministic ones.*

GFG versus Deterministic

Lemma: For Reachability/Safety conditions, **GFG** $\approx$ Det.

Theorem (K., Skrzypczak 2015)

GFG *parity automata can be exponentially more succinct than deterministic ones.*



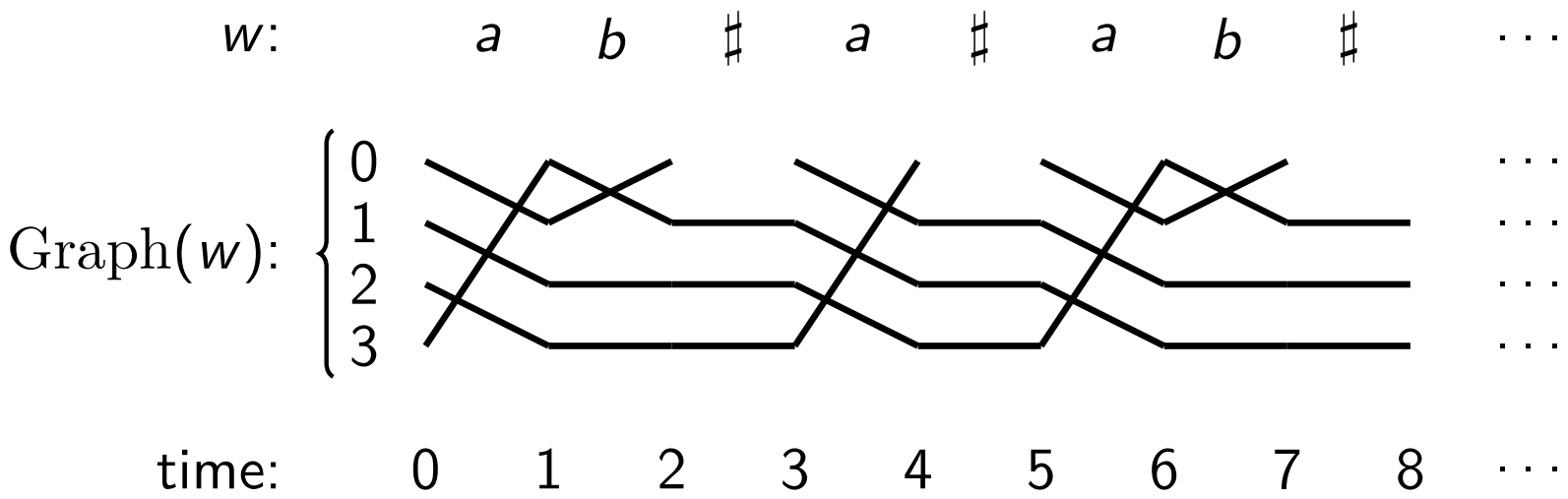
$L_n = \{w \mid \mathrm{Graph}(w) \text{ contains an } \infty \text{ path}\}$.

GFG versus Deterministic

Lemma: For Reachability/Safety conditions, **GFG** $\approx$ Det.

Theorem (K., Skrzypczak 2015)

GFG *parity automata can be exponentially more succinct than deterministic ones.*

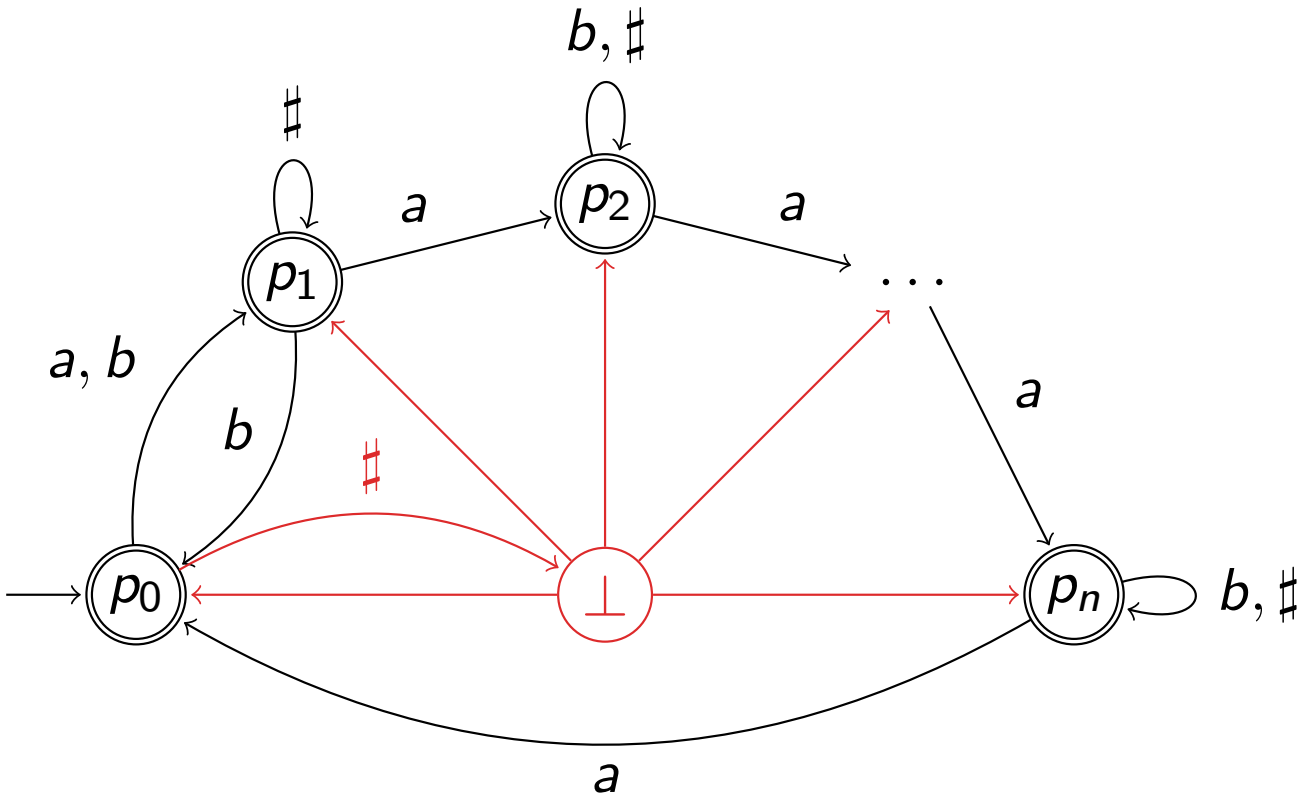


$$L_n = \{w \mid \mathrm{Graph}(w) \text{ contains an } \infty \text{ path}\}.$$

Any Det Parity automaton for L_n needs $\dfrac{2^{\frac{n}{2}}}{n+1}$ states !

A small GFG automaton for L_n

GFG coBüchi automaton:



Idea: Try paths one after the other.

Recognize GFG automata

GFGness problem: input $\mathcal{A}_{\mathrm{ND}}$, is it **GFG**?

Recognize GFG automata

GFGness problem: input $\mathcal{A}_{\mathrm{ND}}$, is it **GFG**?

Solve the GFG game ?

Recognize GFG automata

GFGness problem: input $\mathcal{A}_{\mathrm{ND}}$, is it **GFG**?

Solve the GFG game ?
Acceptance condition of the form "$u \in L \Rightarrow$run accepting"

Recognize GFG automata

GFGness problem: input $\mathcal{A}_{\mathrm{ND}}$, is it **GFG**?

Solve the GFG game ?
Acceptance condition of the form "$u \in L \Rightarrow$run accepting"
$\rightarrow$ We need a **GFG** automaton !
Upper bound: EXPTIME

Recognize GFG automata

GFGness problem: input $\mathcal{A}_{\mathrm{ND}}$, is it **GFG**?

Solve the GFG game ?
Acceptance condition of the form "$u \in L \Rightarrow$run accepting"
$\rightarrow$ We need a **GFG** automaton !
Upper bound: EXPTIME

Theorem (K., Skrypczak 2015)

The GFGness problem is in P for coBüchi automata.

Recognize GFG automata

GFGness problem: input $\mathcal{A}_{\mathrm{ND}}$, is it **GFG**?

Solve the GFG game ?
Acceptance condition of the form "$u \in L \Rightarrow$run accepting"
$\rightarrow$ We need a **GFG** automaton !
Upper bound: EXPTIME

Theorem (K., Skrypczak 2015)

The GFGness problem is in P for coBüchi automata.

Theorem (Bagnol, K. (unpublished))

The GFGness problem is in P for Büchi automata.

For Büchi, we use the parity game G_2 instead of GFG:

- Eve builds one run ρ and Adam two runs τ_1 and τ_2
- Eve wins if (τ_1 or τ_2 accepts)$\Rightarrow$ ρ accepts.

Recognize GFG automata

GFGness problem: input $\mathcal{A}_{\mathrm{ND}}$, is it **GFG**?

Solve the GFG game ?
Acceptance condition of the form "$u \in L \Rightarrow$run accepting"
$\rightarrow$ We need a **GFG** automaton !
Upper bound: EXPTIME

Theorem (K., Skrypczak 2015)

The GFGness problem is in P for coBüchi automata.

Theorem (Bagnol, K. (unpublished))

The GFGness problem is in P for Büchi automata.

For Büchi, we use the parity game G_2 instead of GFG:

- Eve builds one run ρ and Adam two runs τ_1 and τ_2
- Eve wins if (τ_1 or τ_2 accepts)$\Rightarrow$ ρ accepts.

Conjecture: G_2 is equivalent to GFG for all parity automata.

Building GFG automata

From a language L, how to obtain $\mathcal{A}_{\mathrm{GFG}}$ for L ?

Building GFG automata

From a language L, how to obtain $\mathcal{A}_{\mathrm{GFG}}$ for L ?

Incremental construction [K., Majumdar 2018]:
From $\mathcal{A}_{\mathrm{ND}}$: Increase the size of the automaton until it is **GFG**.
Worst case: reach the full determinization construction.

Building GFG automata

From a language L, how to obtain $\mathcal{A}_{\mathrm{GFG}}$ for L ?

Incremental construction [K., Majumdar 2018]:
From $\mathcal{A}_{\mathrm{ND}}$: Increase the size of the automaton until it is **GFG**.
Worst case: reach the full determinization construction.

For **efficiency**: solution to the GFGness problem is needed !
So for now only efficient for coBüchi (and Büchi)

Building GFG automata

From a language L, how to obtain $\mathcal{A}_{\mathrm{GFG}}$ for L ?

Incremental construction [K., Majumdar 2018]:
From $\mathcal{A}_{\mathrm{ND}}$: Increase the size of the automaton until it is **GFG**.
Worst case: reach the full determinization construction.

For **efficiency**: solution to the GFGness problem is needed !
So for now only efficient for coBüchi (and Büchi)

Directly from formula φ [Iosti, K. (unpublished)]
Tailored for **Synthesis Problem**
For now: Fragment of coBüchi languages

Conclusion

- **GFG** automata are intermediate between **Det** and **ND**.
- They are useful to solve complex games.
- They can bring an exponential advantage compared to determinism.
- For simple parity conditions, they are recognized (and built) efficiently.

Conclusion

- **GFG** automata are intermediate between **Det** and **ND**.
- They are useful to solve complex games.
- They can bring an exponential advantage compared to determinism.
- For simple parity conditions, they are recognized (and built) efficiently.

Open problems

- Complexity of **GFGness problem** (even for 3 parity ranks).
- New ways to build **GFG** automata.
- Generalization to other models (alternating, pushdown, . . .).