Deterministic and game separability of tree languages via games

Lorenzo Clemente, Michał Skrzypczak

1/2

UW MIM

Parity index for ω-regular languages

Parity index for ω-regular languages

$C = \{i, \ldots, j\}$

Parity index for ω-regular languages

$C = \{i, \ldots, j\}$

$$P_C \stackrel{\text{def}}{=} \left\{\gamma \in C^\omega \mid \limsup_{n\to\infty} \gamma(n) \equiv 0 \pmod 2\right\}$$

Parity index for ω-regular languages

$$P_C \stackrel{\text{def}}{=} \left\{\gamma \in C^\omega \mid \limsup_{n\to\infty} \gamma(n) \equiv 0 \pmod 2\right\}$$

Problem Given an ω-regular language $L \subseteq A^\omega$ and $C = \{i, \ldots, j\}$,

can L be recognised by a deterministic C-parity automaton?

Parity index for ω-regular languages

$C = \{i, \ldots, j\}$

$$P_C \stackrel{\text{def}}{=} \left\{\gamma \in C^\omega \mid \limsup_{n\to\infty} \gamma(n) \equiv 0 \pmod 2\right\}$$

Problem Given an ω-regular language $L \subseteq A^\omega$ and $C = \{i, \ldots, j\}$,

can L be recognised by a deterministic C-parity automaton?

Game Played in rounds $n = 0, 1, \ldots$

Parity index for ω-regular languages

$C = \{i, \dots, j\}$

$$P_C \stackrel{\text{def}}{=} \left\{\gamma \in C^\omega \mid \limsup_{n\to\infty} \gamma(n) \equiv 0 \pmod 2\right\}$$

Problem Given an ω-regular language $L \subseteq A^\omega$ and $C = \{i, \dots, j\}$,

can L be recognised by a deterministic C-parity automaton?

Game Played in rounds $n = 0, 1, \dots$

$$\left[\quad \text{INPUT: } a_n \quad \right]$$

Parity index for ω-regular languages

$C = \{i, \ldots, j\}$

$$P_C \stackrel{\text{def}}{=} \left\{\gamma \in C^\omega \mid \limsup_{n\to\infty} \gamma(n) \equiv 0 \pmod 2\right\}$$

Problem Given an ω-regular language $L \subseteq A^\omega$ and $C = \{i, \ldots, j\}$,

can L be recognised by a deterministic C-parity automaton?

Game Played in rounds $n = 0, 1, \ldots$

$$\left[\begin{array}{rl} \text{INPUT:} & a_n \\ \text{AUTOMATON:} & c_n \end{array}\right]$$

Parity index for ω-regular languages

$C = \{i, \ldots, j\}$

$$P_C \stackrel{\text{def}}{=} \left\{\gamma \in C^\omega \mid \limsup_{n\to\infty} \gamma(n) \equiv 0 \pmod 2\right\}$$

Problem Given an ω-regular language $L \subseteq A^\omega$ and $C = \{i, \ldots, j\}$,

can L be recognised by a deterministic C-parity automaton?

Game Played in rounds $n = 0, 1, \ldots$

$$\left[\begin{array}{rl} \textsc{Input}: a_n & \rightsquigarrow \alpha \in A^\omega \\ \textsc{Automaton}: c_n & \rightsquigarrow \gamma \in C^\omega \end{array}\right]$$

Parity index for ω-regular languages

$C = \{i, \ldots, j\}$

$$P_C \stackrel{\text{def}}{=} \left\{\gamma \in C^\omega \mid \limsup_{n\to\infty} \gamma(n) \equiv 0 \pmod 2\right\}$$

Problem Given an ω-regular language $L \subseteq A^\omega$ and $C = \{i, \ldots, j\}$,

can L be recognised by a deterministic C-parity automaton?

Game Played in rounds $n = 0, 1, \ldots$

$$\left[\begin{array}{rll} \text{INPUT: } a_n & \rightsquigarrow & \alpha \in A^\omega \\ \text{AUTOMATON: } c_n & \rightsquigarrow & \gamma \in C^\omega \end{array}\right]$$

AUTOMATON wins
iff
$\big(\alpha \in L \iff \gamma \in P_C\big)$

Parity index for ω-regular languages

$C = \{i, \ldots, j\}$

$$P_C \stackrel{\text{def}}{=} \left\{\gamma \in C^\omega \mid \limsup_{n\to\infty} \gamma(n) \equiv 0 \pmod 2\right\}$$

Problem Given an ω-regular language $L \subseteq A^\omega$ and $C = \{i, \ldots, j\}$,

can L be recognised by a deterministic C-parity automaton?

Game Played in rounds $n = 0, 1, \ldots$

$$\left[\begin{array}{rll} \textsc{Input}: & a_n & \rightsquigarrow \alpha \in A^\omega \\ \textsc{Automaton}: & c_n & \rightsquigarrow \gamma \in C^\omega \end{array}\right]$$

$\textsc{Automaton}$ wins

iff

$$\underbrace{\left(\alpha \in L \iff \gamma \in P_C\right)}_{\omega\text{-regular}}$$

Parity index for ω-regular languages

$C = \{i, \dots, j\}$

$$P_C \stackrel{\text{def}}{=} \{\gamma \in C^\omega \mid \limsup_{n\to\infty} \gamma(n) \equiv 0 \pmod 2\}$$

Problem Given an ω-regular language $L \subseteq A^\omega$ and $C = \{i, \dots, j\}$,

can L be recognised by a deterministic C-parity automaton?

Game Played in rounds $n = 0, 1, \dots$

$$\begin{bmatrix} \text{INPUT: } a_n & \rightsquigarrow \alpha \in A^\omega \\ \text{AUTOMATON: } c_n & \rightsquigarrow \gamma \in C^\omega \end{bmatrix}$$

AUTOMATON wins

iff

$$\underbrace{\left(\alpha \in L \iff \gamma \in P_C\right)}_{\omega\text{-regular}}$$

Theorem (Büchi, Landweber '69)

ω-regular games are **effectively finite-memory** determined.

Parity index for ω-regular languages

$C = \{i, \ldots, j\}$

$$P_C \stackrel{\text{def}}{=} \{\gamma \in C^\omega \mid \limsup_{n\to\infty} \gamma(n) \equiv 0 \pmod 2\}$$

Problem Given an ω-regular language $L \subseteq A^\omega$ and $C = \{i, \ldots, j\}$,

can L be recognised by a deterministic C-parity automaton?

Game Played in rounds $n = 0, 1, \ldots$

$$\left[\begin{array}{rl} \textsc{Input}: a_n & \rightsquigarrow \alpha \in A^\omega \\ \textsc{Automaton}: c_n & \rightsquigarrow \gamma \in C^\omega \end{array}\right]$$

Automaton wins **iff**

$$\underbrace{\big(\alpha \in L \iff \gamma \in P_C\big)}_{\omega\text{-regular}}$$

Theorem (Büchi, Landweber '69)

ω**-regular** games are **effectively finite-memory** determined.

(**f-m** win. strategy of Automaton)

Parity index for ω-regular languages

$C = \{i, \ldots, j\}$

$$P_C \stackrel{\text{def}}{=} \{\gamma \in C^\omega \mid \limsup_{n\to\infty} \gamma(n) \equiv 0 \pmod 2\}$$

Problem Given an ω-regular language $L \subseteq A^\omega$ and $C = \{i, \ldots, j\}$,

can L be recognised by a deterministic C-parity automaton?

Game Played in rounds $n = 0, 1, \ldots$

$$\left[\begin{array}{rll} \text{Input: } a_n & \rightsquigarrow & \alpha \in A^\omega \\ \text{Automaton: } c_n & \rightsquigarrow & \gamma \in C^\omega \end{array}\right]$$

Automaton wins

iff

$$\underbrace{\left(\alpha \in L \iff \gamma \in P_C\right)}_{\omega\text{-regular}}$$

Theorem (Büchi, Landweber '69)

ω-**regular** games are **effectively finite-memory** determined.

(**f-m** win. strategy of Automaton) $\leftrightsquigarrow$

Parity index for ω-regular languages

$C = \{i, \ldots, j\}$

$$P_C \stackrel{\text{def}}{=} \left\{\gamma \in C^\omega \mid \limsup_{n\to\infty} \gamma(n) \equiv 0 \pmod 2\right\}$$

Problem Given an ω-regular language $L \subseteq A^\omega$ and $C = \{i, \ldots, j\}$,

can L be recognised by a deterministic C-parity automaton?

Game Played in rounds $n = 0, 1, \ldots$

$$\left[\begin{array}{rll} \text{INPUT: } a_n & \rightsquigarrow & \alpha \in A^\omega \\ \text{AUTOMATON: } c_n & \rightsquigarrow & \gamma \in C^\omega \end{array}\right]$$

AUTOMATON wins

iff

$$\underbrace{\left(\alpha \in L \iff \gamma \in P_C\right)}_{\omega\text{-regular}}$$

Theorem (Büchi, Landweber '69)

ω-**regular** games are **effectively finite-memory** determined.

(**f-m** win. strategy of AUTOMATON) $\leftrightsquigarrow$ (deterministic C-parity aut. for L)

Parity index for ω-regular languages

$C = \{i, \ldots, j\}$

$$P_C \stackrel{\text{def}}{=} \{\gamma \in C^\omega \mid \limsup_{n\to\infty} \gamma(n) \equiv 0 \pmod 2\}$$

Problem Given an ω-regular language $L \subseteq A^\omega$ and $C = \{i, \ldots, j\}$,

can L be recognised by a deterministic C-parity automaton?

Game Played in rounds $n = 0, 1, \ldots$

$$\left[\begin{array}{rll} \textsc{Input}: a_n & \rightsquigarrow & \alpha \in A^\omega \\ \textsc{Automaton}: c_n & \rightsquigarrow & \gamma \in C^\omega \end{array}\right]$$

Wadge game for $L \leqslant_{\mathrm{W}} P_C$

Automaton wins

iff

$$\underbrace{\left(\alpha \in L \iff \gamma \in P_C\right)}_{\omega\text{-regular}}$$

Theorem (Büchi, Landweber '69)

ω-**regular** games are **effectively finite-memory** determined.

(**f-m** win. strategy of Automaton) $\leftrightsquigarrow$ (deterministic C-parity aut. for L)

[Büchi, Landweber '69]

[Büchi, Landweber '69]

(**any win. strategy** of $\forall$)

[Büchi, Landweber '69]

(**any** **win. strategy** of $\forall$) (**f-m** **win. strategy** of $\exists$)

[Büchi, Landweber '69]

(**any win. strategy** of $\forall$) (**f-m win. strategy** of $\exists$)

$\Updownarrow$

mathematical counterexample

[Büchi, Landweber '69]

(**any win. strategy** of $\forall$) ⟷ **mathematical counterexample**

(**f-m win. strategy** of $\exists$) ⟷ **effective witness**

[Büchi, Landweber '69]

(**any win. strategy** of $\forall$) $\leftrightsquigarrow$ **mathematical counterexample**

(**f-m win. strategy** of $\exists$) $\leftrightsquigarrow$ **effective witness**

[Büchi, Landweber '69]

(**any** **win. strategy** of $\forall$) ⟷ **mathematical counterexample**

(**f-m** **win. strategy** of $\exists$) ⟷ **effective witness**

[Büchi, Landweber '69]

(**any** **win. strategy** of $\forall$)

mathematical counterexample

(**f-m** **win. strategy** of $\exists$)

effective witness

Theorem (Colcombet '13)

Domination games for **cost automata**.

[Büchi, Landweber '69]

(any **win. strategy** of $\forall$) ⇝ mathematical counterexample

(f-m **win. strategy** of $\exists$) ⇝ effective witness

Theorem (Colcombet '13)

Domination games for **cost automata**.

Theorem (Bojańczyk '15)

Star-height via **games**.

[Büchi, Landweber '69]

(**any** **win. strategy** of $\forall$) ⇝ **mathematical counterexample**

(**f-m** **win. strategy** of $\exists$) ⇝ **effective witness**

Theorem (Colcombet '13)

Domination games for **cost automata**.

Theorem (Bojańczyk '15)

Star-height via **games**.

Theorem (Zimmermann '15-'18)

Finite-memory strategies for **delay games**.

[Büchi, Landweber '69]

(**any win. strategy** of $\forall$) — (**f-m win. strategy** of $\exists$)

mathematical counterexample — **effective witness**

Theorem (Colcombet '13)

Domination games for **cost automata**.

Theorem (Bojańczyk '15)

Star-height via **games**.

Theorem (Zimmermann '15-'18)

Finite-memory strategies for **delay games**.

⋮

[Büchi, Landweber '69]

(any **win. strategy** of $\forall$) $\leftrightsquigarrow$ **mathematical counterexample**

(**f-m win. strategy** of $\exists$) $\leftrightsquigarrow$ **effective witness**

Theorem (Colcombet '13)

Domination games for **cost automata**.

Theorem (Bojańczyk '15)

Star-height via **games**.

Theorem (Zimmermann '15-'18)

Finite-memory strategies for **delay games**.

⋮

+ Rabin Separation Theorem

Disjointness Game

$\mathrm{Tr}_A \stackrel{\text{def}}{=} \{\mathtt{L}, \mathtt{R}\}^* \to A$

Disjointness Game

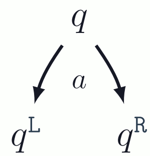
$\mathrm{Tr}_A \stackrel{\text{def}}{=} \{\mathtt{L}, \mathtt{R}\}^* \to A$

Disjointness Game

Non-deterministic tree automata:

$\mathrm{Tr}_A \stackrel{\text{def}}{=} \{\mathtt{L}, \mathtt{R}\}^* \to A$

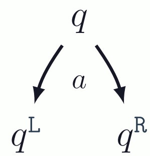
Disjointness Game



Non-deterministic tree automata: branching transitions

$\mathrm{Tr}_A \stackrel{\text{def}}{=} \{\mathtt{L}, \mathtt{R}\}^* \to A$

Disjointness Game

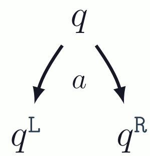


Non-deterministic tree automata: branching transitions

$\mathrm{L}(\mathcal{A}) \stackrel{\text{def}}{=} \{t \in \mathrm{Tr}_A \mid$

$\mathrm{Tr}_A \stackrel{\mathrm{def}}{=} \{\mathtt{L}, \mathtt{R}\}^* \to A$

Disjointness Game

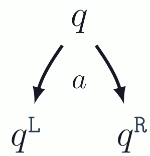


Non-deterministic tree automata: branching transitions

$$\mathrm{L}(\mathcal{A}) \stackrel{\mathrm{def}}{=} \{t \in \mathrm{Tr}_A \mid \underbrace{\exists \rho \in \mathrm{Tr}_Q}_{\text{run}}$$

$\mathrm{Tr}_A \stackrel{\text{def}}{=} \{\mathtt{L}, \mathtt{R}\}^* \to A$

Disjointness Game

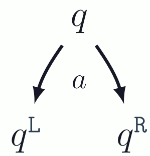


Non-deterministic tree automata: branching transitions

$$\mathrm{L}(\mathcal{A}) \stackrel{\text{def}}{=} \{t \in \mathrm{Tr}_A \mid \underbrace{\exists \rho \in \mathrm{Tr}_Q.}_{\text{run}} \underbrace{\forall \pi \in \{\mathtt{L}, \mathtt{R}\}^\omega}_{\text{branch}}$$

$\mathrm{Tr}_A \stackrel{\mathrm{def}}{=} \{\mathtt{L}, \mathtt{R}\}^* \to A$

Disjointness Game

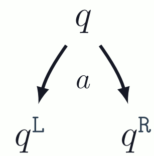


Non-deterministic tree automata: branching transitions

$$\mathrm{L}(\mathcal{A}) \stackrel{\mathrm{def}}{=} \{t \in \mathrm{Tr}_A \mid \underbrace{\exists \rho \in \mathrm{Tr}_Q}_{\text{run}}.\ \underbrace{\forall \pi \in \{\mathtt{L}, \mathtt{R}\}^\omega}_{\text{branch}}.\ \text{states of } \rho \text{ on } \pi \text{ are accepting}\}$$

$\mathrm{Tr}_A \stackrel{\mathrm{def}}{=} \{\mathtt{L}, \mathtt{R}\}^* \to A$

Disjointness Game



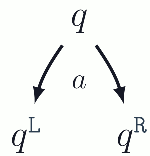
Non-deterministic tree automata: branching transitions

$$\mathrm{L}(\mathcal{A}) \stackrel{\mathrm{def}}{=} \{t \in \mathrm{Tr}_A \mid \underbrace{\exists \rho \in \mathrm{Tr}_Q}_{\text{run}}.\ \underbrace{\forall \pi \in \{\mathtt{L}, \mathtt{R}\}^\omega}_{\text{branch}}.\ \text{states of } \rho \text{ on } \pi \text{ are accepting}\}$$

Problem Given two automata $\mathcal{A}$ and $\mathcal{B}$, decide whether $\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B}) \neq \emptyset$?

$\mathrm{Tr}_A \stackrel{\text{def}}{=} \{\mathtt{L}, \mathtt{R}\}^* \to A$

Disjointness Game



Non-deterministic tree automata: branching transitions

$$\mathrm{L}(\mathcal{A}) \stackrel{\text{def}}{=} \{t \in \mathrm{Tr}_A \mid \underbrace{\exists \rho \in \mathrm{Tr}_Q}_{\text{run}}.\ \underbrace{\forall \pi \in \{\mathtt{L}, \mathtt{R}\}^\omega}_{\text{branch}}.\ \text{states of } \rho \text{ on } \pi \text{ are accepting}\}$$

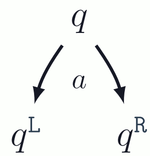
Problem Given two automata $\mathcal{A}$ and $\mathcal{B}$, decide whether $\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B}) \neq \emptyset$?

$$\left[\ (p_n, q_n) \to \qquad\qquad\qquad\qquad\qquad \right]$$

$\mathrm{Tr}_A \stackrel{\text{def}}{=} \{\mathtt{L}, \mathtt{R}\}^* \to A$

Disjointness Game

Non-deterministic tree automata: branching transitions

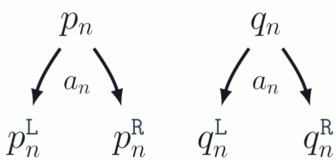


$$\mathrm{L}(\mathcal{A}) \stackrel{\text{def}}{=} \{t \in \mathrm{Tr}_A \mid \underbrace{\exists \rho \in \mathrm{Tr}_Q}_{\text{run}}. \underbrace{\forall \pi \in \{\mathtt{L}, \mathtt{R}\}^\omega}_{\text{branch}}. \text{ states of } \rho \text{ on } \pi \text{ are accepting}\}$$

Problem Given two automata $\mathcal{A}$ and $\mathcal{B}$, decide whether $\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B}) \neq \emptyset$?

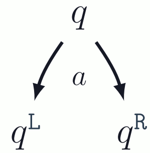
$(p_n, q_n) \to$

AUTOMATON:



$\mathrm{Tr}_A \stackrel{\text{def}}{=} \{\mathtt{L}, \mathtt{R}\}^* \to A$

Disjointness Game



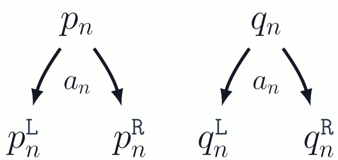
Non-deterministic tree automata: branching transitions

$$\mathrm{L}(\mathcal{A}) \stackrel{\text{def}}{=} \{t \in \mathrm{Tr}_A \mid \underbrace{\exists \rho \in \mathrm{Tr}_Q.}_{\text{run}}\ \underbrace{\forall \pi \in \{\mathtt{L}, \mathtt{R}\}^\omega.}_{\text{branch}} \text{ states of } \rho \text{ on } \pi \text{ are accepting}\}$$

Problem Given two automata $\mathcal{A}$ and $\mathcal{B}$, decide whether $\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B}) \neq \varnothing$?

$(p_n, q_n) \to$

AUTOMATON:

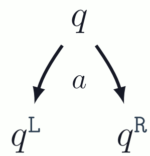


PATHFINDER: $d_n \in \{\mathtt{L}, \mathtt{R}\}$

$\mathrm{Tr}_A \stackrel{\mathrm{def}}{=} \{\mathtt{L}, \mathtt{R}\}^* \to A$

Disjointness Game

Non-deterministic tree automata: branching transitions

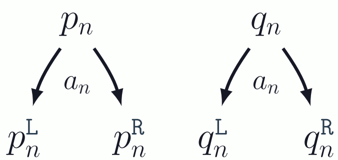


$$\mathrm{L}(\mathcal{A}) \stackrel{\mathrm{def}}{=} \{t \in \mathrm{Tr}_A \mid \underbrace{\exists \rho \in \mathrm{Tr}_Q.}_{\text{run}} \underbrace{\forall \pi \in \{\mathtt{L}, \mathtt{R}\}^\omega.}_{\text{branch}} \text{ states of } \rho \text{ on } \pi \text{ are accepting}\}$$

Problem Given two automata $\mathcal{A}$ and $\mathcal{B}$, decide whether $\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B}) \neq \emptyset$?

$(p_n, q_n) \to$

AUTOMATON:

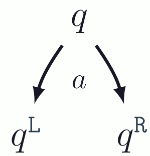


PATHFINDER: $d_n \in \{\mathtt{L}, \mathtt{R}\}$

$\to (p_n^{d_n}, q_n^{d_n})$

$\mathrm{Tr}_A \stackrel{\text{def}}{=} \{\mathtt{L}, \mathtt{R}\}^* \to A$

Disjointness Game



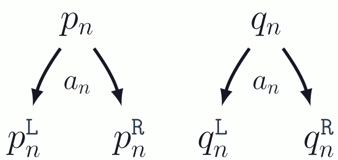
Non-deterministic tree automata: branching transitions

$$\mathrm{L}(\mathcal{A}) \stackrel{\text{def}}{=} \{t \in \mathrm{Tr}_A \mid \underbrace{\exists \rho \in \mathrm{Tr}_Q.}_{\text{run}} \underbrace{\forall \pi \in \{\mathtt{L}, \mathtt{R}\}^\omega.}_{\text{branch}} \text{ states of } \rho \text{ on } \pi \text{ are accepting}\}$$

Problem Given two automata $\mathcal{A}$ and $\mathcal{B}$, decide whether $\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B}) \neq \emptyset$?

$(p_n, q_n) \to$

AUTOMATON:



PATHFINDER: $d_n \in \{\mathtt{L}, \mathtt{R}\}$

$\to (p_n^{d_n}, q_n^{d_n})$

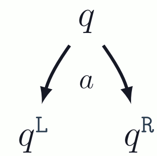
AUTOMATON wins

iff

$(p_n)_{n\in\omega}$ and $(q_n)_{n\in\omega}$ are accepting

$\mathrm{Tr}_A \stackrel{\text{def}}{=} \{\mathtt{L},\mathtt{R}\}^* \to A$

Disjointness Game



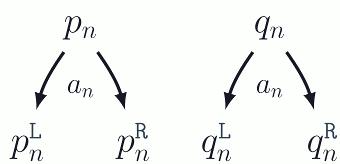
Non-deterministic tree automata: branching transitions

$$\mathrm{L}(\mathcal{A}) \stackrel{\text{def}}{=} \{t \in \mathrm{Tr}_A \mid \underbrace{\exists \rho \in \mathrm{Tr}_Q.}_{\text{run}}\ \underbrace{\forall \pi \in \{\mathtt{L},\mathtt{R}\}^\omega.}_{\text{branch}}\ \text{states of } \rho \text{ on } \pi \text{ are accepting}\}$$

Problem Given two automata $\mathcal{A}$ and $\mathcal{B}$, decide whether $\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B}) \neq \emptyset$?

$(p_n, q_n) \to$

AUTOMATON:



PATHFINDER: $d_n \in \{\mathtt{L},\mathtt{R}\}$

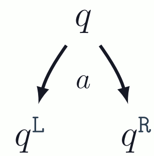
$\to (p_n^{d_n}, q_n^{d_n})$

AUTOMATON wins
iff
$(p_n)_{n\in\omega}$ and $(q_n)_{n\in\omega}$ are accepting

PATHFINDER wins
iff
one of $(p_n)_{n\in\omega}$, $(q_n)_{n\in\omega}$ is rejecting

$\mathrm{Tr}_A \stackrel{\text{def}}{=} \{\mathtt{L}, \mathtt{R}\}^* \to A$

Disjointness Game



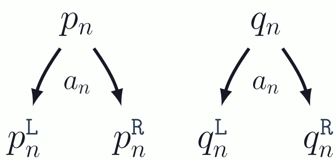
Non-deterministic tree automata: branching transitions

$$\mathrm{L}(\mathcal{A}) \stackrel{\text{def}}{=} \{t \in \mathrm{Tr}_A \mid \underbrace{\exists \rho \in \mathrm{Tr}_Q.}_{\text{run}} \underbrace{\forall \pi \in \{\mathtt{L}, \mathtt{R}\}^\omega.}_{\text{branch}} \text{ states of } \rho \text{ on } \pi \text{ are accepting}\}$$

Problem Given two automata $\mathcal{A}$ and $\mathcal{B}$, decide whether $\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B}) \neq \emptyset$?

$(p_n, q_n) \to$

AUTOMATON:



PATHFINDER: $d_n \in \{\mathtt{L}, \mathtt{R}\}$

$\to (p_n^{d_n}, q_n^{d_n})$

AUTOMATON wins
iff
$(p_n)_{n\in\omega}$ and $(q_n)_{n\in\omega}$ are accepting

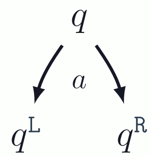
PATHFINDER wins
iff
one of $(p_n)_{n\in\omega}$, $(q_n)_{n\in\omega}$ is rejecting

(win. strategy of AUTOMATON)

$\mathrm{Tr}_A \stackrel{\text{def}}{=} \{\mathtt{L}, \mathtt{R}\}^* \to A$

Disjointness Game

Non-deterministic tree automata: branching transitions

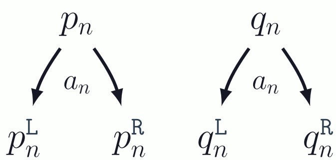


$$\mathrm{L}(\mathcal{A}) \stackrel{\text{def}}{=} \{t \in \mathrm{Tr}_A \mid \underbrace{\exists \rho \in \mathrm{Tr}_Q.}_{\text{run}} \underbrace{\forall \pi \in \{\mathtt{L}, \mathtt{R}\}^\omega.}_{\text{branch}} \text{ states of } \rho \text{ on } \pi \text{ are accepting}\}$$

Problem Given two automata $\mathcal{A}$ and $\mathcal{B}$, decide whether $\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B}) \neq \emptyset$?

$(p_n, q_n) \to$

AUTOMATON:



PATHFINDER: $d_n \in \{\mathtt{L}, \mathtt{R}\}$

$\to (p_n^{d_n}, q_n^{d_n})$

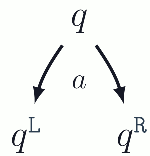
AUTOMATON wins
iff
$(p_n)_{n\in\omega}$ and $(q_n)_{n\in\omega}$ are accepting

PATHFINDER wins
iff
one of $(p_n)_{n\in\omega}$, $(q_n)_{n\in\omega}$ is rejecting

(win. strategy of AUTOMATON) $\leftrightsquigarrow$

$\mathrm{Tr}_A \stackrel{\text{def}}{=} \{\mathtt{L}, \mathtt{R}\}^* \to A$

Disjointness Game



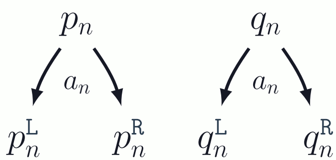
Non-deterministic tree automata: branching transitions

$$\mathrm{L}(\mathcal{A}) \stackrel{\text{def}}{=} \{t \in \mathrm{Tr}_A \mid \underbrace{\exists \rho \in \mathrm{Tr}_Q.}_{\text{run}} \underbrace{\forall \pi \in \{\mathtt{L}, \mathtt{R}\}^\omega.}_{\text{branch}} \text{ states of } \rho \text{ on } \pi \text{ are accepting}\}$$

Problem Given two automata $\mathcal{A}$ and $\mathcal{B}$, decide whether $\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B}) \neq \emptyset$?

$(p_n, q_n) \to$

Automaton:



Pathfinder: $d_n \in \{\mathtt{L}, \mathtt{R}\}$

$\to (p_n^{d_n}, q_n^{d_n})$

Automaton wins
iff
$(p_n)_{n\in\omega}$ and $(q_n)_{n\in\omega}$ are accepting

Pathfinder wins
iff
one of $(p_n)_{n\in\omega}$, $(q_n)_{n\in\omega}$ is rejecting

(win. strategy of Automaton) $\leftrightsquigarrow$ (a tree $t \in \mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B})$)

$\mathrm{Tr}_A \stackrel{\text{def}}{=} \{\mathtt{L}, \mathtt{R}\}^* \to A$

Disjointness Game

Non-deterministic tree automata: branching transitions $q \xrightarrow{a} (q^{\mathtt{L}}, q^{\mathtt{R}})$

$$\mathrm{L}(\mathcal{A}) \stackrel{\text{def}}{=} \{t \in \mathrm{Tr}_A \mid \underbrace{\exists \rho \in \mathrm{Tr}_Q.}_{\text{run}}\ \underbrace{\forall \pi \in \{\mathtt{L}, \mathtt{R}\}^\omega.}_{\text{branch}}\ \text{states of } \rho \text{ on } \pi \text{ are accepting}\}$$

Problem Given two automata $\mathcal{A}$ and $\mathcal{B}$, decide whether $\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B}) \neq \emptyset$?

$(p_n, q_n) \to$

Automaton: $p_n \xrightarrow{a_n} (p_n^{\mathtt{L}}, p_n^{\mathtt{R}})$, $q_n \xrightarrow{a_n} (q_n^{\mathtt{L}}, q_n^{\mathtt{R}})$

Pathfinder: $d_n \in \{\mathtt{L}, \mathtt{R}\}$

$\to (p_n^{d_n}, q_n^{d_n})$

Automaton wins
iff
$(p_n)_{n\in\omega}$ and $(q_n)_{n\in\omega}$ are accepting

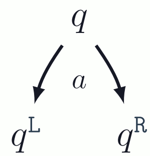
Pathfinder wins
iff
$\underbrace{\text{one of } (p_n)_{n\in\omega},\ (q_n)_{n\in\omega} \text{ is rejecting}}_{\text{Rabin condition}}$

(win. strategy of Automaton) $\leftrightsquigarrow$ (a tree $t \in \mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B})$)

$\mathrm{Tr}_A \stackrel{\text{def}}{=} \{\mathtt{L},\mathtt{R}\}^* \to A$

Disjointness Game

Non-deterministic tree automata: branching transitions

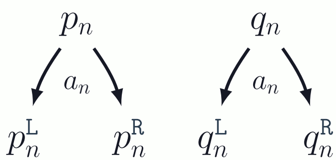


$$\mathrm{L}(\mathcal{A}) \stackrel{\text{def}}{=} \{t \in \mathrm{Tr}_A \mid \underbrace{\exists \rho \in \mathrm{Tr}_Q.}_{\text{run}}\ \underbrace{\forall \pi \in \{\mathtt{L},\mathtt{R}\}^\omega.}_{\text{branch}} \text{ states of } \rho \text{ on } \pi \text{ are accepting}\}$$

Problem Given two automata $\mathcal{A}$ and $\mathcal{B}$, decide whether $\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B}) \neq \varnothing$?

$(p_n, q_n) \to$

AUTOMATON:



PATHFINDER: $d_n \in \{\mathtt{L},\mathtt{R}\}$

$\to (p_n^{d_n}, q_n^{d_n})$

AUTOMATON wins
iff
$(p_n)_{n\in\omega}$ and $(q_n)_{n\in\omega}$ are accepting

PATHFINDER wins
iff
$\underbrace{\text{one of } (p_n)_{n\in\omega},\ (q_n)_{n\in\omega} \text{ is rejecting}}_{\text{Rabin condition}}$

(win. strategy of AUTOMATON) $\leftrightsquigarrow$ (a tree $t \in \mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B})$)

\+ **positional** determinacy for PATHFINDER

Rabin Separation Theorem

Rabin Separation Theorem

Theorem (Rabin '70)

If $\mathcal{A}$, $\mathcal{B}$ are Büchi tree automata and $\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B}) = \emptyset$

Rabin Separation Theorem

Theorem (Rabin '70)

If $\mathcal{A}$, $\mathcal{B}$ are Büchi tree automata and $\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B}) = \emptyset$

then there exists a **weak alternating** automaton $\mathcal{S}$

Rabin Separation Theorem

Theorem (Rabin '70)

If $\mathcal{A}$, $\mathcal{B}$ are Büchi tree automata and $L(\mathcal{A}) \cap L(\mathcal{B}) = \emptyset$

then there exists a **weak alternating** automaton $\mathcal{S}$

that **separates** $L(\mathcal{A})$ and $L(\mathcal{B})$.

Rabin Separation Theorem

Theorem (Rabin '70)

If $\mathcal{A}$, $\mathcal{B}$ are Büchi tree automata and $\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B}) = \varnothing$

then there exists a **weak alternating** automaton $\mathcal{S}$

that **separates** $\mathrm{L}(\mathcal{A})$ and $\mathrm{L}(\mathcal{B})$.

$$\underbrace{}_{\mathrm{L}(\mathcal{A}) \subseteq \mathrm{L}(\mathcal{S}) \text{ and } \mathrm{L}(\mathcal{B}) \subseteq \mathrm{L}(\mathcal{S})^{\mathrm{c}}}$$

Rabin Separation Theorem

Theorem (Rabin '70)

If $\mathcal{A}$, $\mathcal{B}$ are Büchi tree automata and $\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B}) = \emptyset$

then there exists a **weak alternating** automaton $\mathcal{S}$

$$\underbrace{\text{that **separates** } \mathrm{L}(\mathcal{A}) \text{ and } \mathrm{L}(\mathcal{B}).}_{\mathrm{L}(\mathcal{A}) \subseteq \mathrm{L}(\mathcal{S}) \text{ and } \mathrm{L}(\mathcal{B}) \subseteq \mathrm{L}(\mathcal{S})^{\mathrm{c}}}$$

$\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{S})^{\mathrm{c}} = \emptyset$ and $\mathrm{L}(\mathcal{B}) \cap \mathrm{L}(\mathcal{S}) = \emptyset$

Rabin Separation Theorem

Theorem (Rabin '70)

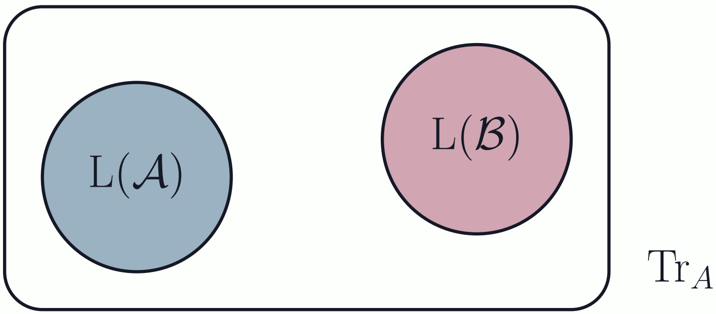
If $\mathcal{A}$, $\mathcal{B}$ are Büchi tree automata and $L(\mathcal{A}) \cap L(\mathcal{B}) = \emptyset$

then there exists a **weak alternating** automaton $\mathcal{S}$

that **separates** $L(\mathcal{A})$ and $L(\mathcal{B})$.

$L(\mathcal{A}) \subseteq L(\mathcal{S})$ and $L(\mathcal{B}) \subseteq L(\mathcal{S})^c$

$L(\mathcal{A}) \cap L(\mathcal{S})^c = \emptyset$ and $L(\mathcal{B}) \cap L(\mathcal{S}) = \emptyset$



Rabin Separation Theorem

Theorem (Rabin '70)

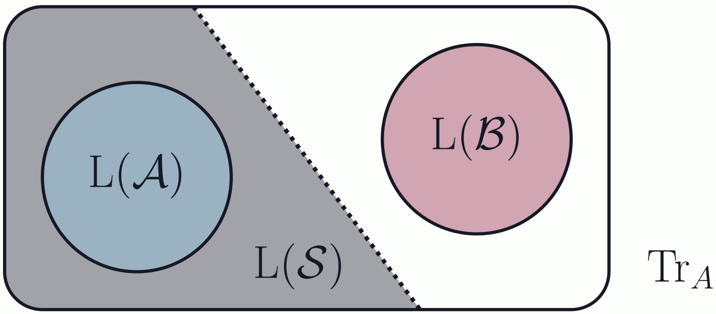
If $\mathcal{A}$, $\mathcal{B}$ are Büchi tree automata and $L(\mathcal{A}) \cap L(\mathcal{B}) = \emptyset$

then there exists a **weak alternating** automaton $\mathcal{S}$

that **separates** $L(\mathcal{A})$ and $L(\mathcal{B})$.

$L(\mathcal{A}) \subseteq L(\mathcal{S})$ and $L(\mathcal{B}) \subseteq L(\mathcal{S})^c$

$L(\mathcal{A}) \cap L(\mathcal{S})^c = \emptyset$ and $L(\mathcal{B}) \cap L(\mathcal{S}) = \emptyset$



Rabin Separation Theorem

Theorem (Rabin '70)

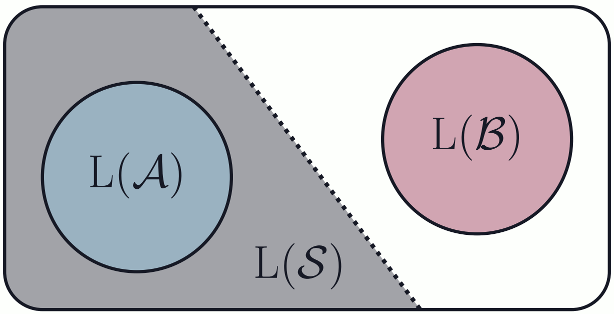
If $\mathcal{A}$, $\mathcal{B}$ are Büchi tree automata and $L(\mathcal{A}) \cap L(\mathcal{B}) = \emptyset$

then there exists a **weak alternating** automaton $\mathcal{S}$

that **separates** $L(\mathcal{A})$ and $L(\mathcal{B})$.

$L(\mathcal{A}) \subseteq L(\mathcal{S})$ and $L(\mathcal{B}) \subseteq L(\mathcal{S})^c$

$L(\mathcal{A}) \cap L(\mathcal{S})^c = \emptyset$ and $L(\mathcal{B}) \cap L(\mathcal{S}) = \emptyset$



Tr_A

If $L(\mathcal{B}) = L(\mathcal{A})^c$ then $L(\mathcal{S}) = L(\mathcal{A})$.

Rabin Separation Theorem

Theorem (Rabin '70)

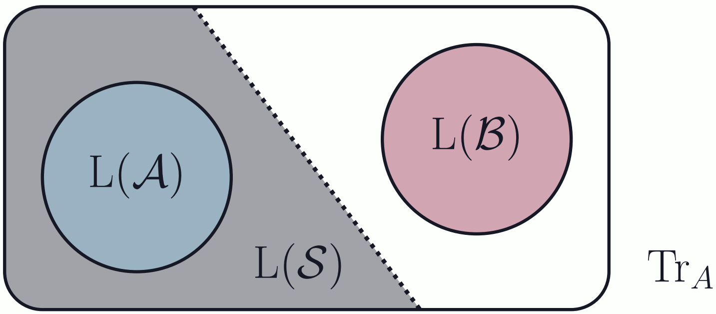
If $\mathcal{A}$, $\mathcal{B}$ are Büchi tree automata and $L(\mathcal{A}) \cap L(\mathcal{B}) = \emptyset$

then there exists a **weak alternating** automaton $\mathcal{S}$

that **separates** $L(\mathcal{A})$ and $L(\mathcal{B})$.

$$L(\mathcal{A}) \subseteq L(\mathcal{S}) \text{ and } L(\mathcal{B}) \subseteq L(\mathcal{S})^c$$

$$L(\mathcal{A}) \cap L(\mathcal{S})^c = \emptyset \text{ and } L(\mathcal{B}) \cap L(\mathcal{S}) = \emptyset$$



If $L(\mathcal{B}) = L(\mathcal{A})^c$ then $L(\mathcal{S}) = L(\mathcal{A})$.

$\rightsquigarrow$ **separation** $\Longrightarrow$ **membership**

Rabin Separation Theorem

Theorem (Rabin '70)

If $\mathcal{A}$, $\mathcal{B}$ are Büchi tree automata and $\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B}) = \emptyset$

then there exists a **weak alternating** automaton $\mathcal{S}$

that **separates** $\mathrm{L}(\mathcal{A})$ and $\mathrm{L}(\mathcal{B})$.

$\mathrm{L}(\mathcal{A}) \subseteq \mathrm{L}(\mathcal{S})$ and $\mathrm{L}(\mathcal{B}) \subseteq \mathrm{L}(\mathcal{S})^{\mathrm{c}}$

$\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{S})^{\mathrm{c}} = \emptyset$ and $\mathrm{L}(\mathcal{B}) \cap \mathrm{L}(\mathcal{S}) = \emptyset$

Rabin Separation Theorem

Theorem (Rabin '70)

If $\mathcal{A}$, $\mathcal{B}$ are Büchi tree automata and $\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B}) = \emptyset$

then there exists a **weak alternating** automaton $\mathcal{S}$

that **separates** $\mathrm{L}(\mathcal{A})$ and $\mathrm{L}(\mathcal{B})$.

$$\mathrm{L}(\mathcal{A}) \subseteq \mathrm{L}(\mathcal{S}) \text{ and } \mathrm{L}(\mathcal{B}) \subseteq \mathrm{L}(\mathcal{S})^{c}$$

$$\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{S})^{c} = \emptyset \text{ and } \mathrm{L}(\mathcal{B}) \cap \mathrm{L}(\mathcal{S}) = \emptyset$$

Proof

Rabin Separation Theorem

Theorem (Rabin '70)

If $\mathcal{A}$, $\mathcal{B}$ are Büchi tree automata and $\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B}) = \emptyset$

then there exists a **weak alternating** automaton $\mathcal{S}$

that **separates** $\mathrm{L}(\mathcal{A})$ and $\mathrm{L}(\mathcal{B})$.

$$\underbrace{\hphantom{xxxxxxxxxx}}$$

$\mathrm{L}(\mathcal{A}) \subseteq \mathrm{L}(\mathcal{S})$ and $\mathrm{L}(\mathcal{B}) \subseteq \mathrm{L}(\mathcal{S})^{\mathrm{c}}$

$\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{S})^{\mathrm{c}} = \emptyset$ and $\mathrm{L}(\mathcal{B}) \cap \mathrm{L}(\mathcal{S}) = \emptyset$

Proof

1. Fix a **positional** (or **f-m**) win. strategy $\boldsymbol{\sigma}$ of Pathfinder

in the disjointness game for $\mathcal{A}$, $\mathcal{B}$.

Rabin Separation Theorem

Theorem (Rabin '70)

If $\mathcal{A}$, $\mathcal{B}$ are Büchi tree automata and $L(\mathcal{A}) \cap L(\mathcal{B}) = \emptyset$

then there exists a **weak alternating** automaton $\mathcal{S}$

that **separates** $L(\mathcal{A})$ and $L(\mathcal{B})$.

$$\underbrace{\text{that separates } L(\mathcal{A}) \text{ and } L(\mathcal{B})}_{L(\mathcal{A}) \subseteq L(\mathcal{S}) \text{ and } L(\mathcal{B}) \subseteq L(\mathcal{S})^c}$$

$L(\mathcal{A}) \cap L(\mathcal{S})^c = \emptyset$ and $L(\mathcal{B}) \cap L(\mathcal{S}) = \emptyset$

Proof

1. Fix a **positional** (or **f-m**) win. strategy $\boldsymbol{\sigma}$ of Pathfinder in the disjointness game for $\mathcal{A}$, $\mathcal{B}$.

2. Prove that $\boldsymbol{\sigma}$ cannot loop between accepting states of $\mathcal{A}$ and $\mathcal{B}$.

Rabin Separation Theorem

Theorem (Rabin '70)

If $\mathcal{A}$, $\mathcal{B}$ are Büchi tree automata and $\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B}) = \emptyset$

then there exists a **weak alternating** automaton $\mathcal{S}$

that **separates** $\mathrm{L}(\mathcal{A})$ and $\mathrm{L}(\mathcal{B})$.

$\mathrm{L}(\mathcal{A}) \subseteq \mathrm{L}(\mathcal{S})$ and $\mathrm{L}(\mathcal{B}) \subseteq \mathrm{L}(\mathcal{S})^{\mathrm{c}}$

$\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{S})^{\mathrm{c}} = \emptyset$ and $\mathrm{L}(\mathcal{B}) \cap \mathrm{L}(\mathcal{S}) = \emptyset$

Proof

1. Fix a **positional** (or **f-m**) win. strategy $\boldsymbol{\sigma}$ of Pathfinder in the disjointness game for $\mathcal{A}$, $\mathcal{B}$.

2. Prove that $\boldsymbol{\sigma}$ cannot loop between accepting states of $\mathcal{A}$ and $\mathcal{B}$.

3. Stratify $\mathcal{A} \times \mathcal{B} \times \boldsymbol{\sigma}$ into **layers**.

Rabin Separation Theorem

Theorem (Rabin '70)

If $\mathcal{A}$, $\mathcal{B}$ are Büchi tree automata and $\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B}) = \varnothing$

then there exists a **weak alternating** automaton $\mathcal{S}$

that **separates** $\mathrm{L}(\mathcal{A})$ and $\mathrm{L}(\mathcal{B})$.

$\mathrm{L}(\mathcal{A}) \subseteq \mathrm{L}(\mathcal{S})$ and $\mathrm{L}(\mathcal{B}) \subseteq \mathrm{L}(\mathcal{S})^{\mathrm{c}}$

$\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{S})^{\mathrm{c}} = \varnothing$ and $\mathrm{L}(\mathcal{B}) \cap \mathrm{L}(\mathcal{S}) = \varnothing$

Proof

1. Fix a **positional** (or **f-m**) win. strategy $\boldsymbol{\sigma}$ of Pathfinder in the disjointness game for $\mathcal{A}$, $\mathcal{B}$.

2. Prove that $\boldsymbol{\sigma}$ cannot loop between accepting states of $\mathcal{A}$ and $\mathcal{B}$.

3. Stratify $\mathcal{A} \times \mathcal{B} \times \boldsymbol{\sigma}$ into **layers**.

4. $\rightsquigarrow$ a **weak alternating** automaton $\mathcal{S}$.

■

Rabin Separation Theorem

Theorem (Rabin '70)

If $\mathcal{A}$, $\mathcal{B}$ are Büchi tree automata and $\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{B}) = \emptyset$

then there exists a **weak alternating** automaton $\mathcal{S}$

that **separates** $\mathrm{L}(\mathcal{A})$ and $\mathrm{L}(\mathcal{B})$.

$$\mathrm{L}(\mathcal{A}) \subseteq \mathrm{L}(\mathcal{S}) \text{ and } \mathrm{L}(\mathcal{B}) \subseteq \mathrm{L}(\mathcal{S})^{c}$$

$$\mathrm{L}(\mathcal{A}) \cap \mathrm{L}(\mathcal{S})^{c} = \emptyset \text{ and } \mathrm{L}(\mathcal{B}) \cap \mathrm{L}(\mathcal{S}) = \emptyset$$

Proof

1. Fix a **positional** (or **f-m**) win. strategy $\boldsymbol{\sigma}$ of Pathfinder in the disjointness game for $\mathcal{A}$, $\mathcal{B}$.

2. Prove that $\boldsymbol{\sigma}$ cannot loop between accepting states of $\mathcal{A}$ and $\mathcal{B}$.

3. Stratify $\mathcal{A} \times \mathcal{B} \times \boldsymbol{\sigma}$ into **layers**.

4. $\rightsquigarrow$ a **weak alternating** automaton $\mathcal{S}$.

■

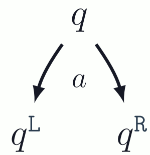
+ extension to higher indices (Arnold, Santocanale '05)

Deterministic and game automata

Deterministic and game automata

Deterministic automata

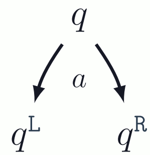
For each $q \in Q$, $a \in A$ there is a **unique** transition



Deterministic and game automata

Deterministic automata

For each $q \in Q$, $a \in A$ there is a **unique** transition

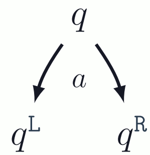


$\rightsquigarrow$ **unique** run on every tree

Deterministic and game automata

Deterministic automata

For each $q \in Q$, $a \in A$ there is a **unique** transition

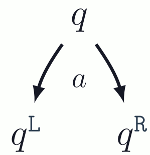


$\rightsquigarrow$ **unique** run on every tree

Acceptance game for $t \overset{?}{\in} \mathrm{L}(\mathcal{A})$

Deterministic automata

For each $q \in Q$, $a \in A$ there is a **unique** transition



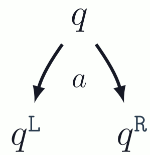
$\rightsquigarrow$ **unique** run on every tree

Acceptance game for $t \overset{?}{\in} \mathrm{L}(\mathcal{A})$

$$\left[\ (q_n, u_n) \rightarrow \qquad\qquad \right]$$

Deterministic automata

For each $q \in Q$, $a \in A$ there is a **unique** transition

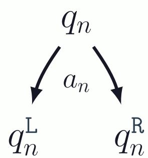


$\rightsquigarrow$ **unique** run on every tree

Acceptance game for $t \overset{?}{\in} \mathrm{L}(\mathcal{A})$

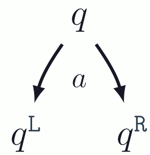
$(q_n, u_n) \to$

TREE: $a_n = t(u_n)$,



Deterministic and game automata

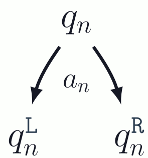
Deterministic automata

For each $q \in Q$, $a \in A$ there is a **unique** transition 

$\rightsquigarrow$ **unique** run on every tree

Acceptance game for $t \overset{?}{\in} \mathrm{L}(\mathcal{A})$

$(q_n, u_n) \rightarrow$

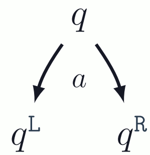
TREE: $a_n = t(u_n)$, 

$\forall$: $d_n \in \{\mathtt{L}, \mathtt{R}\}$

Deterministic and game automata

Deterministic automata

For each $q \in Q$, $a \in A$ there is a **unique** transition

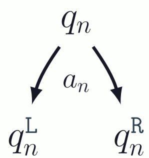


$\rightsquigarrow$ **unique** run on every tree

Acceptance game for $t \stackrel{?}{\in} \mathrm{L}(\mathcal{A})$

$(q_n, u_n) \rightarrow$

Tree: $a_n = t(u_n)$,

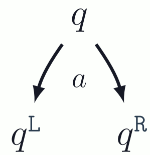


$\forall$: $d_n \in \{\mathtt{L}, \mathtt{R}\}$

$\rightarrow (q_n^{d_n}, u_n d_n)$

Deterministic and game automata

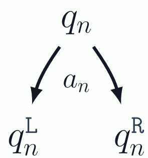
Deterministic automata

For each $q \in Q$, $a \in A$ there is a **unique** transition 

$\rightsquigarrow$ **unique** run on every tree

Acceptance game for $t \overset{?}{\in} \mathrm{L}(\mathcal{A})$

$(q_n, u_n) \to$

TREE: $a_n = t(u_n)$, 

$\forall$: $d_n \in \{\mathtt{L}, \mathtt{R}\}$

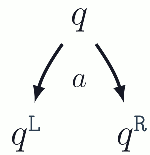
$\to (q_n^{d_n}, u_n d_n)$

TREE wins
iff
$(q_n)_{n\in\omega}$ is accepting

Deterministic and game automata

Deterministic automata

For each $q \in Q$, $a \in A$ there is a **unique** transition

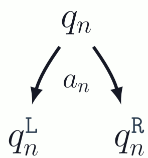


⇝ **unique** run on every tree

Acceptance game for $t \overset{?}{\in} \mathrm{L}(\mathcal{A})$

$(q_n, u_n) \to$

TREE: $a_n = t(u_n)$,



$\forall$: $d_n \in \{\mathtt{L}, \mathtt{R}\}$

$\to (q_n^{d_n}, u_n d_n)$

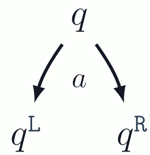
TREE wins
iff
$(q_n)_{n\in\omega}$ is accepting

Deterministic $\equiv$ ***Conjunctive***

Deterministic and game automata

Deterministic automata

For each $q \in Q$, $a \in A$ there is a **unique** transition

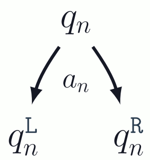


$\rightsquigarrow$ **unique** run on every tree

Acceptance game for $t \overset{?}{\in} \mathrm{L}(\mathcal{A})$

$(q_n, u_n) \to$

TREE: $a_n = t(u_n)$,



$\forall$: $d_n \in \{\mathtt{L}, \mathtt{R}\}$

$\to (q_n^{d_n}, u_n d_n)$

TREE wins
iff
$(q_n)_{n\in\omega}$ is accepting

Deterministic $\equiv$ ***Conjunctive***

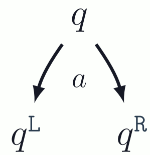
Game automata

As above, but the player $\exists$ / $\forall$ choosing d_n depends on (q_n, a_n).

Deterministic and game automata

Deterministic automata

For each $q \in Q$, $a \in A$ there is a **unique** transition

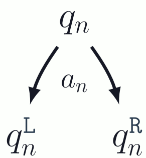


$\rightsquigarrow$ **unique** run on every tree

Acceptance game for $t \overset{?}{\in} \mathrm{L}(\mathcal{A})$

$(q_n, u_n) \to$

TREE: $a_n = t(u_n)$,



$\forall$: $d_n \in \{\mathtt{L}, \mathtt{R}\}$

$\to (q_n^{d_n}, u_n d_n)$

TREE wins
iff
$(q_n)_{n\in\omega}$ is accepting

Deterministic $\equiv$ *Conjunctive*

Game automata

As above, but the player $\exists$ / $\forall$ choosing d_n depends on (q_n, a_n).

Det. $\subsetneq$ Game $\subsetneq$ Non-det.

Our results

Our results

Theorem (Clemente, S. '21)

Given a pair of tree regular languages L_1, L_2 and $C = \{i, \ldots, j\}$,

it is EXPTIME **complete** to decide

whether they can be **separated** by:

Our results

Theorem (Clemente, S. '21)

Given a pair of tree regular languages L_1, L_2 and $C = \{i, \ldots, j\}$,

it is EXPTIME **complete** to decide

whether they can be **separated** by:

1. any **deterministic** parity aut.

Our results

Theorem (Clemente, S. '21)

Given a pair of tree regular languages L_1, L_2 and $C = \{i, \ldots, j\}$,

it is EXPTIME **complete** to decide

whether they can be **separated** by:

SEPARABILITY MEMBERHSHIP

1. any **deterministic** parity aut.

Our results

Theorem (Clemente, S. '21)

Given a pair of tree regular languages L_1, L_2 and $C = \{i, \ldots, j\}$,

it is EXPTIME **complete** to decide

whether they can be **separated** by:

	SEPARABILITY	MEMBERHSHIP
1. any **deterministic** parity aut.	new (?)	

Our results

Theorem (Clemente, S. '21)

Given a pair of tree regular languages L_1, L_2 and $C = \{i, \ldots, j\}$,

it is EXPTime **complete** to decide

whether they can be **separated** by:

	SEPARABILITY	MEMBERHSHIP
1. any **deterministic** parity aut.	new (?)	**[NW '98]**

Our results

Theorem (Clemente, S. '21)

Given a pair of tree regular languages L_1, L_2 and $C = \{i, \ldots, j\}$,

it is EXPTIME **complete** to decide

whether they can be **separated** by:

	SEPARABILITY	MEMBERHSHIP
1. any **deterministic** parity aut.	new (?)	**[NW '98]**
2. a **deterministic** C-parity aut.		

Our results

Theorem (Clemente, S. '21)

Given a pair of tree regular languages L_1, L_2 and $C = \{i, \ldots, j\}$,

it is EXPTIME **complete** to decide

whether they can be **separated** by:

	SEPARABILITY	MEMBERHSHIP
1. any **deterministic** parity aut.	new (?)	**[NW '98]**
2. a **deterministic** C-parity aut.	**new!**	**[NW '98]**

Our results

Theorem (Clemente, S. '21)

Given a pair of tree regular languages L_1, L_2 and $C = \{i, \dots, j\}$,
it is EXPTIME **complete** to decide
whether they can be **separated** by:

	SEPARABILITY	MEMBERHSHIP
1. any **deterministic** parity aut.	new (?)	**[NW '98]**
2. a **deterministic** C-parity aut.	**new!**	**[NW '98]**
3. any **game** parity aut.		

Our results

Theorem (Clemente, S. '21)

Given a pair of tree regular languages L_1, L_2 and $C = \{i, \ldots, j\}$,

it is EXPTIME **complete** to decide

whether they can be **separated** by:

	SEPARABILITY	MEMBERHSHIP
1. any **deterministic** parity aut.	new (?)	**[NW '98]**
2. a **deterministic** C-parity aut.	**new!**	**[NW '98]**
3. any **game** parity aut.	**new!**	**[FMS '13]**

Our results

Theorem (Clemente, S. '21)

Given a pair of tree regular languages L_1, L_2 and $C = \{i, \ldots, j\}$,

it is EXPTIME **complete** to decide

whether they can be **separated** by:

	SEPARABILITY	MEMBERHSHIP
1. any **deterministic** parity aut.	new (?)	**[NW '98]**
2. a **deterministic** C-parity aut.	**new!**	**[NW '98]**
3. any **game** parity aut.	**new!**	**[FMS '13]**
4. a **game** C-parity aut.		

Our results

Theorem (Clemente, S. '21)

Given a pair of tree regular languages L_1, L_2 and $C = \{i, \ldots, j\}$,

it is EXPTIME **complete** to decide

whether they can be **separated** by:

	SEPARABILITY	MEMBERHSHIP
1. any **deterministic** parity aut.	new (?)	**[NW '98]**
2. a **deterministic** C-parity aut.	**new!**	**[NW '98]**
3. any **game** parity aut.	**new!**	**[FMS '13]**
4. a **game** C-parity aut.	**new!**	**new!**