Introductory Lecture

Games Computer Scientists Play

Martin Zimmermann

July 19th, 2018

The Pumping Lemma

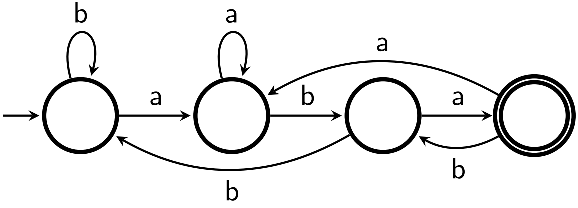
$L \subseteq \Sigma^*$ regular implies

$$\begin{aligned}\exists n \in \mathbb{N}\ \forall w \in L \cap \Sigma^{\geq n}\ \exists x, y, z \in \Sigma^*\ & xyz = w \land \\ & |xy| \leq n \land \\ & |y| > 0 \land \\ & \forall i \in \mathbb{N}\ xy^i z \in L\end{aligned}$$

The Pumping Lemma

$L \subseteq \Sigma^*$ regular implies

$$\exists n \in \mathbb{N}\ \forall w \in L \cap \Sigma^{\geq n}\ \exists x, y, z \in \Sigma^*\ \begin{aligned} & xyz = w \land \\ & |xy| \leq n \land \\ & |y| > 0 \land \\ & \forall i \in \mathbb{N}\ xy^i z \in L \end{aligned}$$



The Pumping Lemma

$$
\begin{aligned}
\forall n \in \mathbb{N}\ \exists w \in L \cap \Sigma^{\geq n}\ \forall x, y, z \in \Sigma^*\ & \\
& (xyz = w \wedge |xy| \leq n \wedge |y| > 0) \rightarrow \\
& \quad \exists i \in \mathbb{N}\ xy^i z \notin L
\end{aligned}
$$

implies that L is not regular.

The Pumping Lemma

$$\forall n \in \mathbb{N}\ \exists w \in L \cap \Sigma^{\geq n}\ \forall x, y, z \in \Sigma^* \\ (xyz = w \land |xy| \leq n \land |y| > 0) \rightarrow \\ \exists i \in \mathbb{N}\ xy^i z \notin L$$

implies that L is not regular.

Example

$L = \{a^n b^n \mid n > 0\} = \{ab, aabb, aaabbb, aaaabbbb, \ldots\}$

The Pumping Lemma

$$\forall n \in \mathbb{N}\ \exists w \in L \cap \Sigma^{\geq n}\ \forall x, y, z \in \Sigma^*$$
$$(xyz = w \wedge |xy| \leq n \wedge |y| > 0) \rightarrow$$
$$\exists i \in \mathbb{N}\ xy^i z \notin L$$

implies that L is not regular.

Example

$L = \{a^n b^n \mid n > 0\} = \{ab, aabb, aaabbb, aaaabbbb, \ldots\}$

n

The Pumping Lemma

$$\forall n \in \mathbb{N}\ \exists w \in L \cap \Sigma^{\geq n}\ \forall x, y, z \in \Sigma^*$$
$$(xyz = w \wedge |xy| \leq n \wedge |y| > 0) \rightarrow$$
$$\exists i \in \mathbb{N}\ xy^i z \notin L$$

implies that L is not regular.

Example

$L = \{a^n b^n \mid n > 0\} = \{ab, aabb, aaabbb, aaaabbbb, \ldots\}$

$$n \longrightarrow\ w = \overbrace{aaaaa \cdots aaaaa}^{n}\ \overbrace{bbbbb \cdots bbbbb}^{n}$$

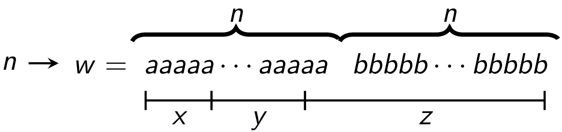
The Pumping Lemma

$$\forall n \in \mathbb{N}\ \exists w \in L \cap \Sigma^{\geq n}\ \forall x, y, z \in \Sigma^*$$
$$(xyz = w \wedge |xy| \leq n \wedge |y| > 0) \rightarrow$$
$$\exists i \in \mathbb{N}\ xy^i z \notin L$$

implies that L is not regular.

Example

$L = \{a^n b^n \mid n > 0\} = \{ab, aabb, aaabbb, aaaabbbb, \ldots\}$



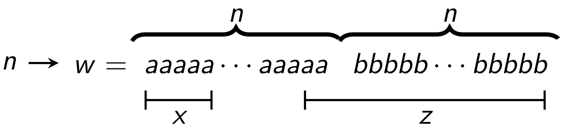
The Pumping Lemma

$$\forall n \in \mathbb{N}\ \exists w \in L \cap \Sigma^{\geq n}\ \forall x, y, z \in \Sigma^*$$
$$(xyz = w \wedge |xy| \leq n \wedge |y| > 0) \rightarrow$$
$$\exists i \in \mathbb{N}\ xy^iz \notin L$$

implies that L is not regular.

Example

$L = \{a^n b^n \mid n > 0\} = \{ab, aabb, aaabbb, aaaabbbb, \ldots\}$



Model Checking

Intuition

Quantifiers and logical connectives correspond to moves in a game between a player trying to satisfy a formula and an opponent trying to falsify it.

Model Checking

Intuition

Quantifiers and logical connectives correspond to moves in a game between a player trying to satisfy a formula and an opponent trying to falsify it.

Model Checking

Given a structure $\mathfrak{A}$ and a sentence φ of first-order logic, decide whether $\mathfrak{A}$ satisfies φ.

Model Checking

Intuition

Quantifiers and logical connectives correspond to moves in a game between a player trying to satisfy a formula and an opponent trying to falsify it.

Model Checking

Given a structure $\mathfrak{A}$ and a sentence φ of first-order logic, decide whether $\mathfrak{A}$ satisfies φ.

Example

$\mathfrak{A} = (\mathbb{N}, <, |, 1)$ and
$\varphi = \forall x \exists y (x < y \wedge \forall z (\neg (z \,|\, y) \vee z = 1) \vee z = y)$

Model Checking Games

A game between Verifier and Falsifier.

- Positions: (ψ, β) where ψ is a subformula of φ and β is a partial variable valuation.
- Moves for Verifier:

$$(\exists x \psi, \beta) \longrightarrow (\psi, \beta[x \mapsto a]) \text{ for all } a \text{ of } \mathfrak{A}$$

$$(\psi_0 \vee \psi_1, \beta) \longrightarrow (\psi_0, \beta)$$

$$(\psi_0 \vee \psi_1, \beta) \longrightarrow (\psi_1, \beta)$$

Model Checking Games

A game between Verifier and Falsifier.

- Positions: (ψ, β) where ψ is a subformula of φ and β is a partial variable valuation.
- Moves for Verifier:

$$(\exists x \psi, \beta) \longrightarrow (\psi, \beta[x \mapsto a]) \text{ for all } a \text{ of } \mathfrak{A}$$

$$(\psi_0 \vee \psi_1, \beta) \longrightarrow (\psi_0, \beta)$$

$$(\psi_0 \vee \psi_1, \beta) \longrightarrow (\psi_1, \beta)$$

- Moves for Falsifier: dual

Model Checking Games

A game between Verifier and Falsifier.

- Positions: (ψ, β) where ψ is a subformula of φ and β is a partial variable valuation.
- Moves for Verifier:

$$(\exists x \psi, \beta) \longrightarrow (\psi, \beta[x \mapsto a]) \text{ for all } a \text{ of } \mathfrak{A}$$

$$(\psi_0 \vee \psi_1, \beta) \longrightarrow (\psi_0, \beta)$$

$$(\psi_0 \vee \psi_1, \beta) \longrightarrow (\psi_1, \beta)$$

- Moves for Falsifier: dual
- Terminal positions: $(R(x_1, \dots, x_n), \beta)$ for relation symbol R. Winning for Verifier if and only if $(\beta(x_1) \dots, \beta(x_n)) \in R^{\mathfrak{A}}$.

Model Checking Games

A game between Verifier and Falsifier.

- Positions: (ψ, β) where ψ is a subformula of φ and β is a partial variable valuation.
- Moves for Verifier:

$$(\exists x \psi, \beta) \longrightarrow (\psi, \beta[x \mapsto a]) \text{ for all } a \text{ of } \mathfrak{A}$$

$$(\psi_0 \vee \psi_1, \beta) \longrightarrow (\psi_0, \beta) \qquad (\psi_0 \vee \psi_1, \beta) \longrightarrow (\psi_1, \beta)$$

- Moves for Falsifier: dual
- Terminal positions: $\neg(R(x_1, \ldots, x_n), \beta)$ for relation symbol R. Winning for Verifier if and only if $(\beta(x_1) \ldots, \beta(x_n)) \notin R^{\mathfrak{A}}$.

Example Continued

$\mathfrak{A} = (\mathbb{N}, <, |, 1)$ and

$\varphi = \forall x \exists y \underbrace{\left(x < y \land \forall z(\neg(z \mid y) \lor z = 1 \lor z = y)\right)}_{\psi}$

Example Continued

$\mathfrak{A} = (\mathbb{N}, <, |, 1)$ and

$$\varphi = \forall x \exists y \underbrace{\left(x < y \land \forall z(\neg(z \mid y) \lor z = 1 \lor z = y)\right)}_{\psi}$$

$$(\forall x \exists y \psi, \emptyset)$$

Example Continued

$\mathfrak{A} = (\mathbb{N}, <, |, 1)$ and

$$\varphi = \forall x \exists y \underbrace{(x < y \land \forall z(\neg(z \mid y) \lor z = 1 \lor z = y))}_{\psi}$$

$$\text{F} \left(\begin{array}{l} (\forall x \exists y \psi, \emptyset) \\ \downarrow \\ (\exists y \psi, x \mapsto 3) \end{array} \right.$$

Example Continued

$\mathfrak{A} = (\mathbb{N}, <, |, 1)$ and

$$\varphi = \forall x \exists y \underbrace{\left(x < y \land \forall z(\neg(z \,|\, y) \lor z = 1 \lor z = y)\right)}_{\psi}$$

$$(\forall x \exists y \psi, \emptyset)$$

F ↓

$$(\exists y \psi, x \mapsto 3)$$

V ↓

$$(x < y \land \forall z(\neg(z \,|\, y) \lor z = 1 \lor z = y), x \mapsto 3, y \mapsto 7)$$

Example Continued

$\mathfrak{A} = (\mathbb{N}, <, |, 1)$ and

$$\varphi = \forall x \exists y \underbrace{(x < y \land \forall z(\neg(z \,|\, y) \lor z = 1 \lor z = y))}_{\psi}$$

$$\begin{array}{l} (\forall x \exists y \psi, \emptyset) \\ \quad \downarrow \text{F} \\ (\exists y \psi, x \mapsto 3) \\ \quad \downarrow \text{V} \\ (x < y \land \forall z(\neg(z \,|\, y) \lor z = 1 \lor z = y), x \mapsto 3, y \mapsto 7) \\ \quad \downarrow \text{F} \\ (\forall z(\neg(z \,|\, y) \lor z = 1 \lor z = y), x \mapsto 3, y \mapsto 7) \end{array}$$

Example Continued

$\mathfrak{A} = (\mathbb{N}, <, |, 1)$ and

$$\varphi = \forall x \exists y \underbrace{\left(x < y \land \forall z(\neg(z \,|\, y) \lor z = 1 \lor z = y)\right)}_{\psi}$$

$$\begin{array}{ll}
 & (\forall x \exists y \psi, \emptyset) \\
\text{F} \downarrow & \\
 & (\exists y \psi, x \mapsto 3) \\
\text{V} \downarrow & \\
 & (x < y \land \forall z(\neg(z \,|\, y) \lor z = 1 \lor z = y), x \mapsto 3, y \mapsto 7) \\
\text{F} \downarrow & \\
 & (\forall z(\neg(z \,|\, y) \lor z = 1 \lor z = y), x \mapsto 3, y \mapsto 7) \\
\text{F} \downarrow & \\
 & (\neg(z \,|\, y) \lor z = 1 \lor z = y, x \mapsto 3, y \mapsto 7, z \mapsto 13)
\end{array}$$

Example Continued

$\mathfrak{A} = (\mathbb{N}, <, |, 1)$ and

$$\varphi = \forall x \exists y \underbrace{(x < y \wedge \forall z(\neg(z \,|\, y) \vee z = 1 \vee z = y))}_{\psi}$$

$$\begin{array}{rl}
 & (\forall x \exists y \psi, \emptyset) \\
\text{F} \downarrow & \\
 & (\exists y \psi, x \mapsto 3) \\
\text{V} \downarrow & \\
 & (x < y \wedge \forall z(\neg(z \,|\, y) \vee z = 1 \vee z = y), x \mapsto 3, y \mapsto 7) \\
\text{F} \downarrow & \\
 & (\forall z(\neg(z \,|\, y) \vee z = 1 \vee z = y), x \mapsto 3, y \mapsto 7) \\
\text{F} \downarrow & \\
 & (\neg(z \,|\, y) \vee z = 1 \vee z = y, x \mapsto 3, y \mapsto 7, z \mapsto 13) \\
\text{V} \downarrow & \\
 & (\neg(z \,|\, y), x \mapsto 3, y \mapsto 7, z \mapsto 13)
\end{array}$$

Example Continued

$\mathfrak{A} = (\mathbb{N}, <, |, 1)$ and

$$\varphi = \forall x \exists y \underbrace{(x < y \wedge \forall z(\neg(z \,|\, y) \vee z = 1 \vee z = y))}_{\psi}$$

$$\begin{aligned}
& (\forall x \exists y \psi, \emptyset) \\
\text{F} \quad & \downarrow \\
& (\exists y \psi, x \mapsto 3) \\
\text{V} \quad & \downarrow \\
& (x < y \wedge \forall z(\neg(z \,|\, y) \vee z = 1 \vee z = y), x \mapsto 3, y \mapsto 7) \\
\text{F} \quad & \downarrow \\
& (\forall z(\neg(z \,|\, y) \vee z = 1 \vee z = y), x \mapsto 3, y \mapsto 7) \\
\text{F} \quad & \downarrow \\
& (\neg(z \,|\, y) \vee z = 1 \vee z = y, x \mapsto 3, y \mapsto 7, z \mapsto 13) \\
\text{V} \quad & \downarrow \\
& (\neg(z \,|\, y), x \mapsto 3, y \mapsto 7, z \mapsto 13)
\end{aligned}$$

Winning for Verifier, as 13 does not divide 7

Example Continued

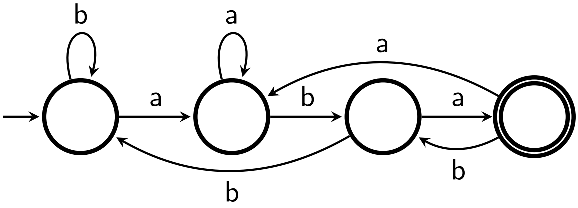
$\mathfrak{A} = (\mathbb{N}, <, |, 1)$ and
$$\varphi = \forall x \exists y \underbrace{\left(x < y \wedge \forall z(\neg(z \,|\, y) \vee z = 1 \vee z = y)\right)}_{\psi}$$

Theorem

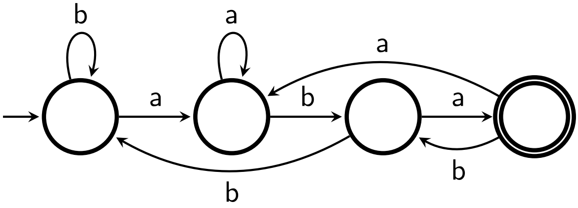
The following are equivalent:

1. *$\mathfrak{A}$ satisfies φ.*
2. *Verifier has a winning strategy for the game induced by $\mathfrak{A}$ and φ.*

Word Automata Emptiness

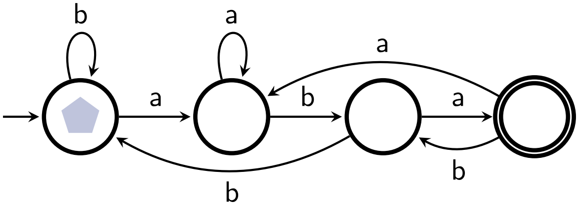


Word Automata Emptiness



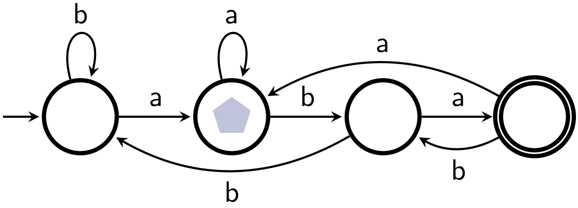
For automata on finite words, emptiness can be expressed as a (trivial) one-player reachability game: find a path from the initial state to some accepting state.

Word Automata Emptiness



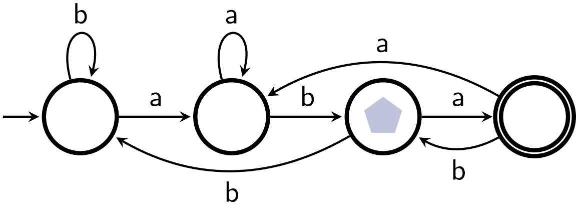
For automata on finite words, emptiness can be expressed as a (trivial) one-player reachability game: find a path from the initial state to some accepting state.

Word Automata Emptiness



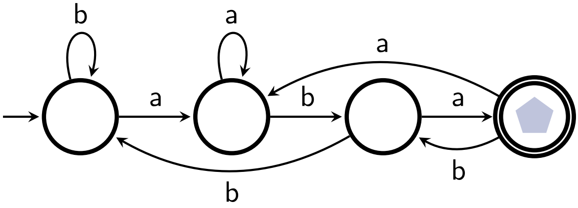
For automata on finite words, emptiness can be expressed as a (trivial) one-player reachability game: find a path from the initial state to some accepting state.

Word Automata Emptiness



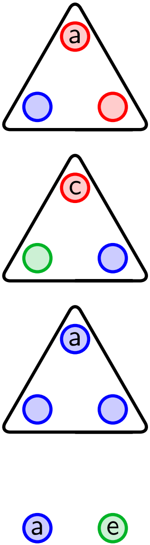
For automata on finite words, emptiness can be expressed as a (trivial) one-player reachability game: find a path from the initial state to some accepting state.

Word Automata Emptiness

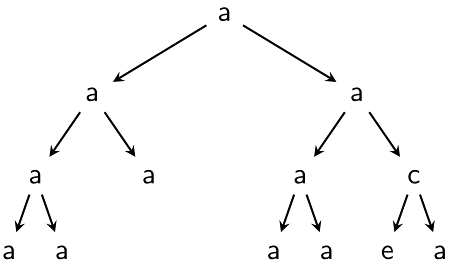


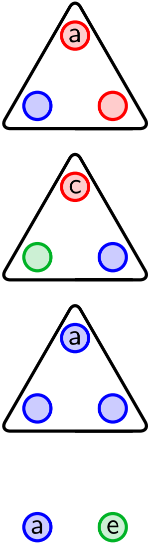
For automata on finite words, emptiness can be expressed as a (trivial) one-player reachability game: find a path from the initial state to some accepting state.

Tree Automata Emptiness

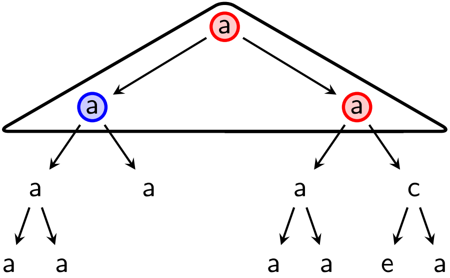


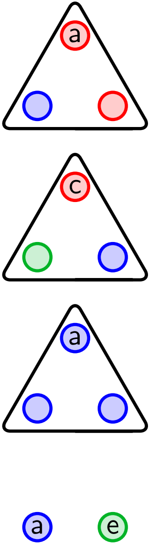
Tree Automata Emptiness



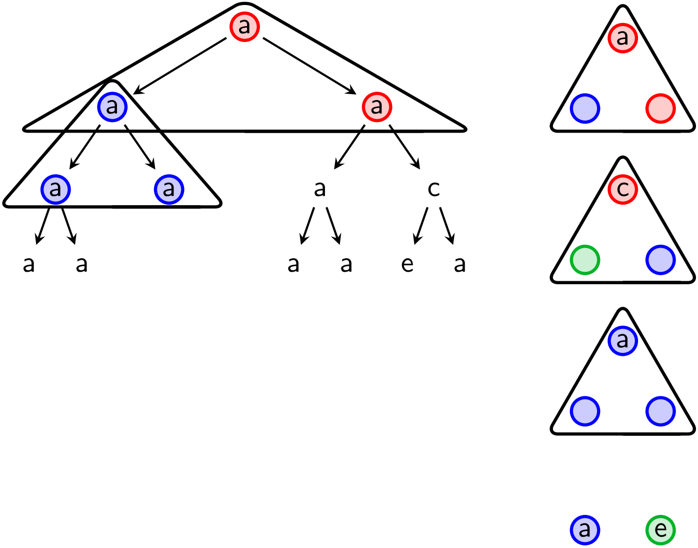


Tree Automata Emptiness

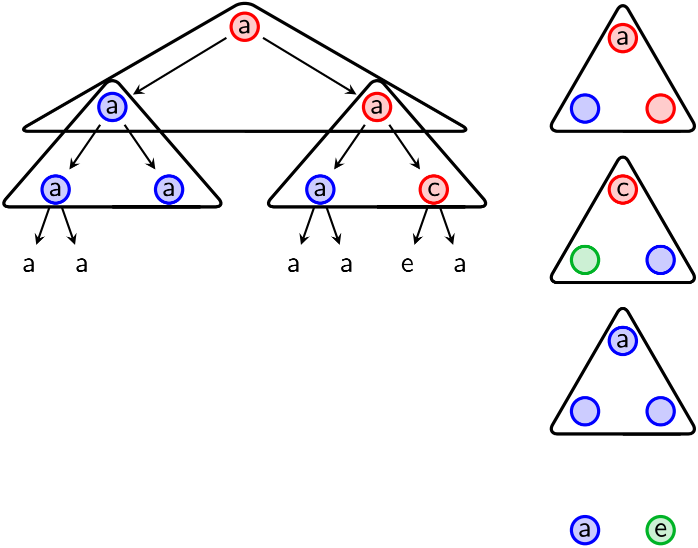




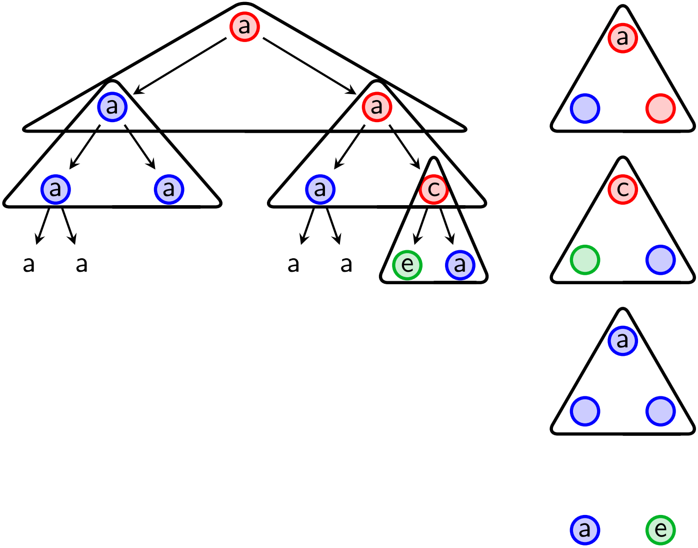
Tree Automata Emptiness



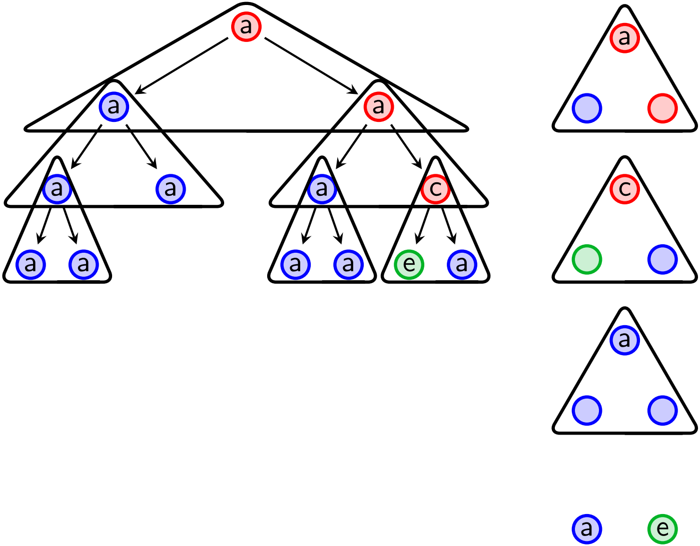
Tree Automata Emptiness



Tree Automata Emptiness

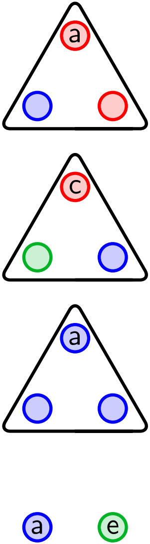


Tree Automata Emptiness



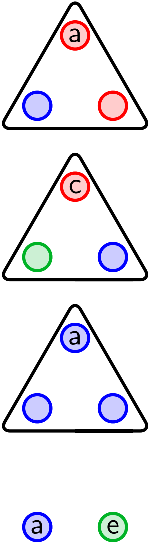
The Emptiness Game

One player picks transitions, the other (implicitly) the structure of the input tree.



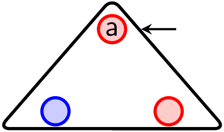
The Emptiness Game

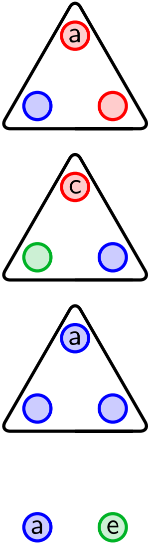
One player picks transitions, the other (implicitly) the structure of the input tree.



The Emptiness Game

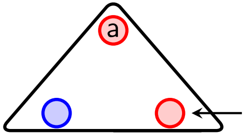
One player picks transitions, the other (implicitly) the structure of the input tree.

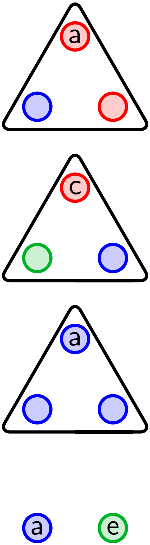




The Emptiness Game

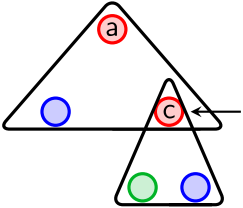
One player picks transitions, the other (implicitly) the structure of the input tree.

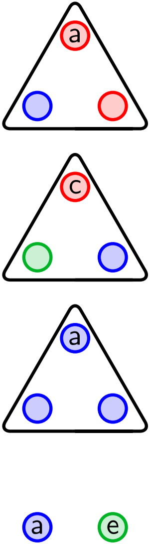




The Emptiness Game

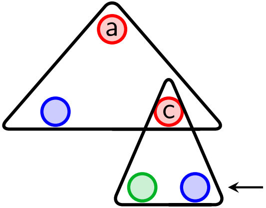
One player picks transitions, the other (implicitly) the structure of the input tree.

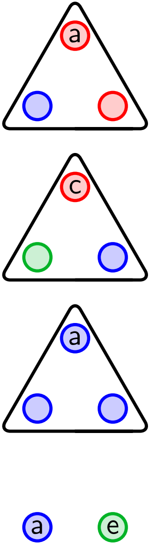




The Emptiness Game

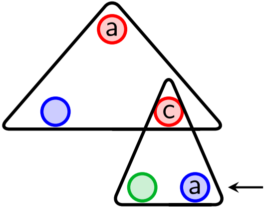
One player picks transitions, the other (implicitly) the structure of the input tree.

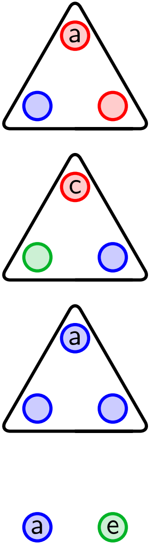




The Emptiness Game

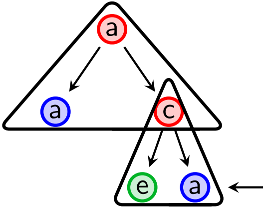
One player picks transitions, the other (implicitly) the structure of the input tree.

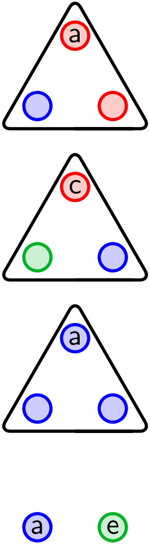




The Emptiness Game

One player picks transitions, the other (implicitly) the structure of the input tree.





The Emptiness Game

One player picks transitions, the other (implicitly) the structure of the input tree.

Theorem

An automaton has a non-empty language if and only if the player constructing a run has a winning strategy for the induced game.

The Emptiness Game

One player picks transitions, the other (implicitly) the structure of the input tree.

Theorem

An automaton has a non-empty language if and only if the player constructing a run has a winning strategy for the induced game.

An analogous result holds for automata on infinite trees. However, the resulting game is an infinite-duration game.

Determinacy

- All games considered thus far, at most one player can have a winning strategy.

Determinacy

- All games considered thus far, at most one player can have a winning strategy.
- A game is determined, if one of the players has a winning strategy for it.

Determinacy

- All games considered thus far, at most one player can have a winning strategy.
- A game is determined, if one of the players has a winning strategy for it.

Theorem (Zermelo 1913)

Every finite-duration two-player zero-sum game of perfect information is determined.

Determinacy

- All games considered thus far, at most one player can have a winning strategy.
- A game is determined, if one of the players has a winning strategy for it.

Theorem (Zermelo 1913)

Every finite-duration two-player zero-sum game of perfect information is determined.

The proof works by bottom-up induction over the finite tree of positions.

Determinacy

- All games considered thus far, at most one player can have a winning strategy.
- A game is determined, if one of the players has a winning strategy for it.

Theorem (Zermelo 1913)

Every finite-duration two-player zero-sum game of perfect information is determined.

Question

Is every infinite-duration two-player zero-sum game of perfect information determined?

Chomp

- There is a (rectangular) chocolate bar with $m \times n$ pieces.
- A move consists of taking a piece and all others that are to the right and above.
- Two players, Player 0 and Player 1, move in alternation, starting with Player 0.
- The player who takes the bottom-left piece loses.

Let's Play

Let's Play

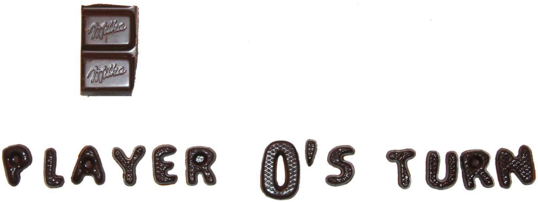
Let's Play

Let's Play

Let's Play

Let's Play

Let’s Play



Let's Play

Let's Play

Strategy Stealing

Claim

Player 0 has a winning strategy for every bar (unless $m = n = 1$).

Strategy Stealing

Claim

Player 0 has a winning strategy for every bar (unless $m = n = 1$).

- Assume Player 1 has a winning strategy.
- Look how this strategy reacts to Player 0 only taking the top-right piece in the first move.
- Let Player 0 use this strategy from the beginning.
- This is winning for Player 0, which is a contradiction.
- As Chomp is determined, this means Player 0 must have a winning strategy.

Strategy Stealing

Claim

Player 0 has a winning strategy for every bar (unless $m = n = 1$).

- Assume Player 1 has a winning strategy.
- Look how this strategy reacts to Player 0 only taking the top-right piece in the first move.
- Let Player 0 use this strategy from the beginning.
- This is winning for Player 0, which is a contradiction.
- As Chomp is determined, this means Player 0 must have a winning strategy.

Note

- The proof is non-constructive..
- ..winning strategy only known for special cases $n \times n$, $n \times 2$, $2 \times n$, $n \times 1$, and $1 \times n$ (try to find them).

Hamming Distance

In the following: $\mathbb{B} = \{0, 1\}$

Definition

For $x = x_0 x_1 x_2 \cdots$ and $y = y_0 y_1 y_2 \cdots$ in $\mathbb{B}^\omega$, the *Hamming distance* between x and y is defined as

$$\mathrm{hd}(x, y) = |\{n \in \mathbb{N} \mid x_n \neq y_n\}| \in \mathbb{N} \cup \{\infty\}.$$

Example

- $\mathrm{hd}(0101101000 \cdots, 1010100000 \cdots) = 5$
- $\mathrm{hd}(1010101010 \cdots, 0101010101 \cdots) = \infty$
- $\mathrm{hd}(1010101010 \cdots, 1111111111 \cdots) = \infty.$

Infinite XOR Functions

Definition

A function $f : \mathbb{B}^\omega \to \mathbb{B}$ is an *infinite XOR function*, if $hd(x, y) = 1$ implies $f(x) \neq f(y)$ for all $x, y \in \mathbb{B}^\omega$.

Infinite XOR Functions

Definition

A function $f : \mathbb{B}^\omega \to \mathbb{B}$ is an *infinite XOR function*, if $hd(x, y) = 1$ implies $f(x) \neq f(y)$ for all $x, y \in \mathbb{B}^\omega$.

Example

Infinite XOR Functions

Definition

A function $f : \mathbb{B}^\omega \to \mathbb{B}$ is an *infinite XOR function*, if $hd(x, y) = 1$ implies $f(x) \neq f(y)$ for all $x, y \in \mathbb{B}^\omega$.

Example

I have none.. we will come back to this later.

Infinite XOR Functions

Definition

A function $f : \mathbb{B}^\omega \to \mathbb{B}$ is an *infinite XOR function*, if $hd(x, y) = 1$ implies $f(x) \neq f(y)$ for all $x, y \in \mathbb{B}^\omega$.

Example

I have none.. we will come back to this later.

Theorem

There exists an infinite XOR function.

Infinite XOR Functions

Definition

A function $f : \mathbb{B}^\omega \to \mathbb{B}$ is an *infinite XOR function*, if $hd(x, y) = 1$ implies $f(x) \neq f(y)$ for all $x, y \in \mathbb{B}^\omega$.

Example

I have none.. we will come back to this later.

Theorem

There exists an infinite XOR function.

The proof requires the axion of choice.

The Game $\mathcal{G}_f$

- Fix some infinite XOR function f.
- We define a game $\mathcal{G}_f$ between Player 0 and Player 1 who pick sequences of bits in alternation.

The Game $\mathcal{G}_f$

- Fix some infinite XOR function f.
- We define a game $\mathcal{G}_f$ between Player 0 and Player 1 who pick sequences of bits in alternation.

Example

The Game $\mathcal{G}_f$

- Fix some infinite XOR function f.
- We define a game $\mathcal{G}_f$ between Player 0 and Player 1 who pick sequences of bits in alternation.

Example

1100

The Game $\mathcal{G}_f$

- Fix some infinite XOR function f.
- We define a game $\mathcal{G}_f$ between Player 0 and Player 1 who pick sequences of bits in alternation.

Example

1100 0

The Game $\mathcal{G}_f$

- Fix some infinite XOR function f.
- We define a game $\mathcal{G}_f$ between Player 0 and Player 1 who pick sequences of bits in alternation.

Example

1100 0 000000110000

The Game $\mathcal{G}_f$

- Fix some infinite XOR function f.
- We define a game $\mathcal{G}_f$ between Player 0 and Player 1 who pick sequences of bits in alternation.

Example

1100 0 000000110000 1100101

The Game $\mathcal{G}_f$

- Fix some infinite XOR function f.
- We define a game $\mathcal{G}_f$ between Player 0 and Player 1 who pick sequences of bits in alternation.

Example

1100 0 000000110000 1100101 1

The Game $\mathcal{G}_f$

- Fix some infinite XOR function f.
- We define a game $\mathcal{G}_f$ between Player 0 and Player 1 who pick sequences of bits in alternation.

Example

1100 0 000000110000 1100101 1 100000

The Game $\mathcal{G}_f$

- Fix some infinite XOR function f.
- We define a game $\mathcal{G}_f$ between Player 0 and Player 1 who pick sequences of bits in alternation.

Example

$$1100\ 0\ 000000110000\ 1100101\ 1\ 100000\ \cdots$$

The Game $\mathcal{G}_f$

- Fix some infinite XOR function f.
- We define a game $\mathcal{G}_f$ between Player 0 and Player 1 who pick sequences of bits in alternation.

Example

winner: Player $f($ 1100 0 000000110000 1100101 1 100000 $\cdots$ $)$

The Game $\mathcal{G}_f$

- Fix some infinite XOR function f.
- We define a game $\mathcal{G}_f$ between Player 0 and Player 1 who pick sequences of bits in alternation.

Example

winner: Player $f($ 1100 0 000000110000 1100101 1 100000 $\cdots$ $)$

- Formally, $\mathcal{G}_f$ is played in rounds $n = 0, 1, 2, \ldots$.
- In round n, first Player 0 picks $w_{2n} \in \mathbb{B}^+$, then Player 1 picks $w_{2n+1} \in \mathbb{B}^+$.
- Play $w_0, w_1, w_2, \ldots$ is won by Player $f(w_0 w_1 w_2 \cdots)$.

There are Undetermined Games

Theorem

Let f be an infinite XOR function. No player has a winning strategy for $\mathcal{G}_f$.

Proof Idea

Strategy stealing:

- For every strategy τ of Player 1, we construct two counter strategies σ and σ' that mimic τ.
- The only difference between σ and σ' is that one starts by playing a 0, the other by playing a 1.
- The remainder of the plays resulting from playing σ and σ' against τ are equal.
- Hence, their Hamming distance is 1 and one of the plays is won by Player 0.
- Thus, τ is not a winning strategy.

Proof Idea

Strategy stealing:

- For every strategy τ of Player 1, we construct two counter strategies σ and σ' that mimic τ.
- The only difference between σ and σ' is that one starts by playing a 0, the other by playing a 1.
- The remainder of the plays resulting from playing σ and σ' against τ are equal.
- Hence, their Hamming distance is 1 and one of the plays is won by Player 0.
- Thus, τ is not a winning strategy.

The argument showing that Player 0 has no winning strategy is similar.

Proof

Let τ be a strategy for Player 1 in $\mathcal{G}_f$. We show that τ is not winning by constructing counter strategies σ and σ' as above.

Proof

Let τ be a strategy for Player 1 in $\mathcal{G}_f$. We show that τ is not winning by constructing counter strategies σ and σ' as above.

Proof

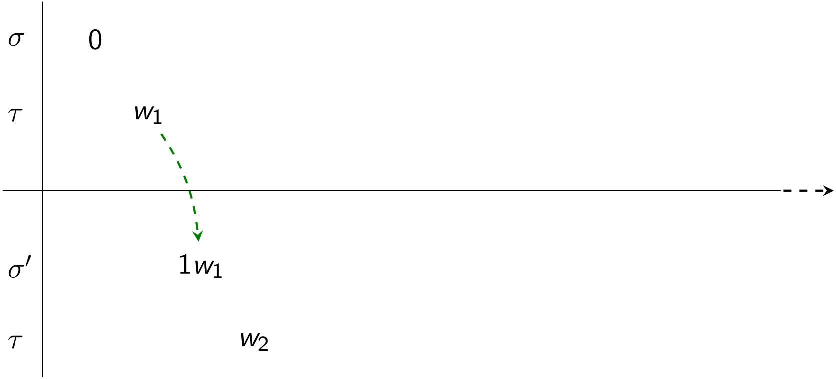
Let τ be a strategy for Player 1 in $\mathcal{G}_f$. We show that τ is not winning by constructing counter strategies σ and σ' as above.

Proof

Let τ be a strategy for Player 1 in $\mathcal{G}_f$. We show that τ is not winning by constructing counter strategies σ and σ' as above.

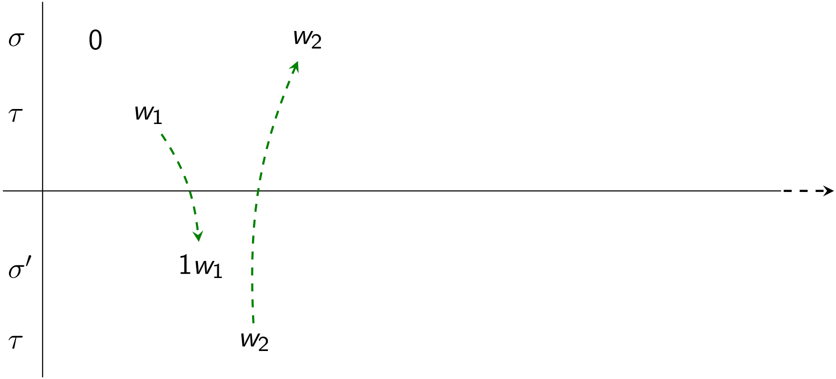
Proof

Let τ be a strategy for Player 1 in $\mathcal{G}_f$. We show that τ is not winning by constructing counter strategies σ and σ' as above.



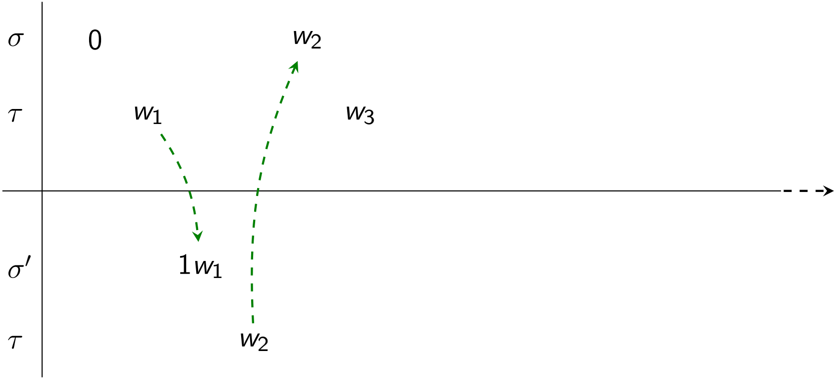
Proof

Let τ be a strategy for Player 1 in $\mathcal{G}_f$. We show that τ is not winning by constructing counter strategies σ and σ' as above.



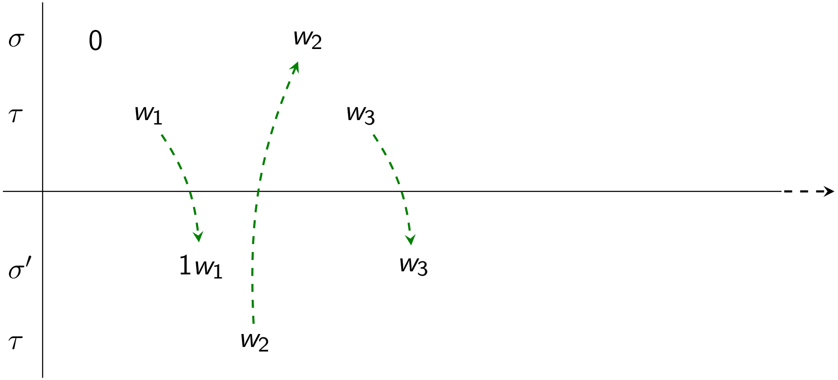
Proof

Let τ be a strategy for Player 1 in $\mathcal{G}_f$. We show that τ is not winning by constructing counter strategies σ and σ' as above.



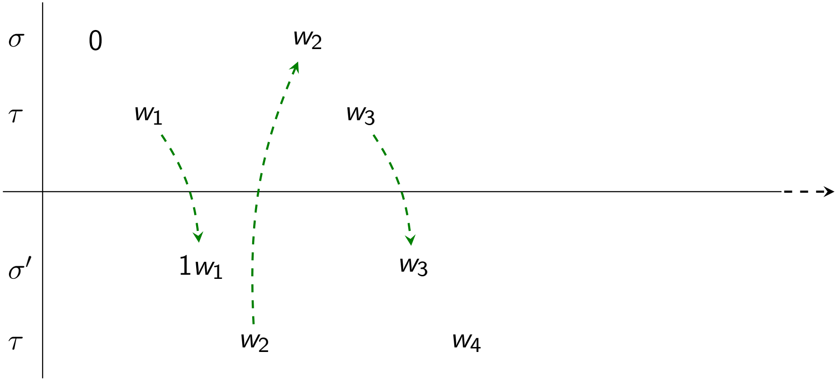
Proof

Let τ be a strategy for Player 1 in $\mathcal{G}_f$. We show that τ is not winning by constructing counter strategies σ and σ' as above.



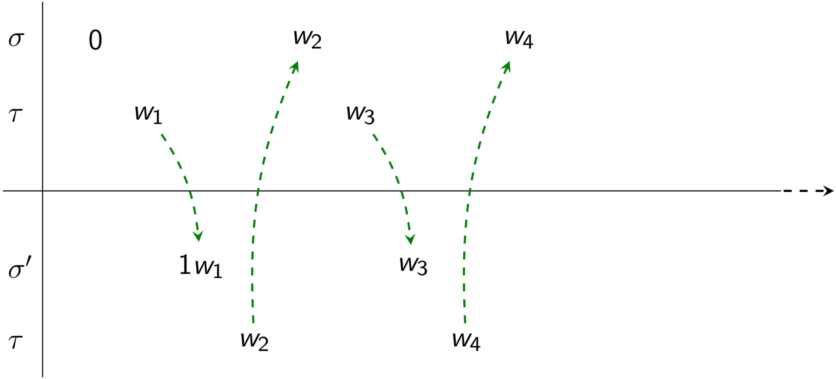
Proof

Let τ be a strategy for Player 1 in $\mathcal{G}_f$. We show that τ is not winning by constructing counter strategies σ and σ' as above.



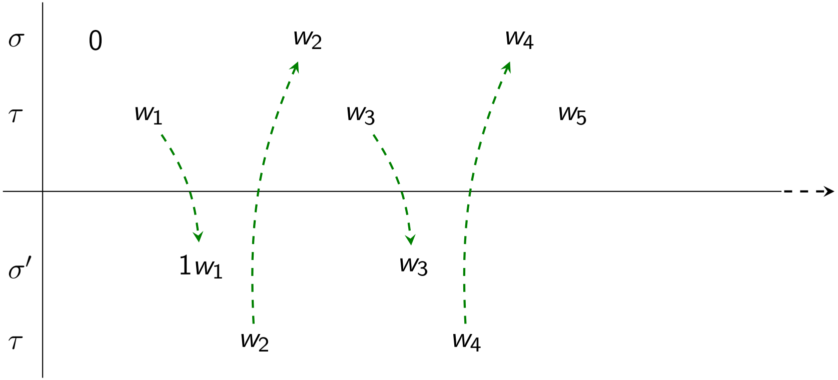
Proof

Let τ be a strategy for Player 1 in $\mathcal{G}_f$. We show that τ is not winning by constructing counter strategies σ and σ' as above.



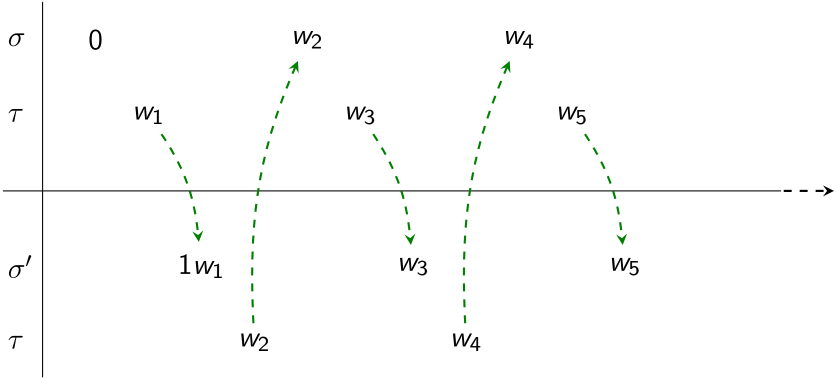
Proof

Let τ be a strategy for Player 1 in $\mathcal{G}_f$. We show that τ is not winning by constructing counter strategies σ and σ' as above.



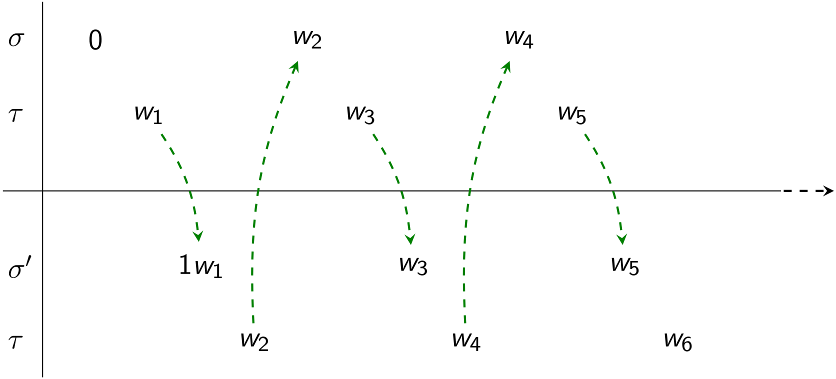
Proof

Let τ be a strategy for Player 1 in $\mathcal{G}_f$. We show that τ is not winning by constructing counter strategies σ and σ' as above.



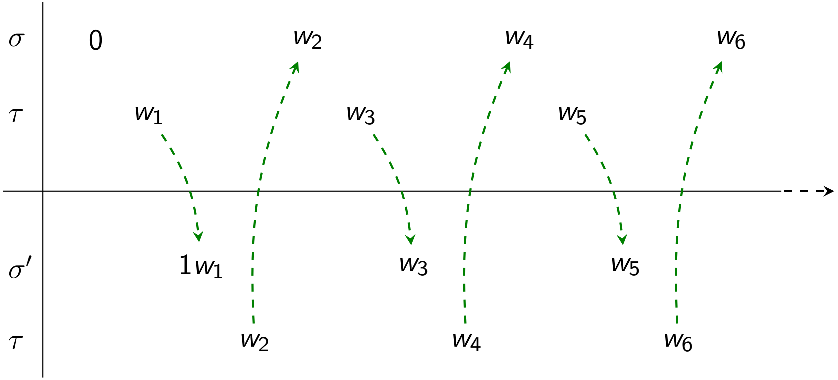
Proof

Let τ be a strategy for Player 1 in $\mathcal{G}_f$. We show that τ is not winning by constructing counter strategies σ and σ' as above.



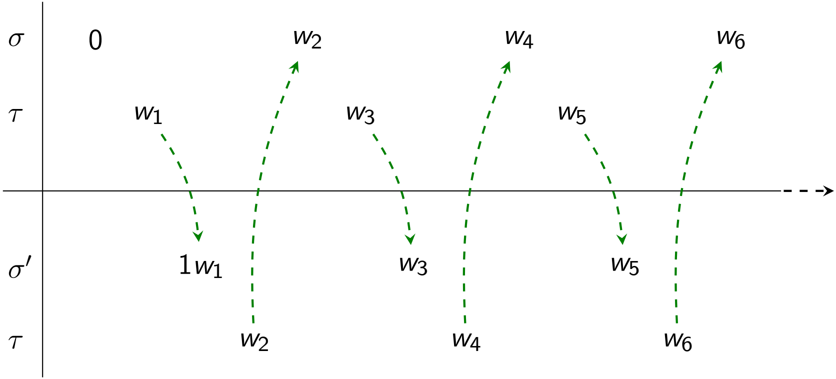
Proof

Let τ be a strategy for Player 1 in $\mathcal{G}_f$. We show that τ is not winning by constructing counter strategies σ and σ' as above.



Proof

Let τ be a strategy for Player 1 in $\mathcal{G}_f$. We show that τ is not winning by constructing counter strategies σ and σ' as above.



Consider the resulting plays: they differ only at their first position. Hence, Player 0 wins one of them. Thus, τ is not winning.

Proof

Let σ be a strategy for Player 0 in $\mathcal{G}_f$. We show that σ is not winning by constructing counter strategies τ and τ' as above.

Proof

Let σ be a strategy for Player 0 in $\mathcal{G}_f$. We show that σ is not winning by constructing counter strategies τ and τ' as above.

Proof

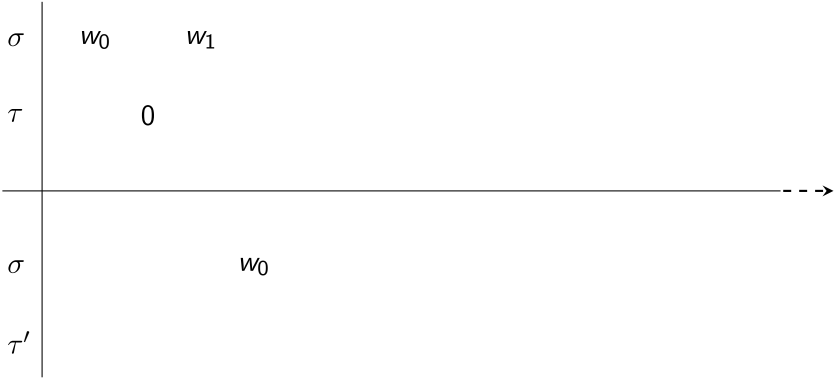
Let σ be a strategy for Player 0 in $\mathcal{G}_f$. We show that σ is not winning by constructing counter strategies τ and τ' as above.

Proof

Let σ be a strategy for Player 0 in $\mathcal{G}_f$. We show that σ is not winning by constructing counter strategies τ and τ' as above.

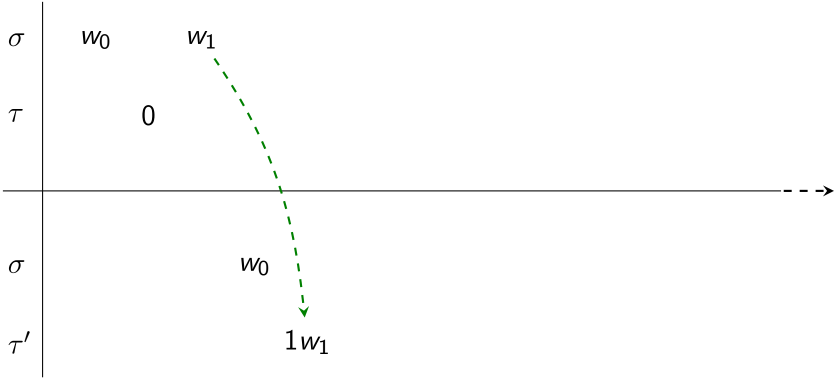
Proof

Let σ be a strategy for Player 0 in $\mathcal{G}_f$. We show that σ is not winning by constructing counter strategies τ and τ' as above.



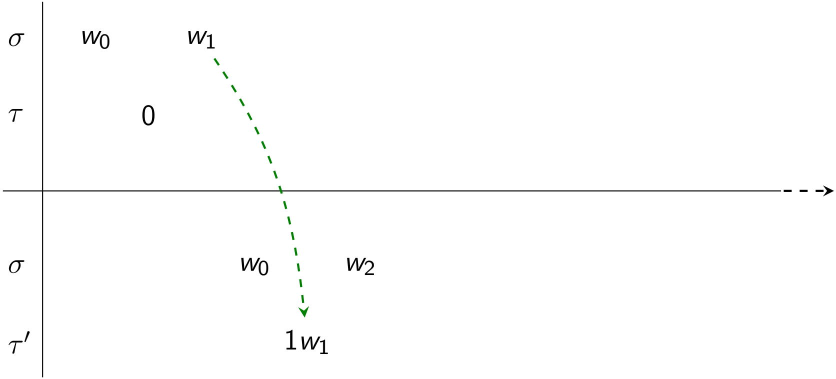
Proof

Let σ be a strategy for Player 0 in $\mathcal{G}_f$. We show that σ is not winning by constructing counter strategies τ and τ' as above.



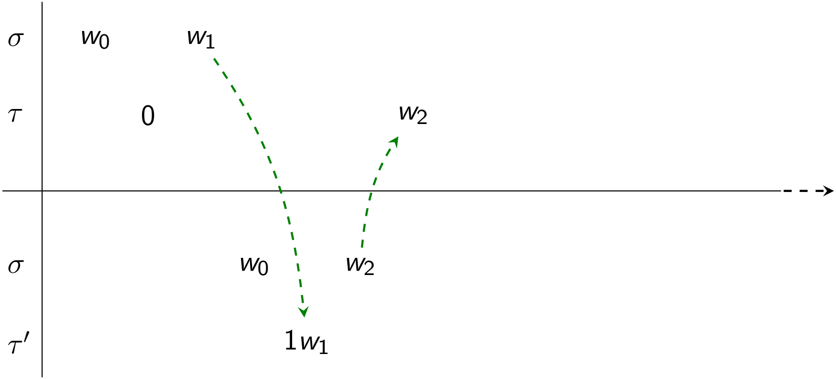
Proof

Let σ be a strategy for Player 0 in $\mathcal{G}_f$. We show that σ is not winning by constructing counter strategies τ and τ' as above.



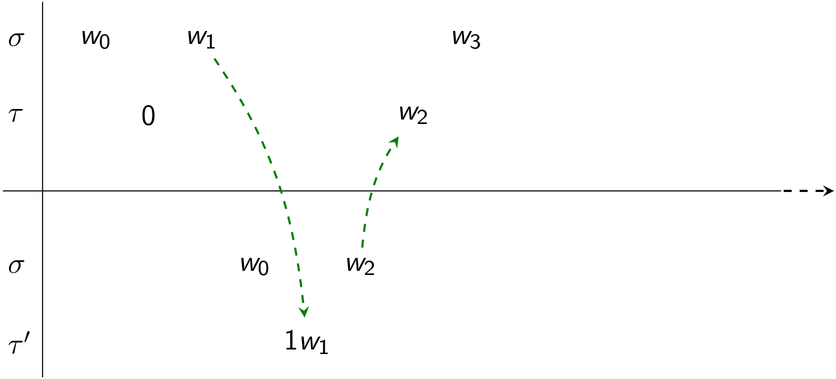
Proof

Let σ be a strategy for Player 0 in $\mathcal{G}_f$. We show that σ is not winning by constructing counter strategies τ and τ' as above.



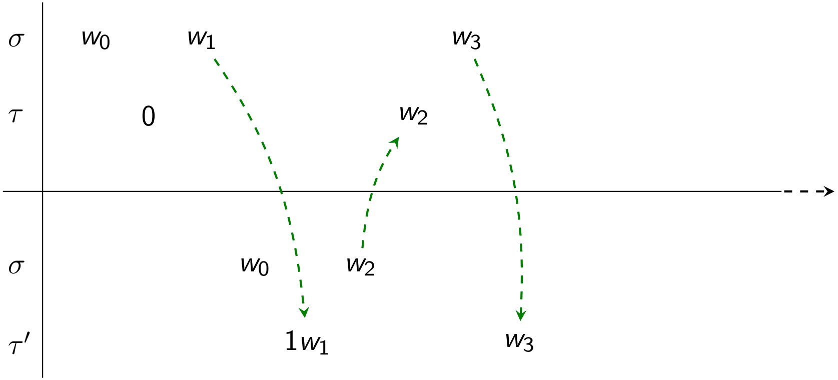
Proof

Let σ be a strategy for Player 0 in $\mathcal{G}_f$. We show that σ is not winning by constructing counter strategies τ and τ' as above.



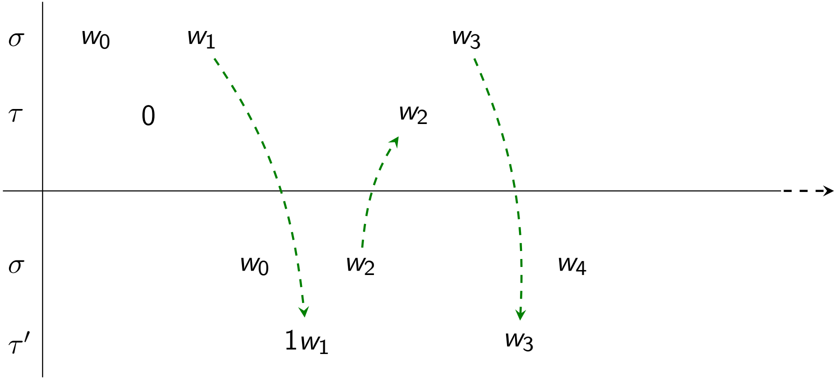
Proof

Let σ be a strategy for Player 0 in $\mathcal{G}_f$. We show that σ is not winning by constructing counter strategies τ and τ' as above.



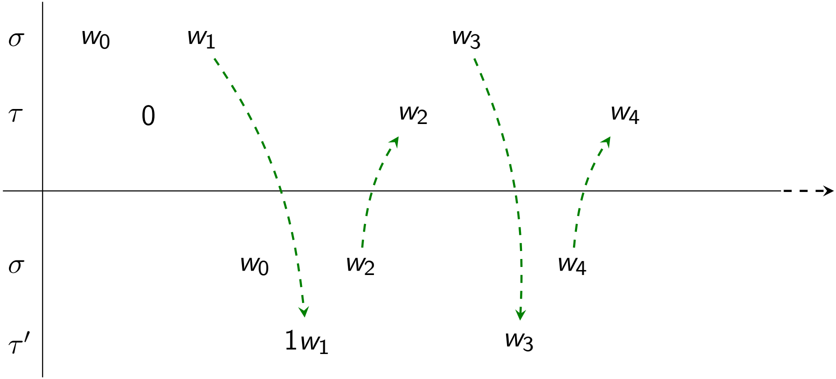
Proof

Let σ be a strategy for Player 0 in $\mathcal{G}_f$. We show that σ is not winning by constructing counter strategies τ and τ' as above.



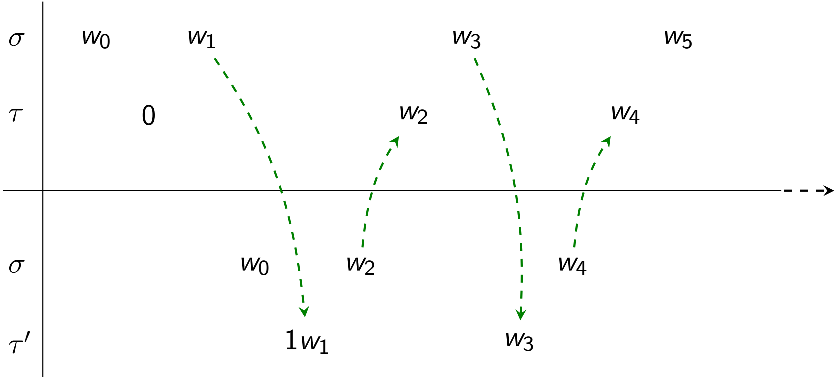
Proof

Let σ be a strategy for Player 0 in $\mathcal{G}_f$. We show that σ is not winning by constructing counter strategies τ and τ' as above.



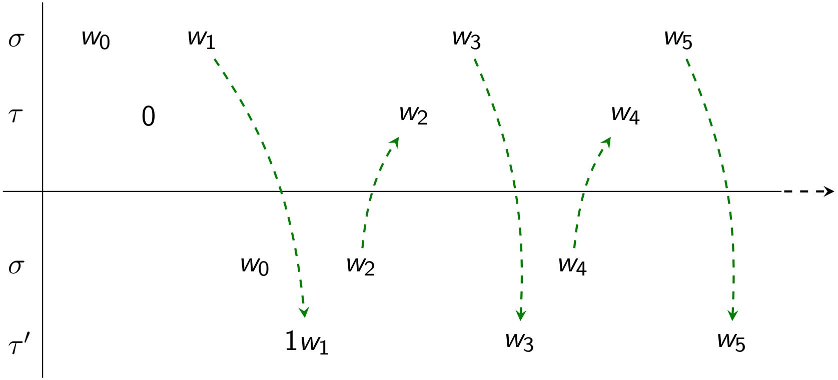
Proof

Let σ be a strategy for Player 0 in $\mathcal{G}_f$. We show that σ is not winning by constructing counter strategies τ and τ' as above.



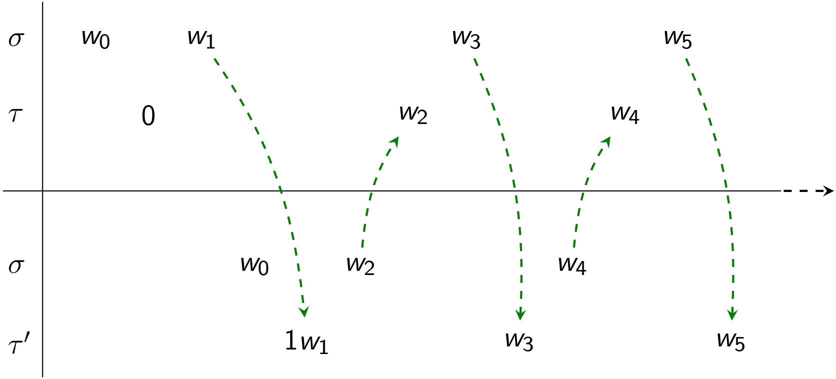
Proof

Let σ be a strategy for Player 0 in $\mathcal{G}_f$. We show that σ is not winning by constructing counter strategies τ and τ' as above.



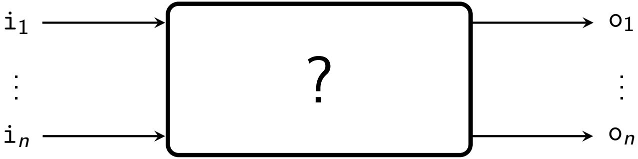
Proof

Let σ be a strategy for Player 0 in $\mathcal{G}_f$. We show that σ is not winning by constructing counter strategies τ and τ' as above.



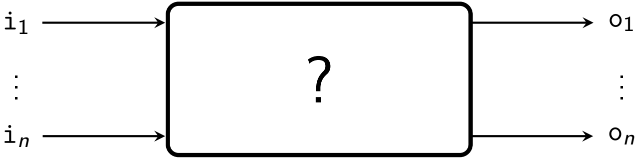
Consider the resulting plays: they differ only at their first position. Hence, Player 1 wins one of them. Thus, σ is not winning.

Church's Synthesis Problem



Church 1957: Given a specification on the input/output behavior of a circuit (in some suitable logical language), decide whether such a circuit exists, and, if yes, compute one.

Church's Synthesis Problem



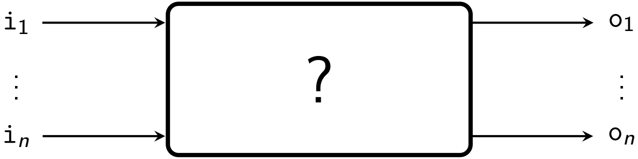
Example

Interpret input $\mathtt{i}_j = 1$ as client j requesting a shared resource and output $\mathtt{o}_j = 1$ as the corresponding grant to client j.

Typical properties:

1. Every request is eventually answered.
2. At most one grant at a time (mutual exclusion).
3. No spurious grants.

Church's Synthesis Problem



Solved by Büchi & Landweber in 1969.

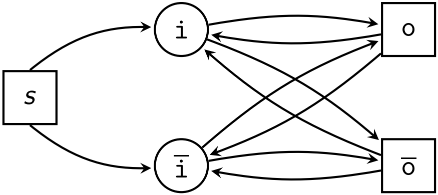
Insight: Problem can be expressed as two-player game of infinite duration between the environment (producing inputs) and the circuit (producing outputs).

Back to the Example

Consider the one-client case!

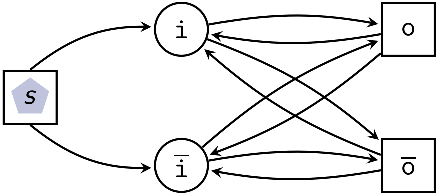
Back to the Example

Consider the one-client case!



Back to the Example

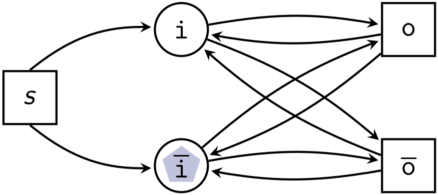
Consider the one-client case!



Input:
Output:

Back to the Example

Consider the one-client case!

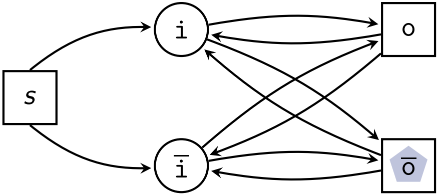


Input: 0

Output:

Back to the Example

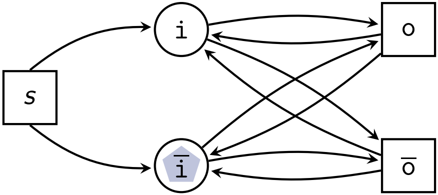
Consider the one-client case!



Input: 0
Output: 0

Back to the Example

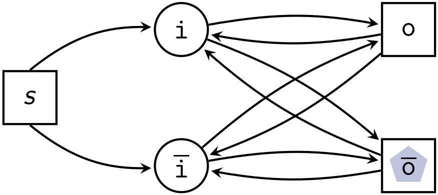
Consider the one-client case!



Input: 0 0
Output: 0

Back to the Example

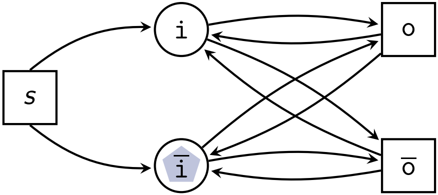
Consider the one-client case!



Input: 0 0
Output: 0 0

Back to the Example

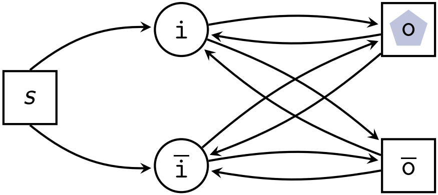
Consider the one-client case!



Input: 0 0 0
Output: 0 0

Back to the Example

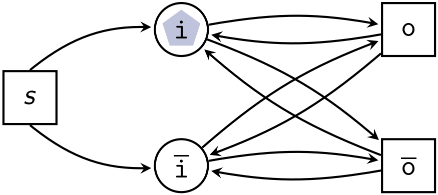
Consider the one-client case!



Input:	0	0	0
Output:	0	0	1

Back to the Example

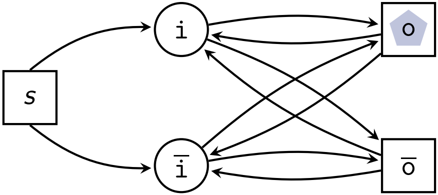
Consider the one-client case!



Input:	0	0	0	1
Output:	0	0	1	

Back to the Example

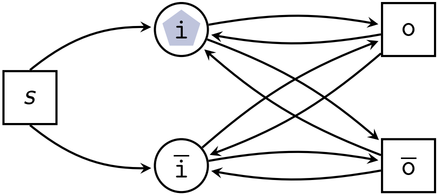
Consider the one-client case!



Input:	0	0	0	1
Output:	0	0	1	1

Back to the Example

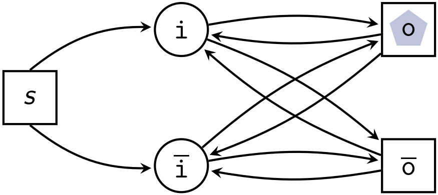
Consider the one-client case!



Input:	0	0	0	1	1
Output:	0	0	1	1	

Back to the Example

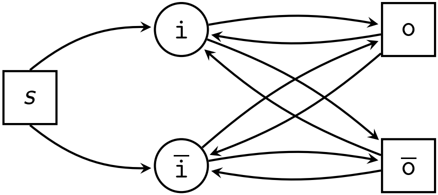
Consider the one-client case!



Input:	0	0	0	1	1
Output:	0	0	1	1	1

Back to the Example

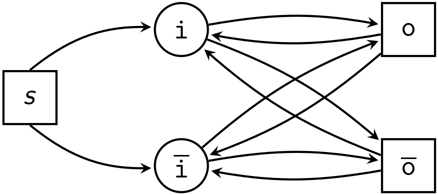
Consider the one-client case!



Input:	0	0	0	1	1	⋯
Output:	0	0	1	1	1	⋯

Back to the Example

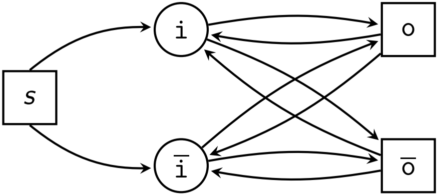
Consider the one-client case!



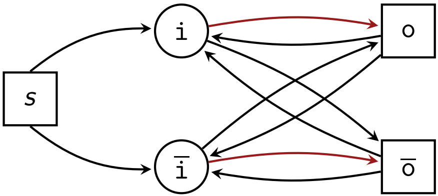
Winning plays for circuit player have to satisfy

1. if `i` is visited, then `o` as well at a later position, and
2. if `o` is visited, then it has not been visited since the last visit of `i`.

Büchi-Landweber in a Nutshell

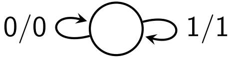


Büchi-Landweber in a Nutshell



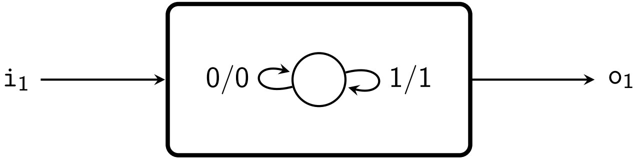
- Circuit player has a (memoryless) winning strategy,

Büchi-Landweber in a Nutshell



- Circuit player has a (memoryless) winning strategy,
- which can be turned into an automaton with output,

Büchi-Landweber in a Nutshell



- Circuit player has a (memoryless) winning strategy,
- which can be turned into an automaton with output,
- which can be turned into a circuit satisfying the specification.

Even More Games

- Logics
  - Ehrenfeucht Fraisse Games
- Set theory
  - Banach Mazur Games
  - Wadge Games
- Complexity theory
- Proof theory
- Automata theory
- Economics