Layered Automata

A canonical model for ω-regular languages

Antonio Casares · RPTU Kaiserslautern

Joint work with: Christof Löding and Igor Walukiewicz

GT Vérif, 7 Novembre 2025

Everybody loves automata

✦ Finite automata $\longleftrightarrow$ MSO-logic *(Büchi, Elgot, Trakhtenbrot '60)*

✦ Finite automata ⟷ MSO-logic *(Büchi, Elgot, Trakhtenbrot '60)*

A minimal, canonical DFA
for each regular language

✦ Finite automata ⟷ MSO-logic *(Büchi, Elgot, Trakhtenbrot '60)*

Beautiful mathematical theory!

A minimal, canonical DFA
for each regular language

Efficiency in applications!

- Finite automata $\longleftrightarrow$ MSO-logic *(Büchi, Elgot, Trakhtenbrot '60)*

Beautiful mathematical theory!

A minimal, canonical DFA
for each regular language

Efficiency in applications!

- Minimisation in $O(n \log n)$ *(Hopcroft '71)*

- Finite automata $\longleftrightarrow$ MSO-logic *(Büchi, Elgot, Trakhtenbrot '60)*

Beautiful mathematical theory!

A minimal, canonical DFA
for each regular language

Efficiency in applications!

- Minimisation in $O(n \log n)$ *(Hopcroft '71)*
- Automata learning in PTIME *(Gold '67, Angluin 87')*

✦ Finite automata ⟷ MSO-logic *(Büchi, Elgot, Trakhtenbrot '60)*

Beautiful mathematical theory!

A minimal, canonical DFA
for each regular language

Efficiency in applications!

- ✦ Minimisation in $O(n \log n)$ *(Hopcroft '71)*
- ✦ Automata learning in PTIME *(Gold '67, Angluin 87')*
- ✦ Simple characterization of FO-definability

⋮

Everybody loves automata

- Finite automata ⟷ MSO-logic *(Büchi, Elgot, Trakhtenbrot '60)*

Beautiful mathematical theory!

A minimal, canonical DFA for each regular language

Efficiency in applications!

- Minimisation in $O(n \log n)$ *(Hopcroft '71)*
- Automata learning in PTIME *(Gold '67, Angluin 87')*
- Simple characterization of FO-definability

⋮

Everybody loves automata

- Finite automata ⟷ MSO-logic *(Büchi, Elgot, Trakhtenbrot '60)*

Beautiful mathematical theory!

A minimal, canonical DFA for each regular language

Efficiency in applications!

- Minimisation in $O(n \log n)$ *(Hopcroft '71)*
- Automata learning in PTIME *(Gold '67, Angluin 87')*
- Simple characterization of FO-definability

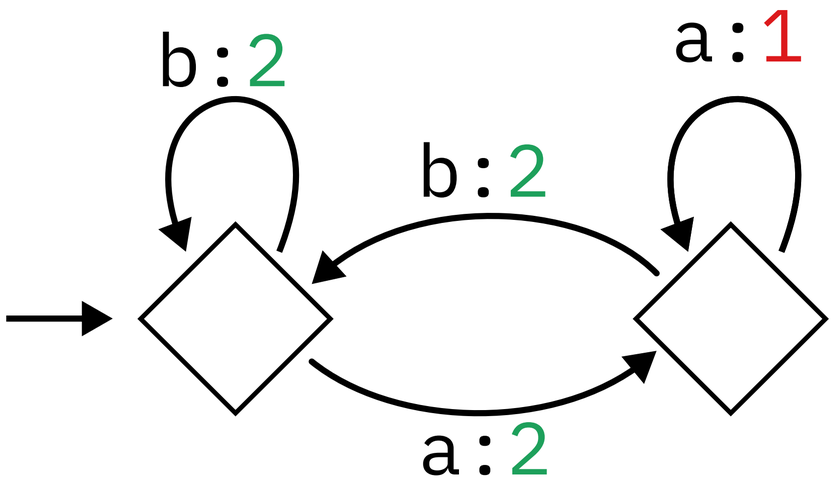
⋮

Let's generalize to infinite words!

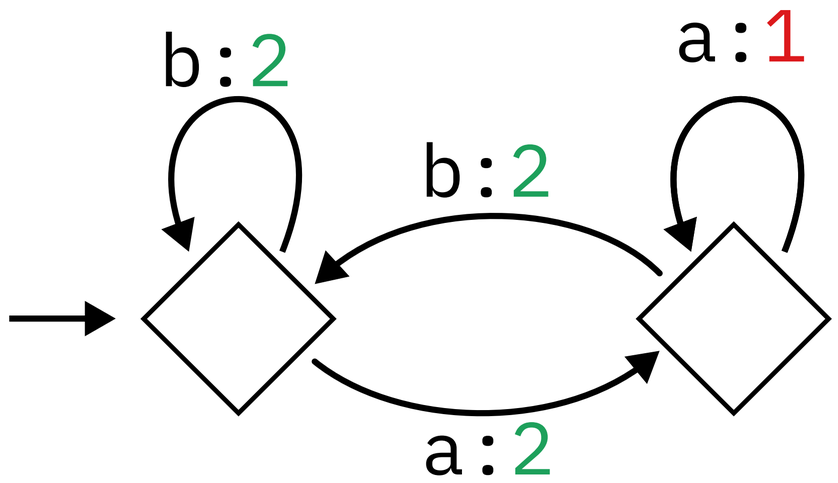
ω-automata $\longleftrightarrow$ MSO-logic *(Büchi '62)*

- ω-automata $\longleftrightarrow$ MSO-logic *(Büchi '62)*
- Applications in verification and reactive synthesis

Parity automata

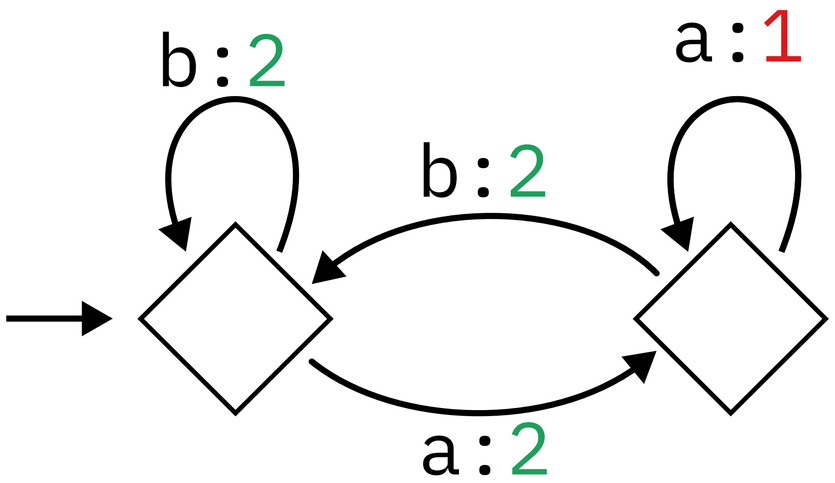


Parity automata



Input: $abaababbaa\ldots \in \Sigma^{\omega}$

Parity automata

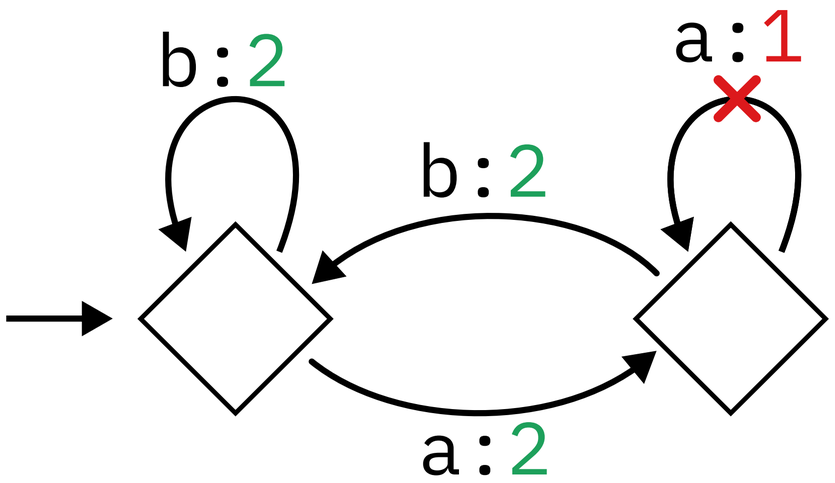


Run accepting if

min {*priorities seen infinitely often*}
is even

Input: abaababbaa... $\in \Sigma^{\omega}$

Parity automata

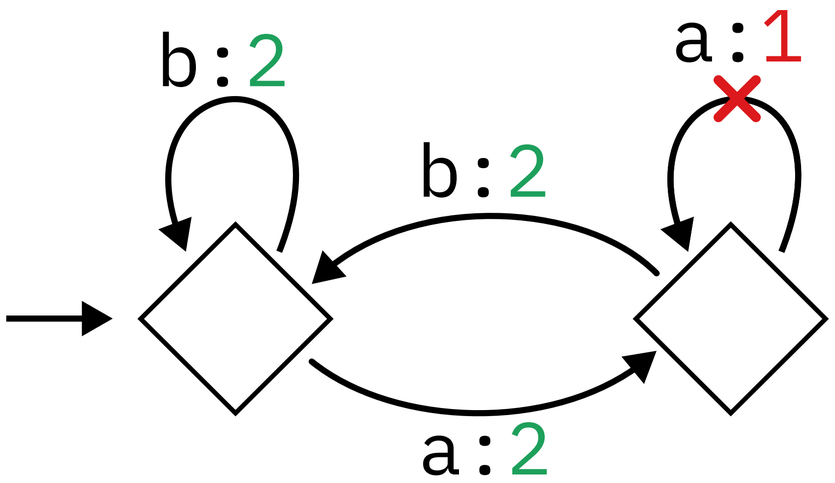


Run accepting if

min {*priorities seen infinitely often*}
is even

Input: abaababbaa... $\in \Sigma^{\omega}$

Parity automata



Run accepting if

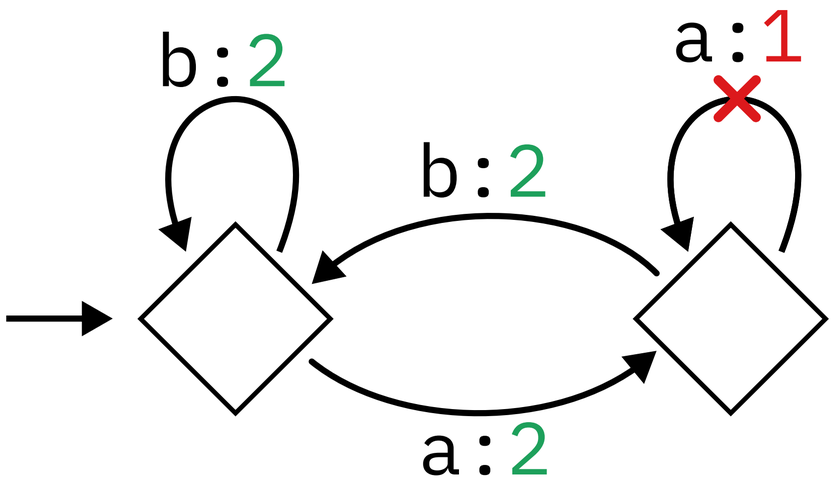
min {*priorities seen infinitely often*}
is even

Input: `abaababbaa...` $\in \Sigma^{\omega}$

$L(A)$ = `Finitely Often` `` `aa` ``

Parity automata

Acceptance on transitions!



Run accepting if

min {*priorities seen infinitely often*}
is even

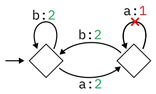
Input: `abaababbaa...` $\in \Sigma^{\omega}$

$L(A)$ = `Finitely Often` `` `aa` ``

- ω-automata ⟷ MSO-logic *(Büchi '62)*
- Applications in verification and reactive synthesis

Parity automata

Acceptance on transitions!



Run accepting if
min {*priorities seen infinitely often*}
is even

Input: `abaababbaa...` $\in \Sigma^\omega$

L(A) = `Finitely Often 'aa'`

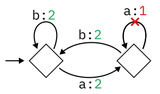
ω-automata ⟷ MSO-logic *(Büchi '62)*

Applications in verification and reactive synthesis

No unique minimal deterministic automata

Parity automata

Acceptance on transitions!

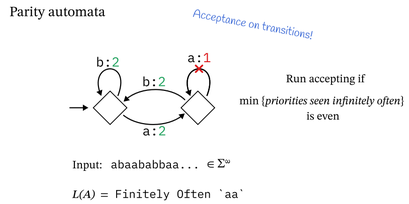


Run accepting if
min {*priorities seen infinitely often*}
is even

Input: `abaababbaa...` $\in \Sigma^\omega$

$L(A)$ = `Finitely Often 'aa'`

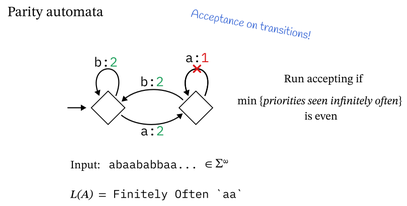
- ω-automata ⟷ MSO-logic *(Büchi '62)*
- Applications in verification and reactive synthesis



No unique minimal deterministic automata

Use of ω-automata very costly in applications!

ω-automata ⟷ MSO-logic *(Büchi '62)*



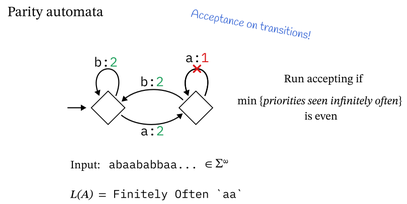
Applications in verification and reactive synthesis

No unique minimal deterministic automata

Use of ω-automata very costly in applications!

Minimisation is NP-complete *(Schewe '10, AbuRadi-Ehlers '25)*

✦ ω-automata ⟷ MSO-logic *(Büchi '62)*



✦ Applications in verification and reactive synthesis

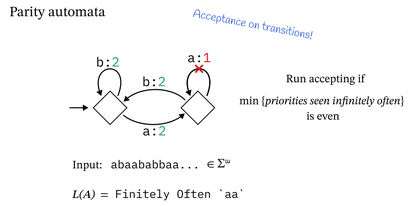
No unique minimal deterministic automata

Use of ω-automata very costly in applications!

✘ Minimisation is NP-complete *(Schewe '10, AbuRadi-Ehlers '25)*

✘ Costly determinization and boolean operations

✦ ω-automata ⟷ MSO-logic *(Büchi '62)*



✦ Applications in verification and reactive synthesis

No unique minimal deterministic automata

Use of ω-automata very costly in applications!

✘ Minimisation is NP-complete *(Schewe '10, AbuRadi-Ehlers '25)*

✘ Costly determinization and boolean operations

✘ No efficient learning of ω-automata

Everybody loves automata

- Finite automata ⟷ MSO-logic *(Büchi, Elgot, Trakhtenbrot '60)*

Beautiful mathematical theory!

A minimal, canonical DFA for each regular language

Efficiency in applications!

- Minimisation in $O(n \log n)$ *(Hopcroft '71)*
- Automata learning in PTIME *(Gold '67, Angluin 87')*
- Simple characterization of FO-definability

⋮

Let's generalize to infinite words!

- ω-automata ⟷ MSO-logic *(Büchi '62)*
- Applications in verification and reactive synthesis

No unique minimal deterministic automata

Use of ω-automata very costly in applications!

- ✗ Minimisation is NP-complete *(Schewe '10, AbuRadi-Ehlers '25)*
- ✗ Costly determinization and boolean operations
- ✗ No efficient learning of ω-automata

Everybody loves automata

- Finite automata ⟷ MSO-logic *(Büchi, Elgot, Trakhtenbrot '60)*

Beautiful mathematical theory!

A minimal, canonical DFA for each regular language

Efficiency in applications!

- Minimisation in $O(n \log n)$ *(Hopcroft '71)*
- Automata learning in PTIME *(Gold '67, Angluin 87')*
- Simple characterization of FO-definability

⋮

Let's generalize to infinite words!

- ω-automata ⟷ MSO-logic *(Büchi '62)*

- Applications in verification and reactive synthesis

No unique minimal deterministic automata

Use of ω-automata very costly in applications!

- ✗ Minimisation is NP-complete *(Schewe '10, AbuRadi-Ehlers '25)*
- ✗ Costly determinization and boolean operations
- ✗ No efficient learning of ω-automata

The theory of deterministic parity automata is not satisfactory...

Everybody loves automata

- Finite automata ⟷ MSO-logic *(Büchi, Elgot, Trakhtenbrot '60)*

Beautiful mathematical theory!

A minimal, canonical DFA for each regular language

Efficiency in applications!

- Minimisation in $O(n \log n)$ *(Hopcroft '71)*
- Automata learning in PTIME *(Gold '67, Angluin 87')*
- Simple characterization of FO-definability

⋮

Let's generalize to infinite words!

- ω-automata ⟷ MSO-logic *(Büchi '62)*

- Applications in verification and reactive synthesis

No unique minimal deterministic automata

Use of ω-automata very costly in applications!

- ✗ Minimisation is NP-complete *(Schewe '10, AbuRadi-Ehlers '25)*
- ✗ Costly determinization and boolean operations
- ✗ No efficient learning of ω-automata

The theory of deterministic parity automata is not satisfactory...

Instead, use ***Layered Automata***

Layered Automata

Formalism to represent a subclass of alternating ω-automata

Layered Automata

Formalism to represent a subclass of alternating ω-automata

Non deterministic: w is accepted if ***some*** run is accepting.

Non deterministic: w is accepted if ***some*** run is accepting.

An existential player tries to build an accepting run.

Non deterministic: w is accepted if ***some*** run is accepting.

An existential player tries to build an accepting run.

Universal: w is accepted if ***all*** run are accepting

Non deterministic: w is accepted if ***some*** run is accepting.

An existential player tries to build an accepting run.

Universal: w is accepted if ***all*** run are accepting $=$ w is rejected if ***some*** run is rejecting.

Non deterministic: w is accepted if ***some*** run is accepting.

An existential player tries to build an accepting run.

Universal: w is accepted if ***all*** run are accepting $=$ w is rejected if ***some*** run is rejecting.

A universal player tries to build a rejecting run.

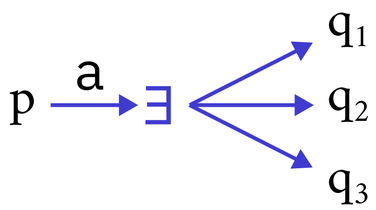
Non deterministic: w is accepted if ***some*** run is accepting.

An existential player tries to build an accepting run.

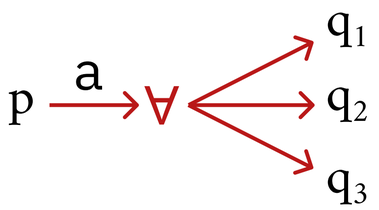
Universal: w is accepted if ***all*** run are accepting $=$ w is rejected if ***some*** run is rejecting.

A universal player tries to build a rejecting run.

Alternating automata:



existential choices



universal choices

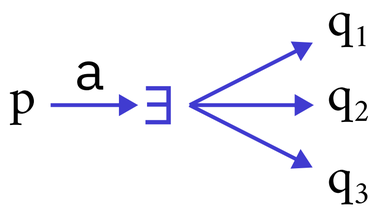
Non deterministic: w is accepted if ***some*** run is accepting.

An existential player tries to build an accepting run.

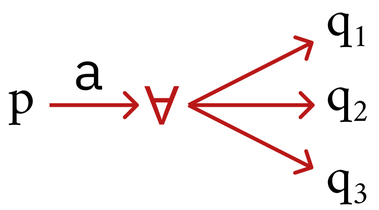
Universal: w is accepted if ***all*** run are accepting $=$ w is rejected if ***some*** run is rejecting.

A universal player tries to build a rejecting run.

Alternating automata:



existential choices



universal choices

A word w induces a parity game. $w \in L(\mathcal{A})$ if the existential player wins this game.

Layered Automata

Formalism to represent a subclass of alternating ω-automata

Non deterministic: w is accepted if ***some*** run is accepting.

An existential player tries to build an accepting run.

Universal: w is accepted if ***all*** run are accepting $\quad = \quad$ w is rejected if ***some*** run is rejecting.

A universal player tries to build a rejecting run.

Alternating automata: $p \xrightarrow{a} \exists \{q_1, q_2, q_3\}$ $\qquad$ $p \xrightarrow{a} \forall \{q_1, q_2, q_3\}$

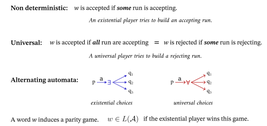
existential choices $\qquad$ *universal choices*

A word w induces a parity game. $\quad w \in L(\mathcal{A})$ if the existential player wins this game.

Layered Automata

Formalism to represent a subclass of alternating ω-automata

Alternating automata are too complex to use in practice 😢



Layered Automata

well-behaved

Formalism to represent a subclass of alternating ω-automata

Some subclasses are usable in verification!! 🥳

Non deterministic: w is accepted if ***some*** run is accepting.

An existential player tries to build an accepting run.

Universal: w is accepted if ***all*** run are accepting $=$ w is rejected if ***some*** run is rejecting.

A universal player tries to build a rejecting run.

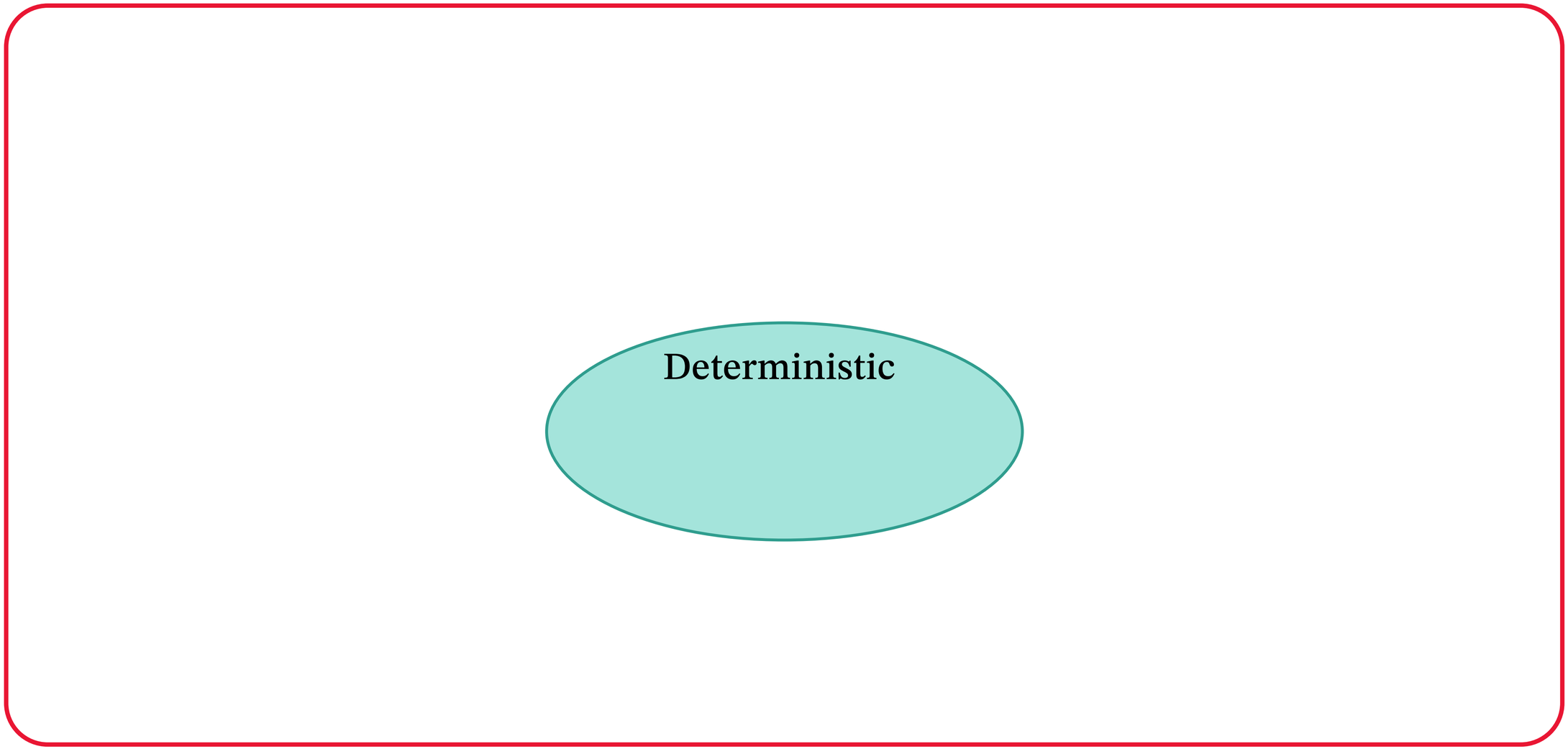
Alternating automata:

existential choices *universal choices*

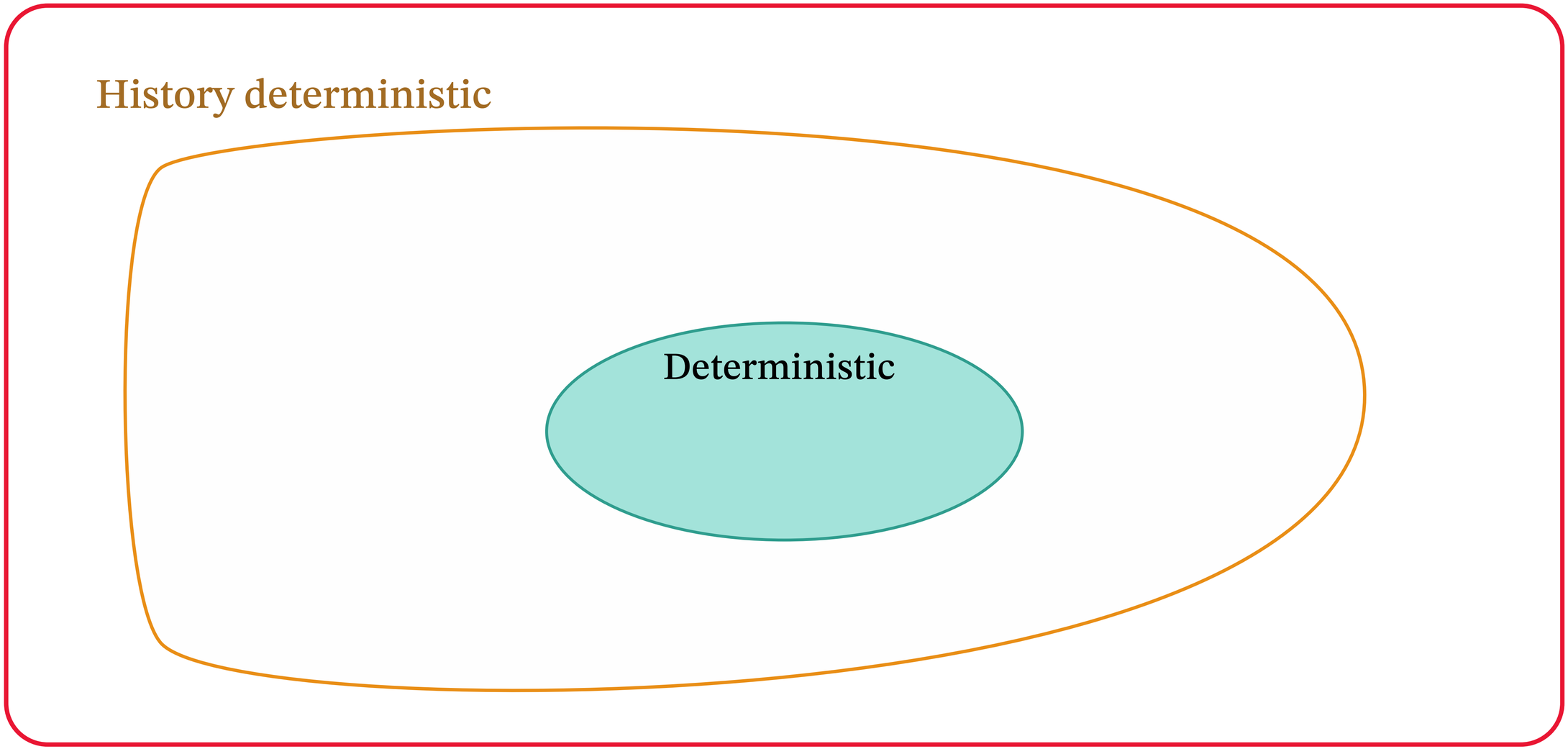
A word w induces a parity game. $w \in L(\mathcal{A})$ if the existential player wins this game.

Alternating parity automata

Alternating parity automata



Alternating parity automata



Non-deterministic case (*For alternating automata* $\rightarrow$ *natural generalization*)

Non-deterministic case (*For alternating automata* $\rightarrow$ *natural generalization*)

An automaton is ***history-deterministic*** if its non-determinism can be resolved on-the-fly.

Non-deterministic case (*For alternating automata → natural generalization*)

An automaton is ***history-deterministic*** if its non-determinism can be resolved on-the-fly.

= If Eve has a winning strategy in the ***letter game***:

Adam gives letters:

Eve takes transitions:

Non-deterministic case (*For alternating automata → natural generalization*)

An automaton is ***history-deterministic*** if its non-determinism can be resolved on-the-fly.

= If Eve has a winning strategy in the ***letter game***:

Adam gives letters: a

Eve takes transitions:

Non-deterministic case (*For alternating automata* $\rightarrow$ *natural generalization*)

An automaton is ***history-deterministic*** if its non-determinism can be resolved on-the-fly.

$=$ If Eve has a winning strategy in the ***letter game***:

Adam gives letters: a

Eve takes transitions: $p_0 \xrightarrow{a} p_1$

Non-deterministic case (*For alternating automata → natural generalization*)

An automaton is ***history-deterministic*** if its non-determinism can be resolved on-the-fly.

= If Eve has a winning strategy in the ***letter game***:

Adam gives letters: a b

Eve takes transitions: $p_0 \xrightarrow{a} p_1$

Non-deterministic case (*For alternating automata → natural generalization*)

An automaton is ***history-deterministic*** if its non-determinism can be resolved on-the-fly.

= If Eve has a winning strategy in the ***letter game***:

Adam gives letters: a b

Eve takes transitions: $p_0 \xrightarrow{a} p_1$ $p_1 \xrightarrow{b} p_2$

Non-deterministic case (*For alternating automata → natural generalization*)

An automaton is ***history-deterministic*** if its non-determinism can be resolved on-the-fly.

= If Eve has a winning strategy in the ***letter game***:

Adam gives letters: a b b a . . . $w \in \Sigma^\omega$

Eve takes transitions: $p_0 \xrightarrow{a} p_1 \quad p_1 \xrightarrow{b} p_2 \quad p_2 \xrightarrow{b} p_3$ . . . ρ run in the automaton

Non-deterministic case (*For alternating automata → natural generalization*)

An automaton is ***history-deterministic*** if its non-determinism can be resolved on-the-fly.

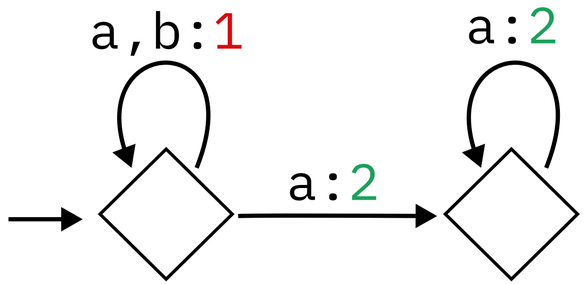
= If Eve has a winning strategy in the ***letter game***:

Adam gives letters: a b b a ... $w \in \Sigma^\omega$

Eve takes transitions: $p_0 \xrightarrow{a} p_1 \quad p_1 \xrightarrow{b} p_2 \quad p_2 \xrightarrow{b} p_3$ ... ρ run in the automaton

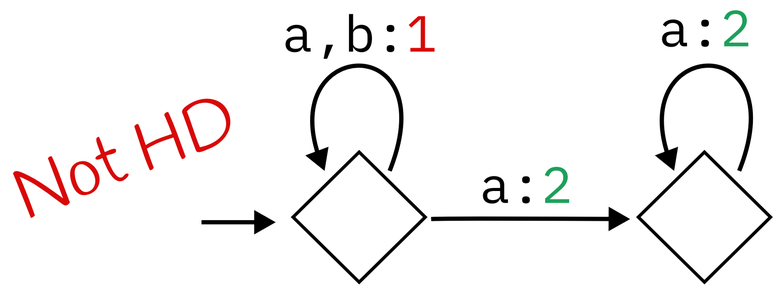
Eve wins if $w \in L(\mathcal{A}) \implies$ ρ is accepting

Examples



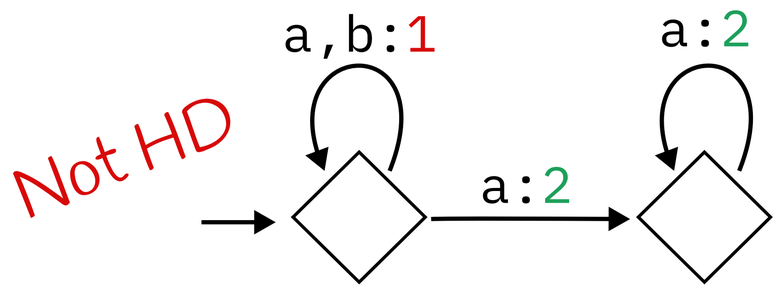
$L(A) = \texttt{Fin(b)}$

Examples

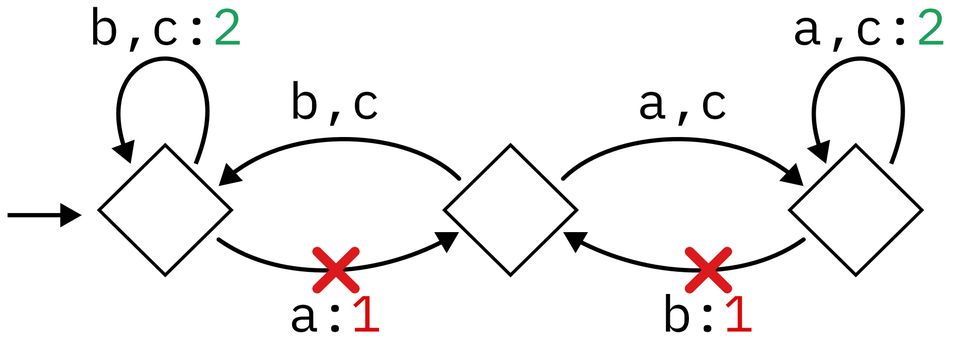


$L(A) = \texttt{Fin(b)}$

Examples

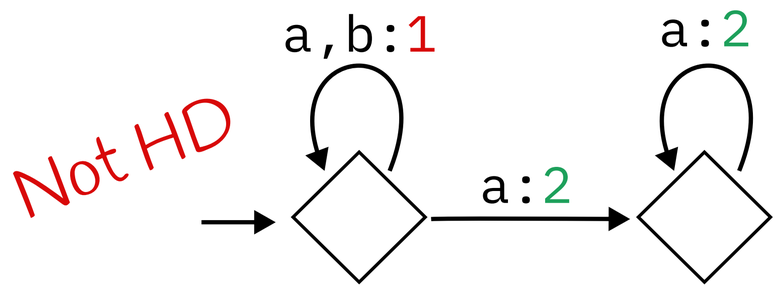


$L(A)$ = Fin(b)

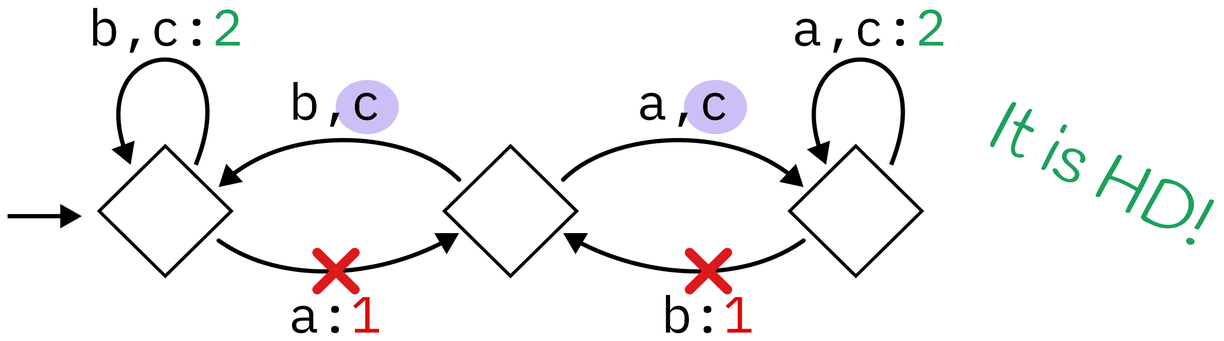


$L(A)$ = Fin(a) or Fin(b)

Examples



$L(A) = \texttt{Fin(b)}$



$L(A) = \texttt{Fin(a) or Fin(b)}$

Non-deterministic case (*For alternating automata* $\longrightarrow$ *natural generalization*)

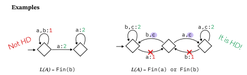
An automaton is ***history-deterministic*** if its non-determinism can be resolved on-the-fly.

$=$ If Eve has a winning strategy in the ***letter game***:

Adam gives letters: a b b a ... $w \in \Sigma^\omega$

Eve takes transitions: $p_0 \xrightarrow{a} p_1 \quad p_1 \xrightarrow{b} p_2 \quad p_2 \xrightarrow{b} p_3$ ... ρ run in the automaton

Eve wins if $w \in L(\mathcal{A}) \implies$ ρ is accepting



Non-deterministic case (*For alternating automata → natural generalization*)

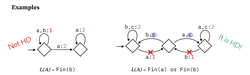
An automaton is ***history-deterministic*** if its non-determinism can be resolved on-the-fly.

= If Eve has a winning strategy in the ***letter game***:

Adam gives letters: a b b a . . . $w \in \Sigma^\omega$

Eve takes transitions: $p_0 \xrightarrow{a} p_1$ $p_1 \xrightarrow{b} p_2$ $p_2 \xrightarrow{b} p_3$ . . . ρ run in the automaton

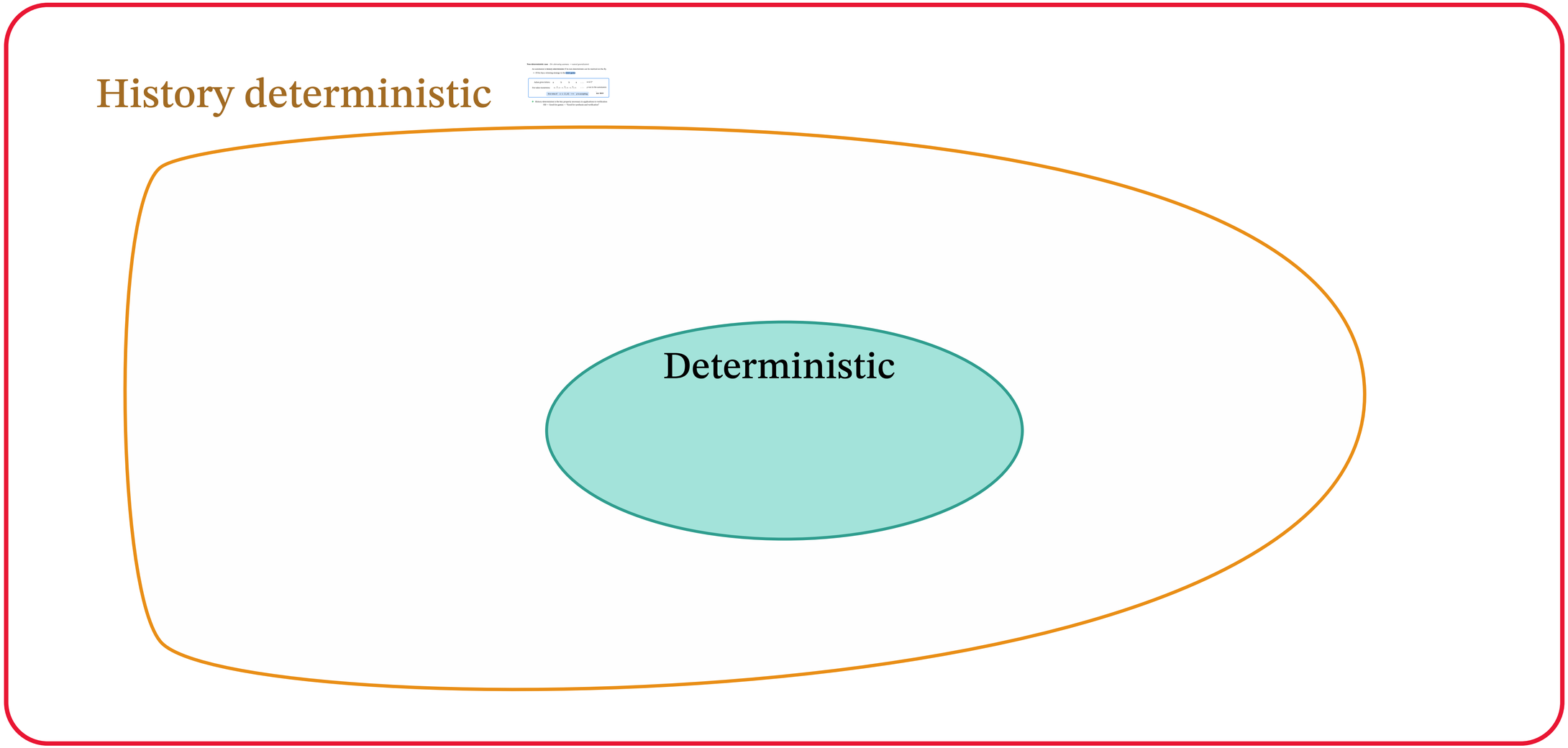
Eve wins if $w \in L(\mathcal{A}) \implies$ ρ is accepting



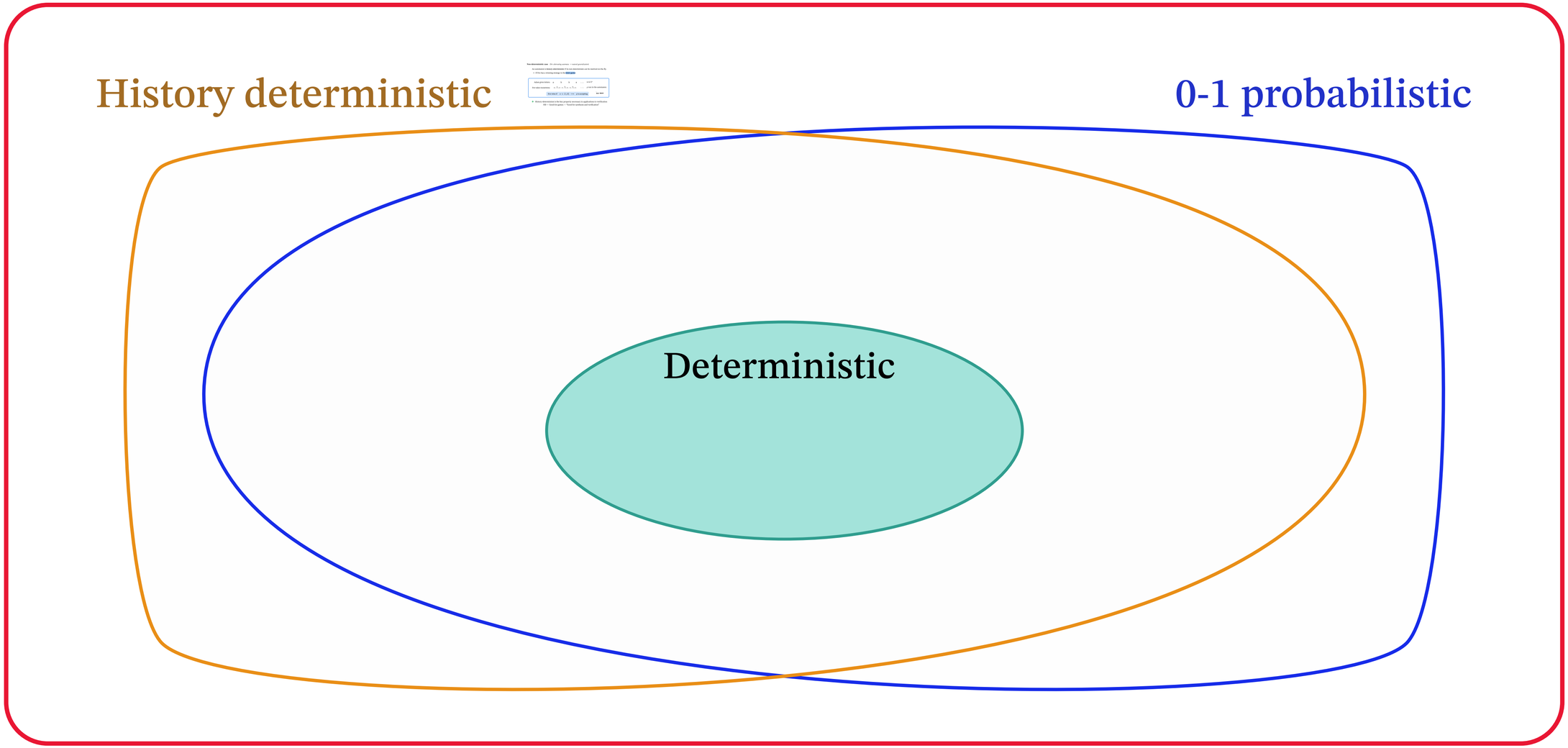
✦ History-determinism is the key property necessary in applications to verification

HD = Good-for-games = “Good for synthesis and verification”

Alternating parity automata



Alternating parity automata



An automaton is ***0-1 probabilistic*** if for every *w,* and a random run ρ over *w:*

- $Pr(\rho \text{ is accepting}) = 1 \iff w \in L(\mathcal{A})$
- $Pr(\rho \text{ is accepting}) = 0 \iff w \notin L(\mathcal{A})$

see the automaton as a graph

An automaton is ***0-1 probabilistic*** if for every *w,* and a random run ρ over *w:*

- $Pr(\rho \text{ is accepting}) = 1 \iff w \in L(\mathcal{A})$
- $Pr(\rho \text{ is accepting}) = 0 \iff w \notin L(\mathcal{A})$

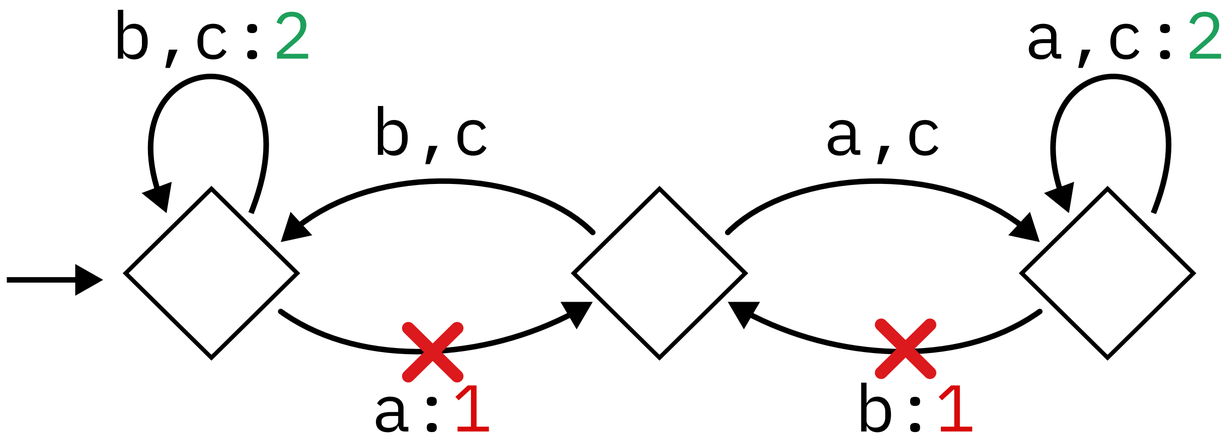
see the automaton as a graph

An automaton is ***0-1 probabilistic*** if for every *w*, and a random run ρ over *w*:

- $Pr(\rho \text{ is accepting}) = 1 \iff w \in L(\mathcal{A})$
- $Pr(\rho \text{ is accepting}) = 0 \iff w \notin L(\mathcal{A})$

The players do not need to think!

Example:



$L(A) = \texttt{Fin(a) or Fin(b)}$

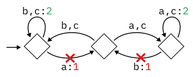
see the automaton as a graph

An automaton is ***0-1 probabilistic*** if for every *w*, and a random run ρ over *w*:

- $Pr(\rho \text{ is accepting}) = 1 \iff w \in L(\mathcal{A})$
- $Pr(\rho \text{ is accepting}) = 0 \iff w \notin L(\mathcal{A})$

The players do not need to think!

Example:



L(A) = Fin(a) or Fin(b)

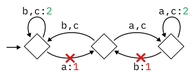
see the automaton as a graph

An automaton is ***0-1 probabilistic*** if for every *w*, and a random run ρ over *w*:

- $Pr(\rho \text{ is accepting}) = 1 \iff w \in L(\mathcal{A})$
- $Pr(\rho \text{ is accepting}) = 0 \iff w \notin L(\mathcal{A})$

The players do not need to think!

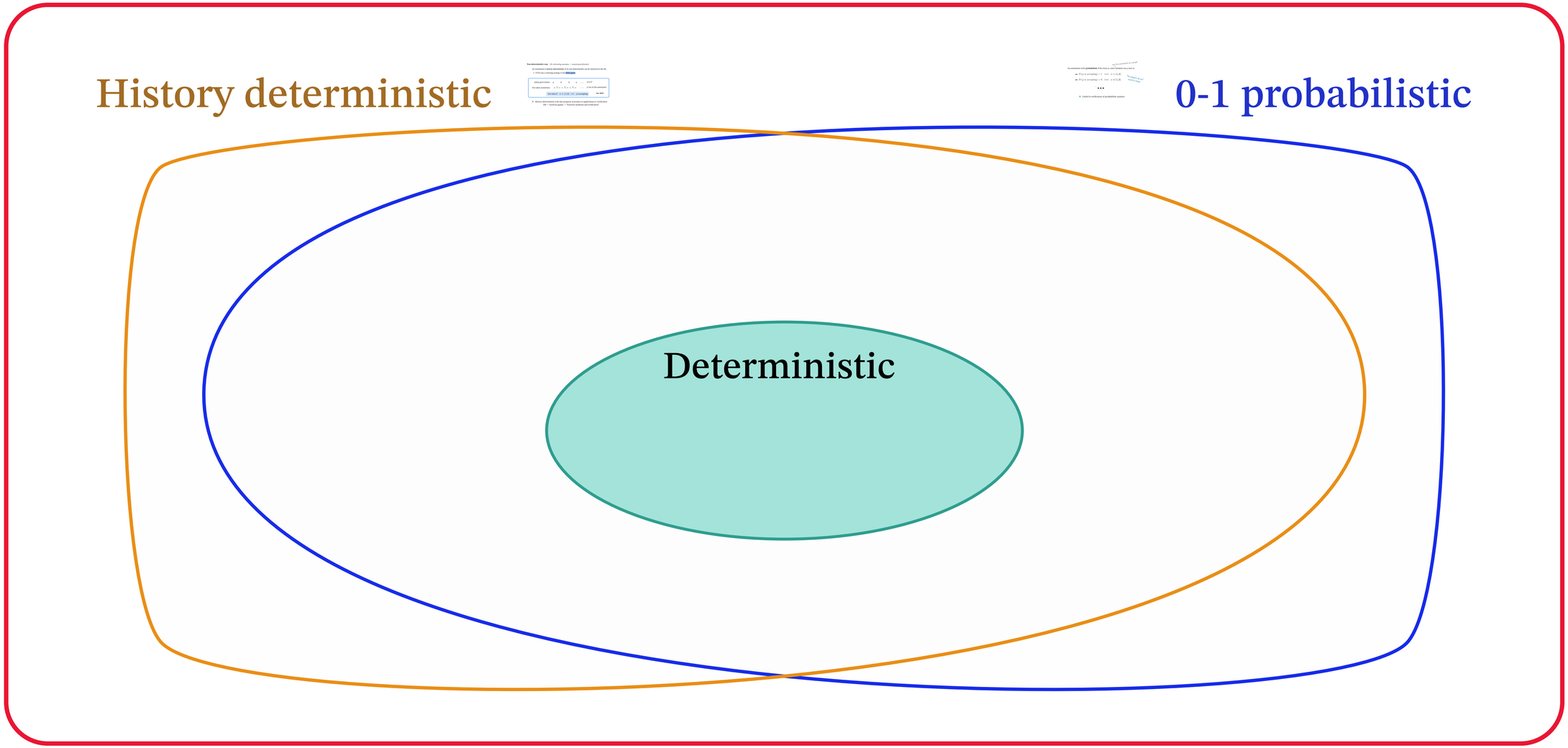
Example:



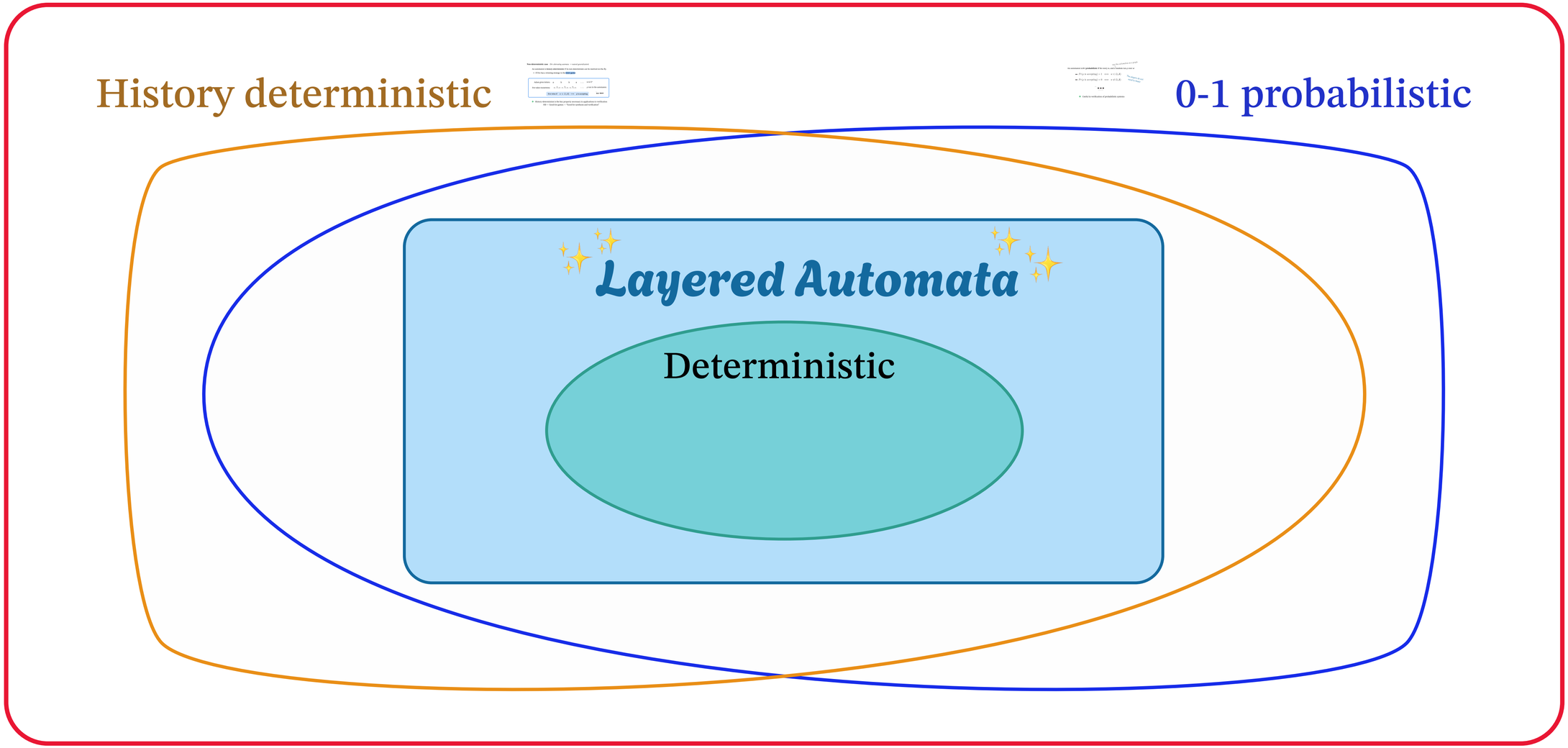
$L(A) =$ Fin(a) or Fin(b)

✦ Useful in verification of probabilistic systems

Alternating parity automata



Alternating parity automata

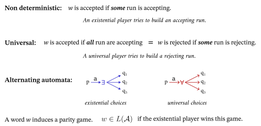


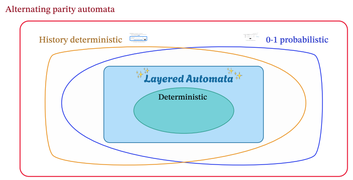
Layered Automata

well-behaved

Formalism to represent a subclass of alternating ω-automata

Some subclasses are usable in verification!!

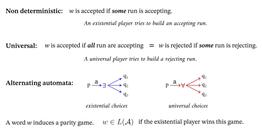




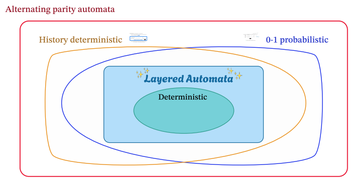
Layered Automata

well-behaved

Formalism to represent a subclass of alternating ω-automata



Some subclasses are usable in verification!!



Definition

Canonicity properties

Layered automaton: A sequence of deterministic transition systems with morphisms

$$\mathcal{T}_1 \xleftarrow{\mu_1} \mathcal{T}_2 \xleftarrow{\mu_2} \quad \dots \quad \mathcal{T}_{d-1} \xleftarrow{\mu_{d-1}} \mathcal{T}_d$$

ms with morphisms

$$\mu : V \to V'$$

$$\mathrm{p} \xrightarrow{\mathtt{a}} \mathrm{q} \quad \text{in } \mathcal{T} \qquad \Longrightarrow \qquad \mu(\mathrm{p}) \xrightarrow{\mathtt{a}} \mu(\mathrm{q}) \quad \text{in } \mathcal{T}'$$

Layered automaton: A sequence of deterministic transition systems with morphisms

$$\mathcal{T}_1 \xleftarrow{\mu_1} \mathcal{T}_2 \xleftarrow{\mu_2} \quad \dots \quad \mathcal{T}_{d-1} \xleftarrow{\mu_{d-1}} \mathcal{T}_d$$

$\mu : V \to V'$

$p \xrightarrow{a} q$ in $\mathcal{T}$ $\implies$ $\mu(p) \xrightarrow{a} \mu(q)$ in $\mathcal{T}'$

incomplete

Layered automaton: A sequence of deterministic transition systems with morphisms

$$\mathcal{T}_1 \xleftarrow{\mu_1} \mathcal{T}_2 \xleftarrow{\mu_2} \quad \dots \quad \mathcal{T}_{d-1} \xleftarrow{\mu_{d-1}} \mathcal{T}_d$$

$\mu : V \to V'$

$p \xrightarrow{a} q$ in $\mathcal{T}$ $\implies$ $\mu(p) \xrightarrow{a} \mu(q)$ in $\mathcal{T}'$

incomplete

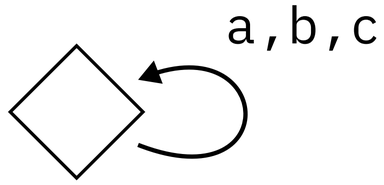
Layered automaton: A sequence of deterministic transition systems with morphisms

$$\mathcal{T}_1 \xleftarrow{\mu_1} \mathcal{T}_2 \xleftarrow{\mu_2} \quad \dots \quad \mathcal{T}_{d-1} \xleftarrow{\mu_{d-1}} \mathcal{T}_d$$

$\mu : V \to V'$

$p \xrightarrow{a} q$ in $\mathcal{T}$ $\implies$ $\mu(p) \xrightarrow{a} \mu(q)$ in $\mathcal{T}'$

$\mathcal{T}_1$



incomplete

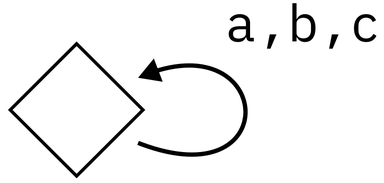
Layered automaton: A sequence of deterministic transition systems with morphisms

$$\mathcal{T}_1 \xleftarrow{\mu_1} \mathcal{T}_2 \xleftarrow{\mu_2} \quad \dots \quad \mathcal{T}_{d-1} \xleftarrow{\mu_{d-1}} \mathcal{T}_d$$

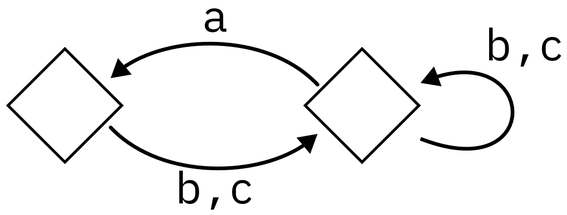
$\mu : V \to V'$

$p \xrightarrow{a} q$ in $\mathcal{T}$ $\implies$ $\mu(p) \xrightarrow{a} \mu(q)$ in $\mathcal{T}'$

$\mathcal{T}_1$



$\mathcal{T}_2$



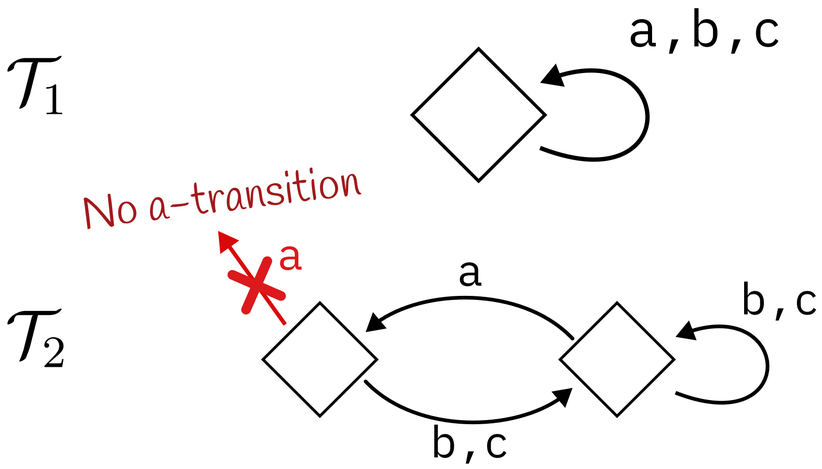
incomplete

Layered automaton: A sequence of deterministic transition systems with morphisms

$$\mathcal{T}_1 \xleftarrow{\mu_1} \mathcal{T}_2 \xleftarrow{\mu_2} \quad \dots \quad \mathcal{T}_{d-1} \xleftarrow{\mu_{d-1}} \mathcal{T}_d$$

$\mu : V \to V'$

$p \xrightarrow{a} q$ in $\mathcal{T}$ $\implies$ $\mu(p) \xrightarrow{a} \mu(q)$ in $\mathcal{T}'$



incomplete

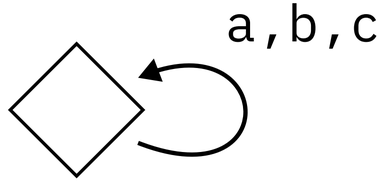
Layered automaton: A sequence of deterministic transition systems with morphisms

$$\mathcal{T}_1 \xleftarrow{\mu_1} \mathcal{T}_2 \xleftarrow{\mu_2} \quad \dots \quad \mathcal{T}_{d-1} \xleftarrow{\mu_{d-1}} \mathcal{T}_d$$

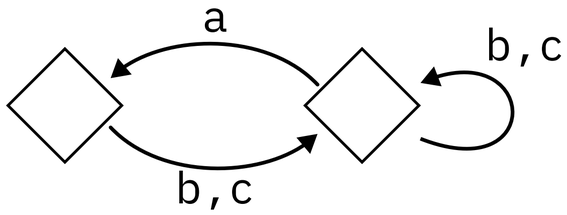
$\mu : V \to V'$

$p \xrightarrow{a} q$ in $\mathcal{T}$ $\implies$ $\mu(p) \xrightarrow{a} \mu(q)$ in $\mathcal{T}'$

$\mathcal{T}_1$



$\mathcal{T}_2$



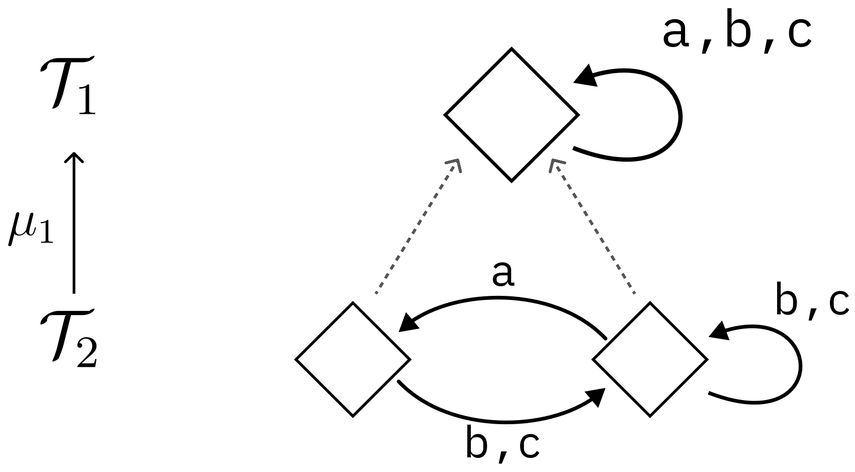
incomplete

Layered automaton: A sequence of deterministic transition systems with morphisms

$$\mathcal{T}_1 \xleftarrow{\mu_1} \mathcal{T}_2 \xleftarrow{\mu_2} \quad \dots \quad \mathcal{T}_{d-1} \xleftarrow{\mu_{d-1}} \mathcal{T}_d$$

$\mu : V \to V'$

$p \xrightarrow{a} q$ in $\mathcal{T}$ $\implies$ $\mu(p) \xrightarrow{a} \mu(q)$ in $\mathcal{T}'$



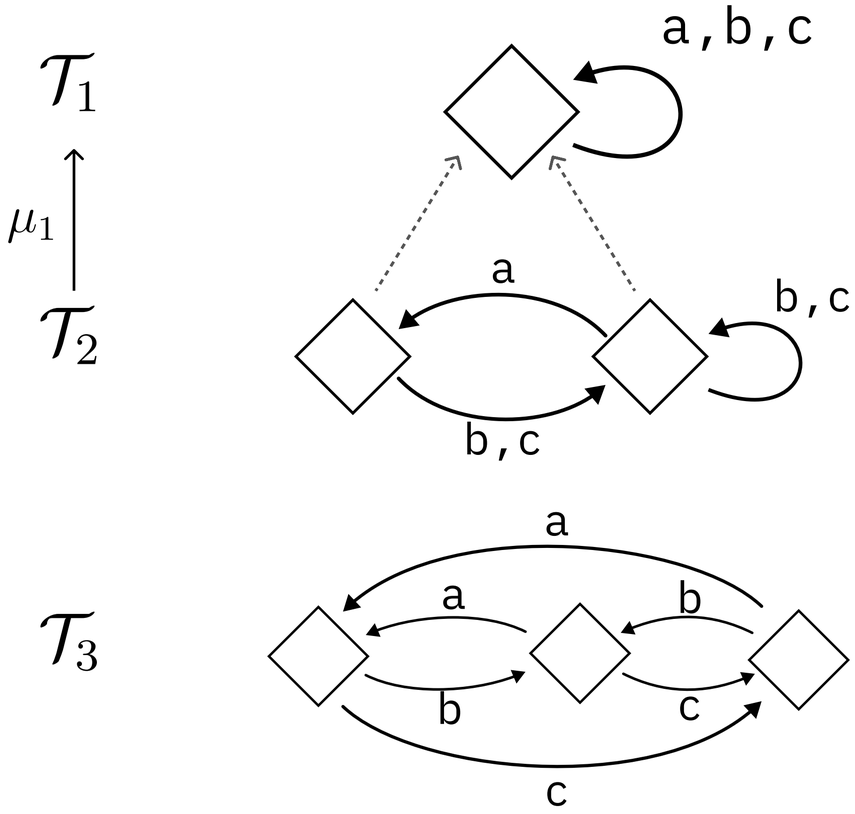
incomplete

Layered automaton: A sequence of deterministic transition systems with morphisms

$$\mathcal{T}_1 \xleftarrow{\mu_1} \mathcal{T}_2 \xleftarrow{\mu_2} \quad \dots \quad \mathcal{T}_{d-1} \xleftarrow{\mu_{d-1}} \mathcal{T}_d$$

$\mu : V \to V'$

$p \xrightarrow{a} q$ in $\mathcal{T}$ $\implies$ $\mu(p) \xrightarrow{a} \mu(q)$ in $\mathcal{T}'$



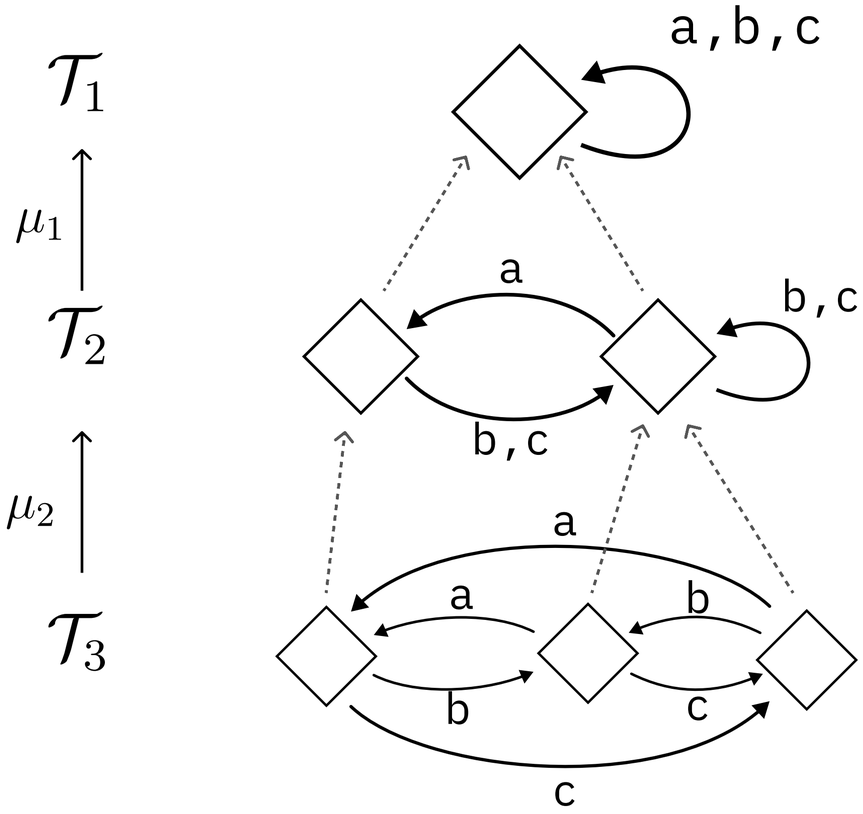
incomplete

Layered automaton: A sequence of deterministic transition systems with morphisms

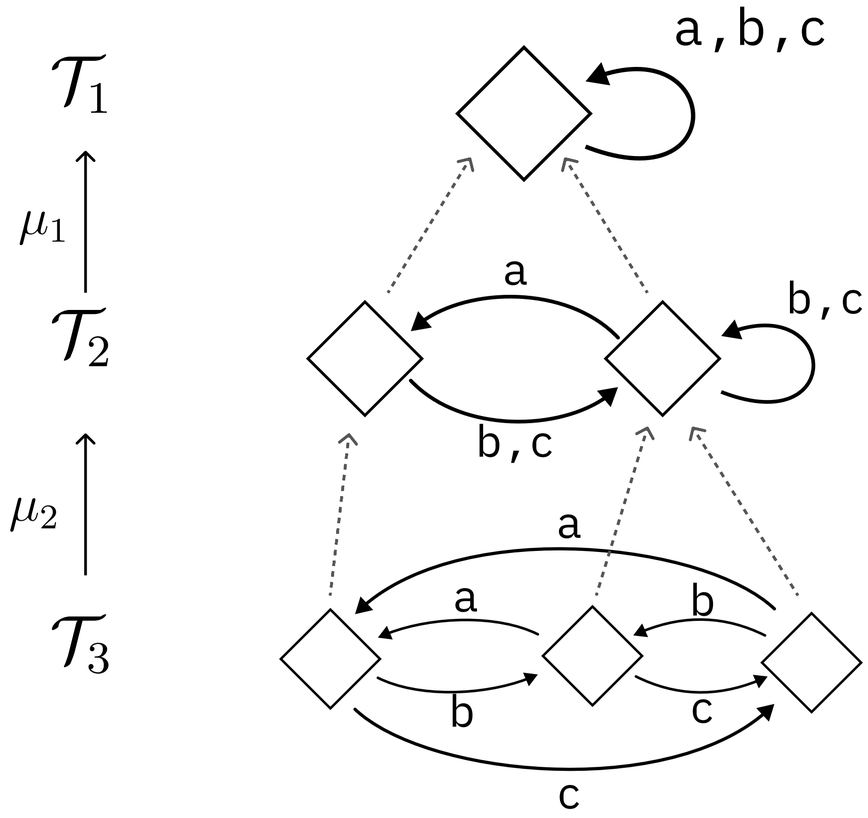
$$\mathcal{T}_1 \xleftarrow{\mu_1} \mathcal{T}_2 \xleftarrow{\mu_2} \quad \dots \quad \mathcal{T}_{d-1} \xleftarrow{\mu_{d-1}} \mathcal{T}_d$$

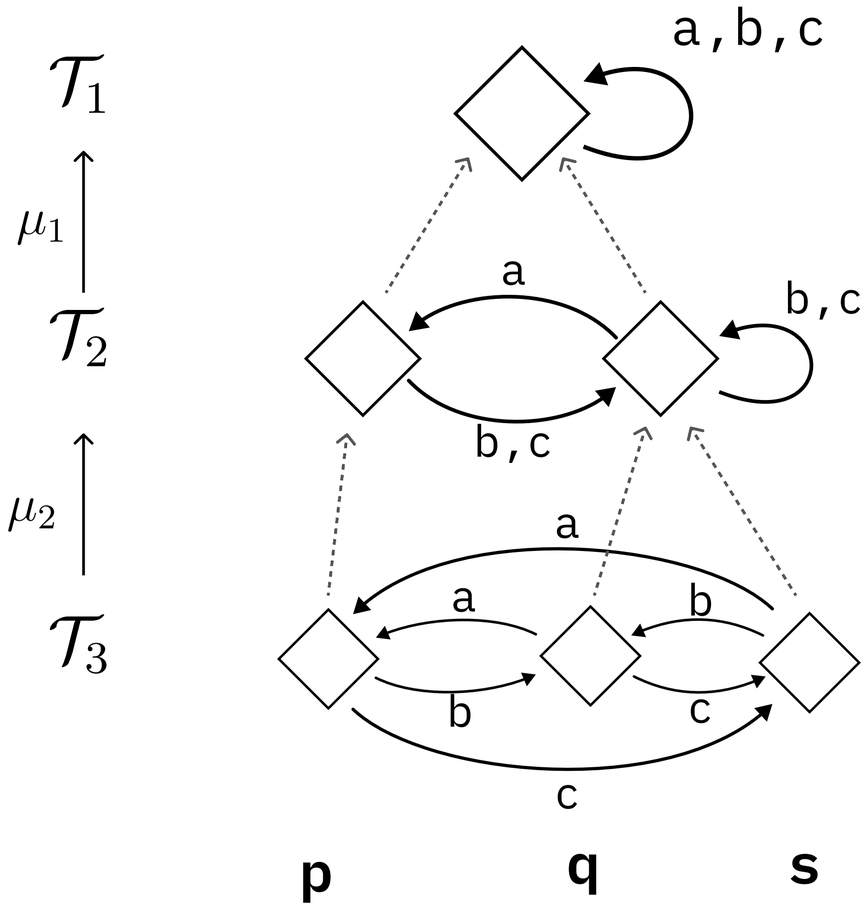
$\mu : V \to V'$

$p \xrightarrow{a} q$ in $\mathcal{T}$ $\implies$ $\mu(p) \xrightarrow{a} \mu(q)$ in $\mathcal{T}'$



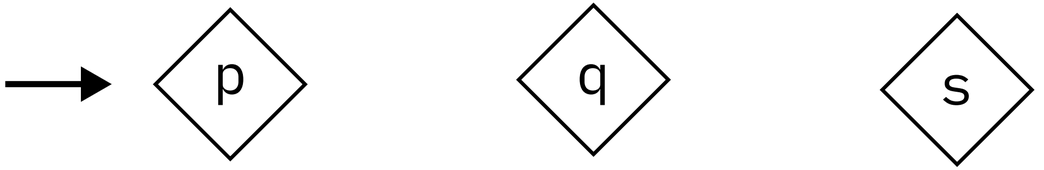
Alternating Parity Automaton

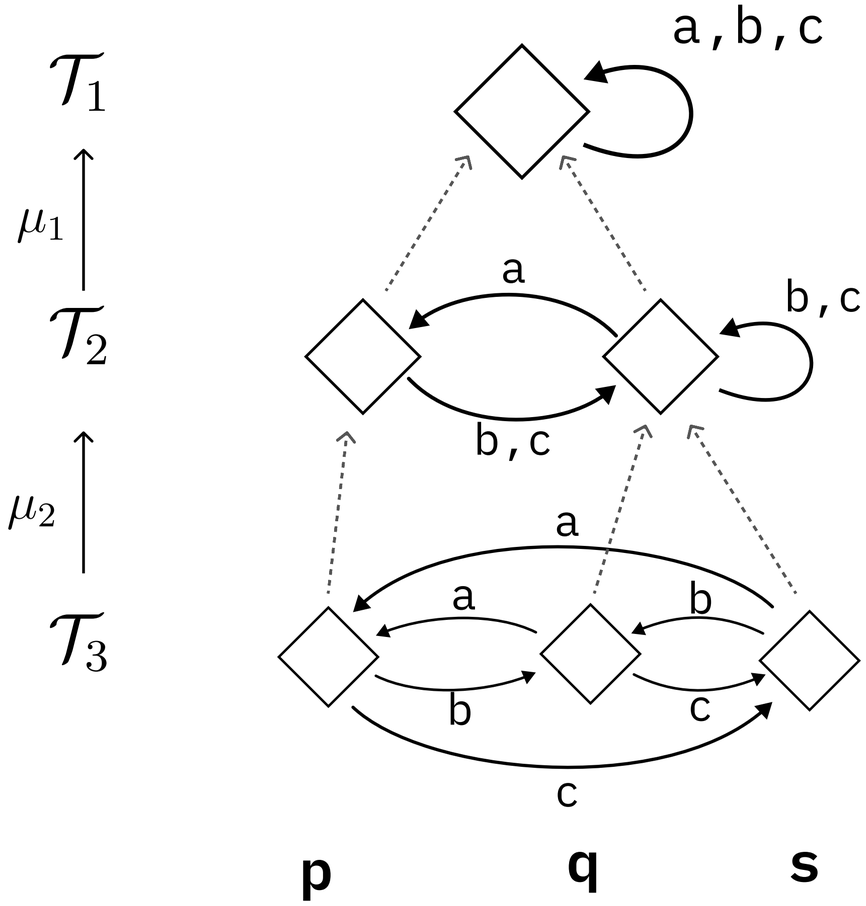




Alternating Parity Automaton

States: leaves

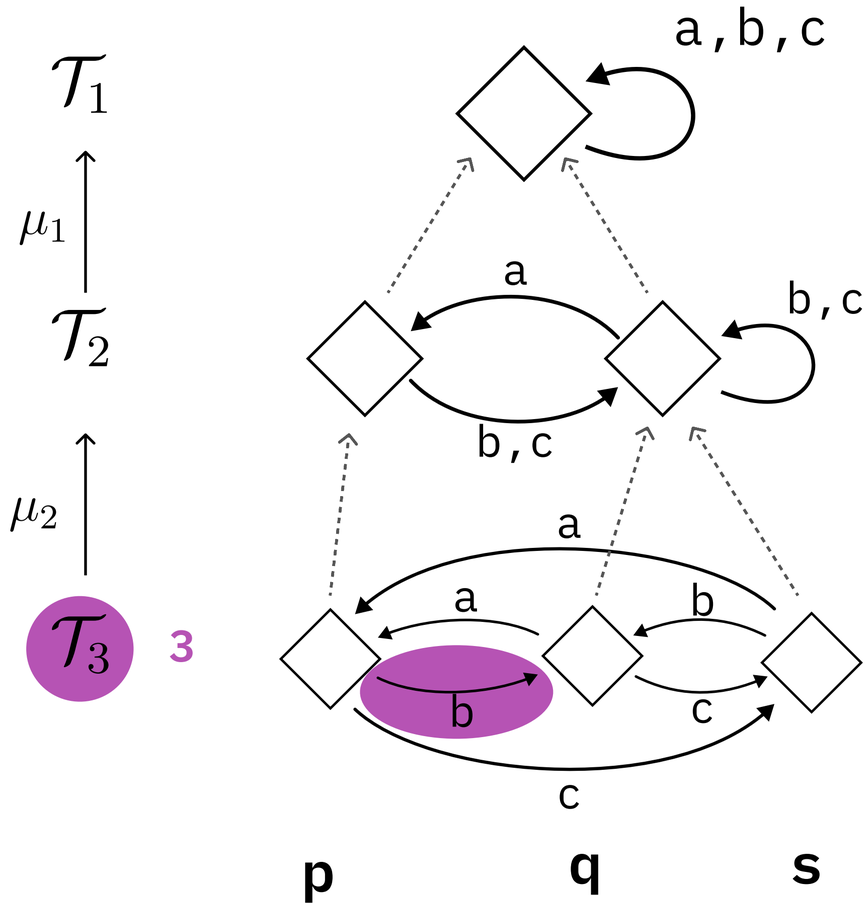




Alternating Parity Automaton

States: leaves

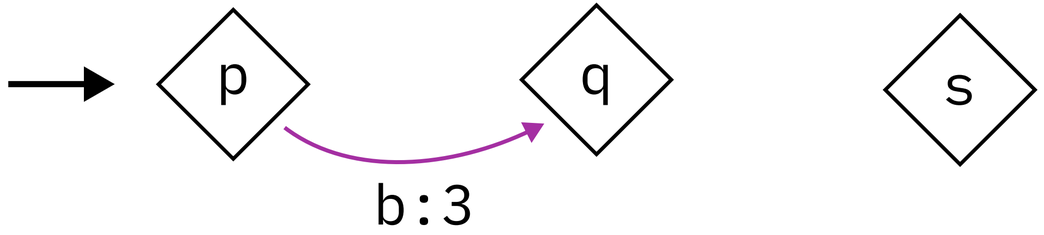
Transitions: given by the deepest $\mathcal{T}_i$ where the transition exists

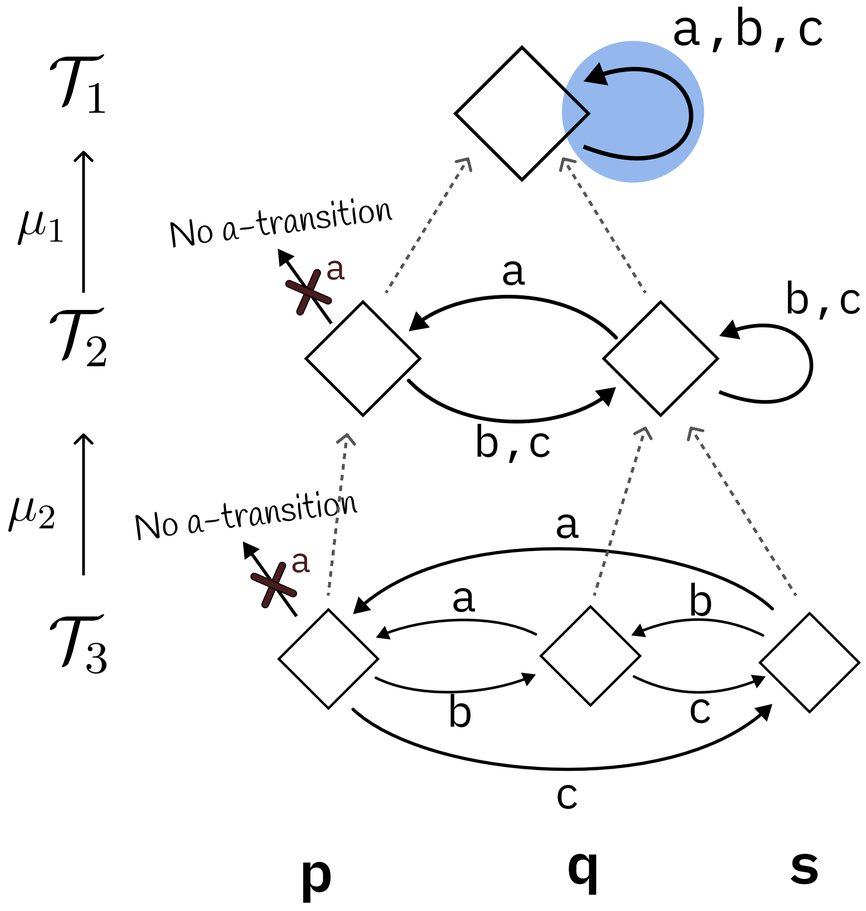


Alternating Parity Automaton

States: leaves

Transitions: given by the deepest $\mathcal{T}_i$ where the transition exists

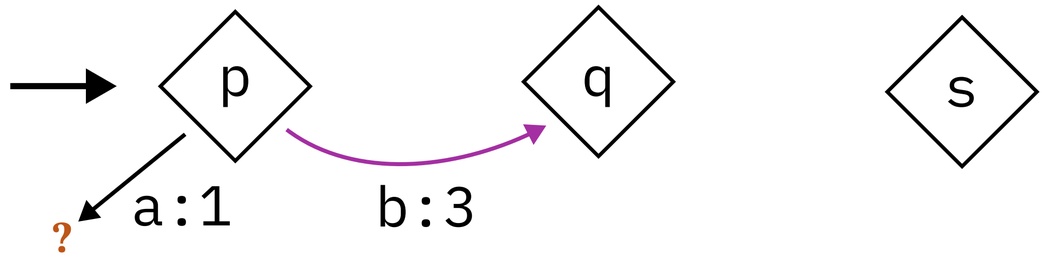


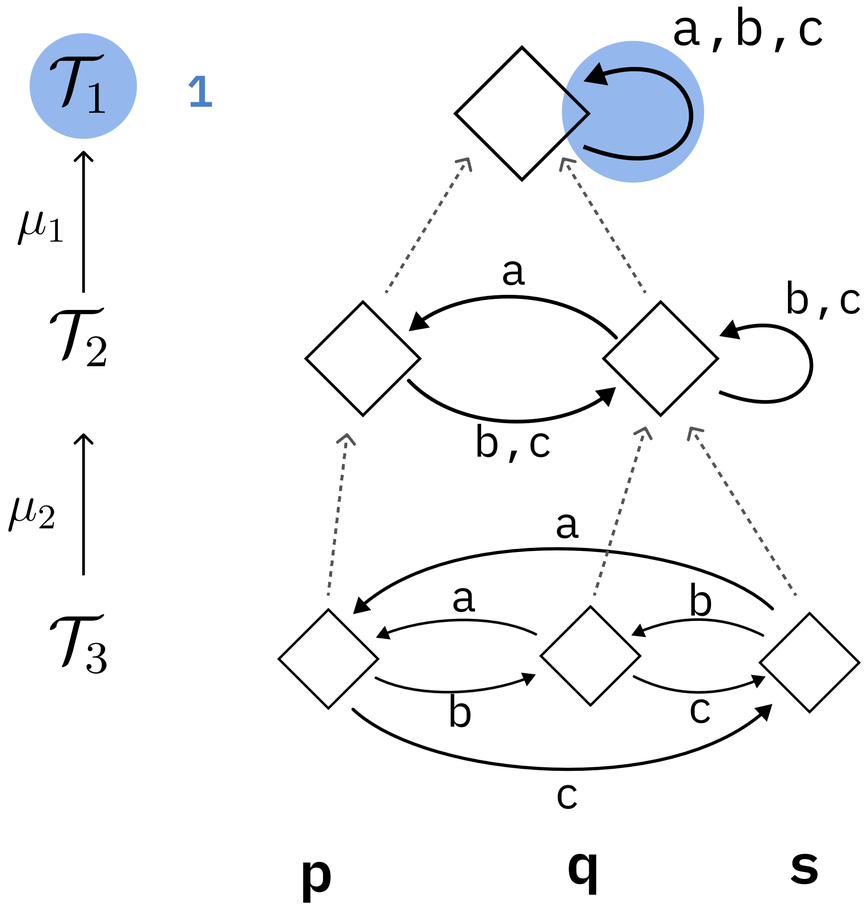


Alternating Parity Automaton

States: leaves

Transitions: given by the deepest $\mathcal{T}_i$ where the transition exists



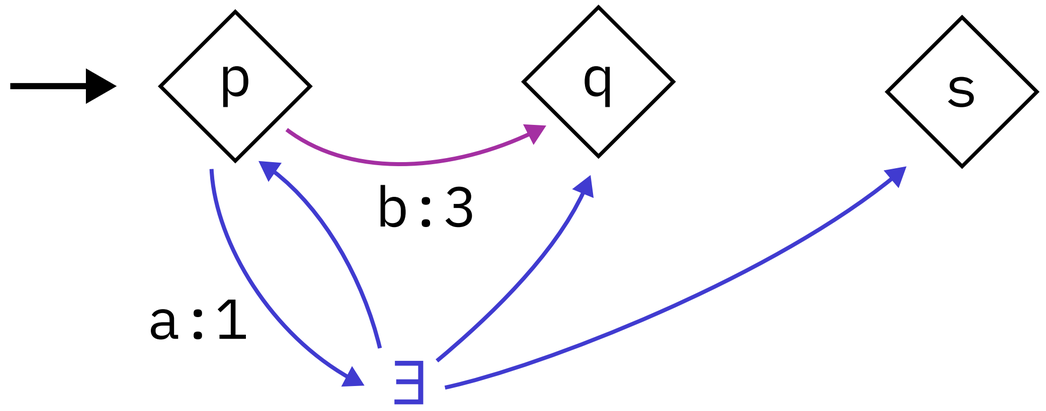


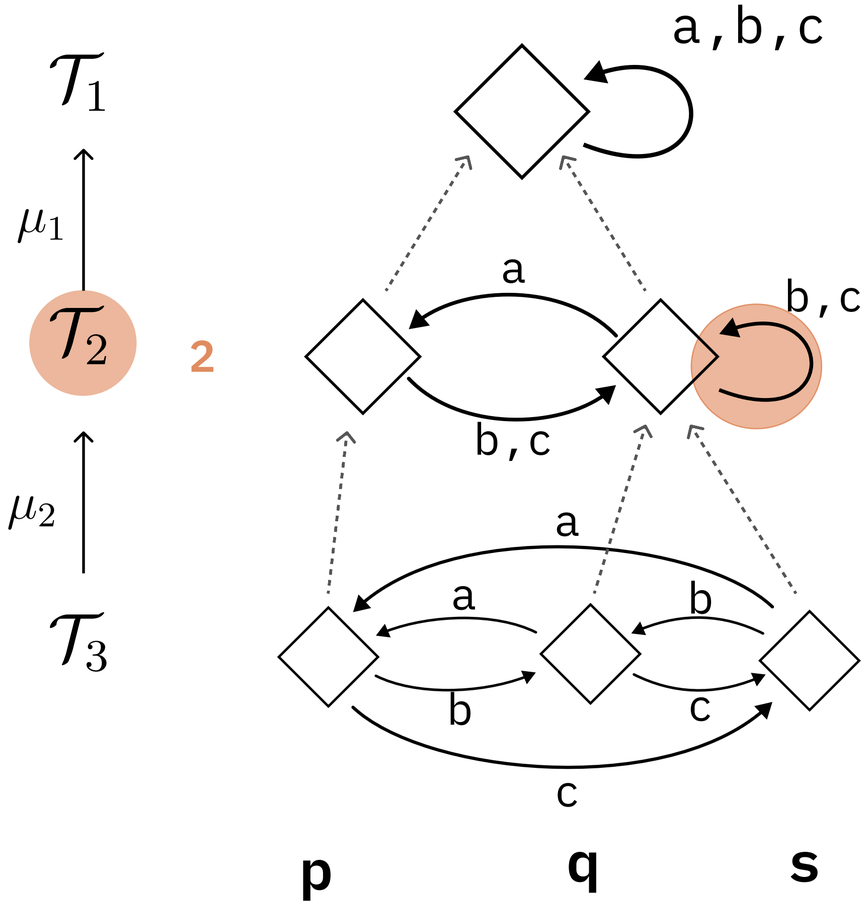
Odd → existential choice

Alternating Parity Automaton

States: leaves

Transitions: given by the deepest $\mathcal{T}_i$ where the transition exists

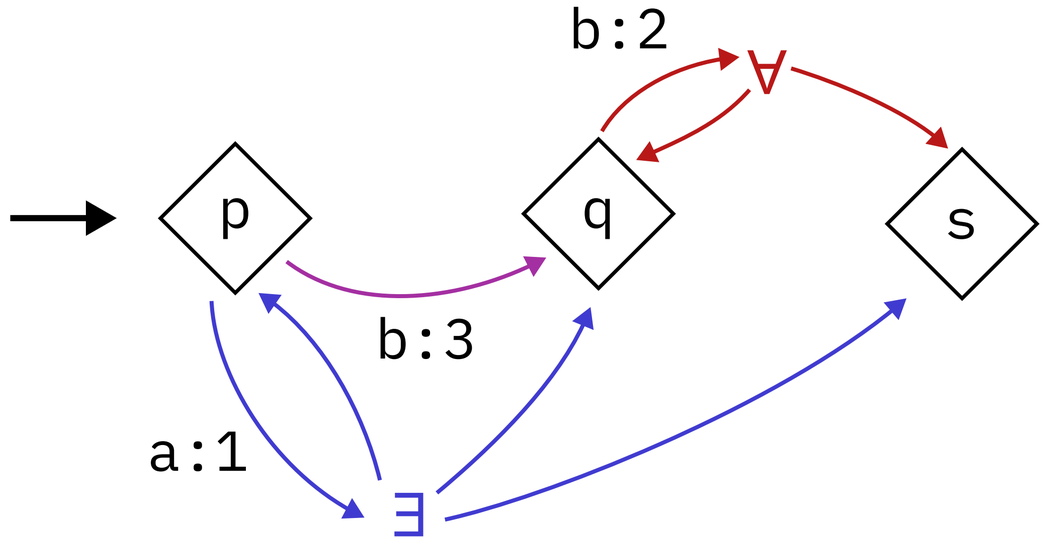




Alternating Parity Automaton

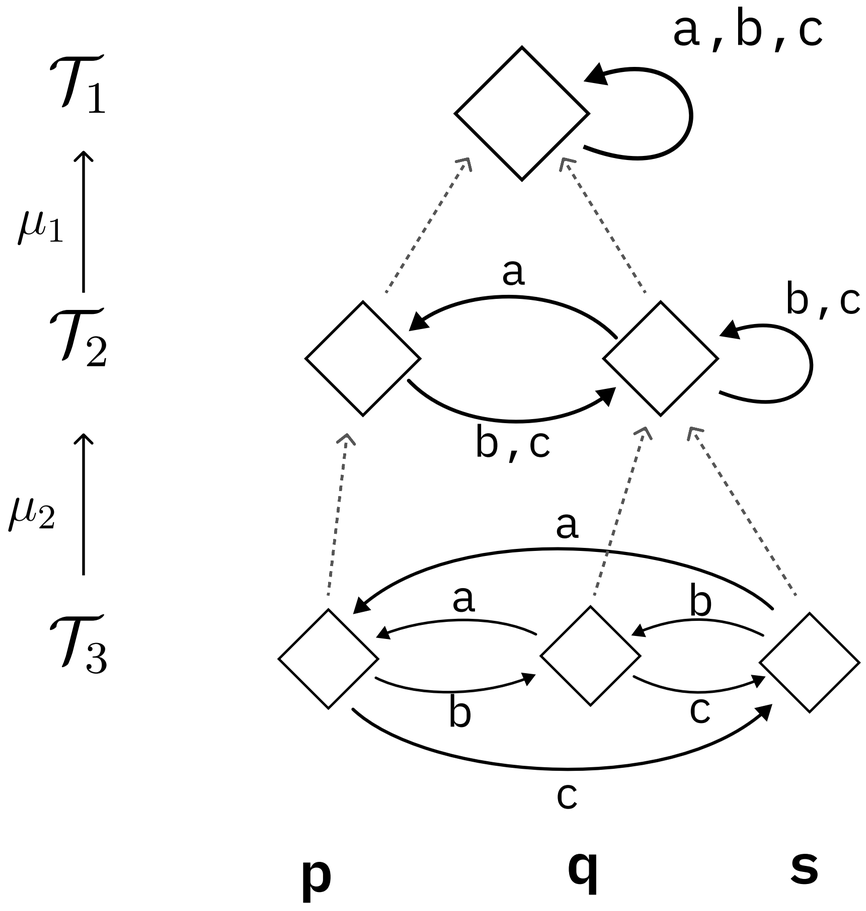
States: leaves

Transitions: given by the deepest $\mathcal{T}_i$ where the transition exists



Odd → existential choice

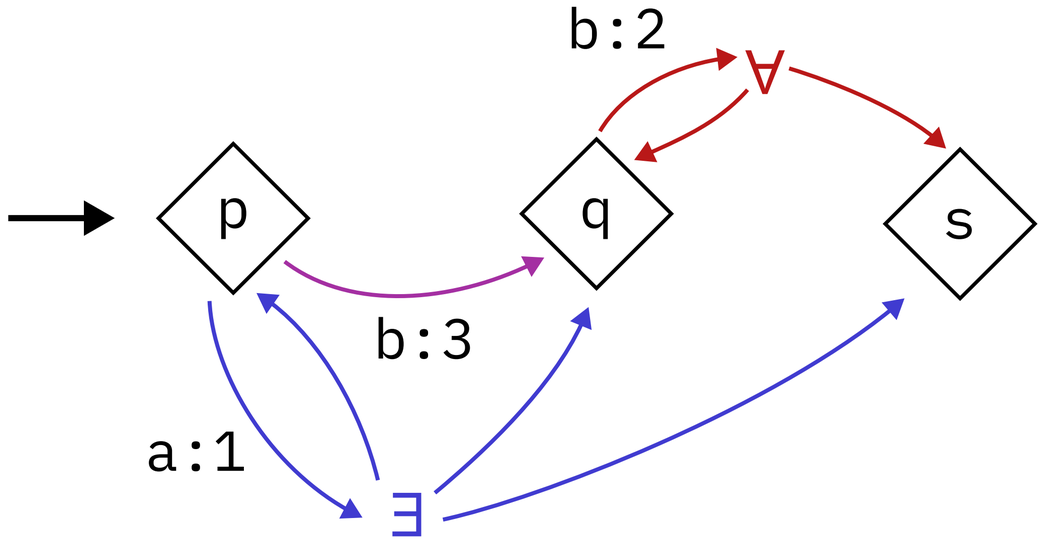
Even → universal choice



Alternating Parity Automaton

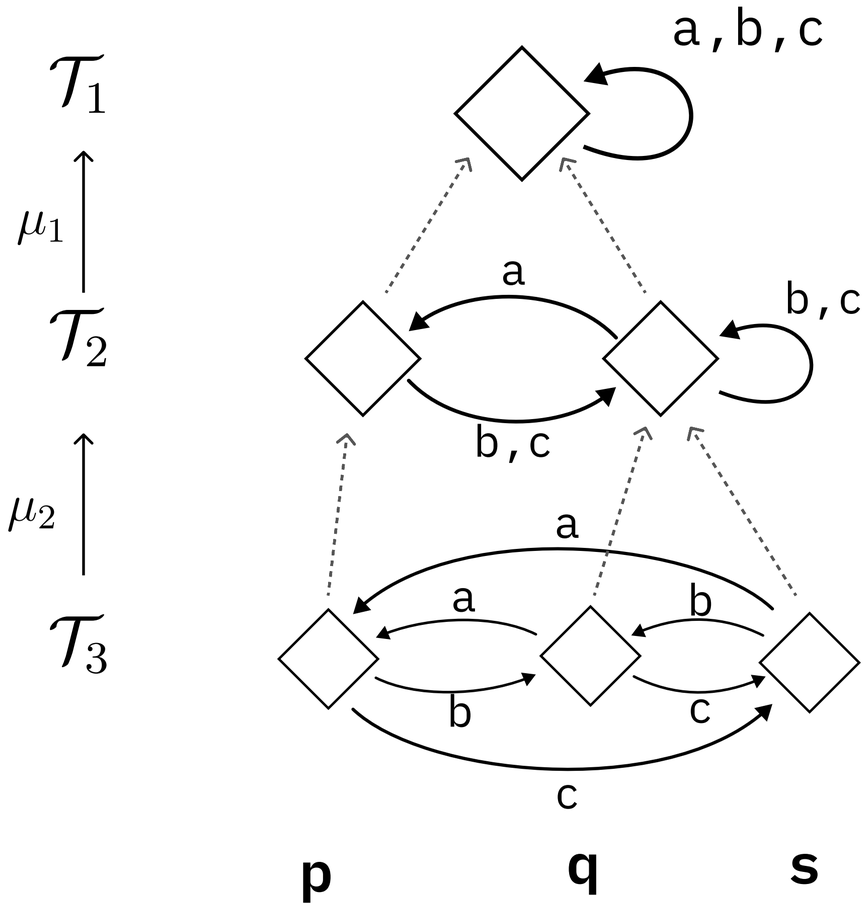
States: leaves

Transitions: given by the deepest $\mathcal{T}_i$ where the transition exists



Odd → existential choice

Even → universal choice



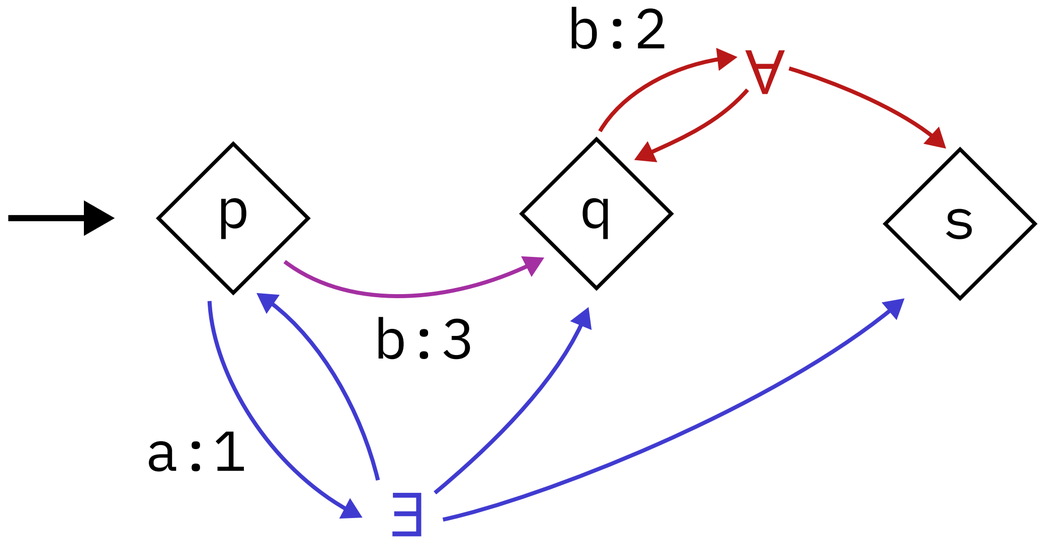
Odd → existential choice

Even → universal choice

Alternating Parity Automaton

States: leaves

Transitions: given by the deepest $\mathcal{T}_i$ where the transition exists

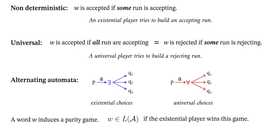


+ $\mathcal{T}_1$ must correspond to residuals

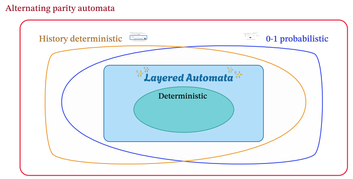
Layered Automata

well-behaved

Formalism to represent a subclass of alternating ω-automata



Some subclasses are usable in verification!!



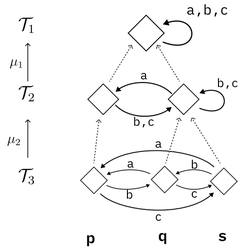
Definition

incomplete

Layered automaton: A sequence of deterministic transition systems with morphisms

$$\mathcal{T}_1 \xleftarrow{\mu_1} \mathcal{T}_2 \xleftarrow{\mu_2} \cdots \mathcal{T}_{d-1} \xleftarrow{\mu_{d-1}} \mathcal{T}_d$$

$$\mu : V \to V' \quad \text{s.t.} \quad s \xrightarrow{a} s \implies \mu s \xrightarrow{a} \mu s \text{ in } \mathcal{T}'$$



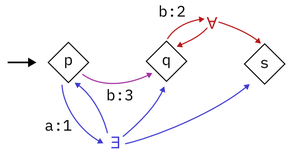
Odd → existential choice

Even → universal choice

Alternating Parity Automaton

States: leaves

Transitions: given by the deepest $\mathcal{T}_i$ where the transition exists



\+ $\mathcal{T}_1$ must correspond to residuals

Canonicity properties

✦ THEOREM: All layered automata are history deterministic and 0-1 probabilistic.

✦ THEOREM: All layered automata are history deterministic and 0-1 probabilistic.

THEOREM

Each ω-regular language admits a canonical minimal layered automaton.

It can be computed in PTIME.

✦ THEOREM: All layered automata are history deterministic and 0-1 probabilistic.

THEOREM

Each ω-regular language admits a canonical minimal layered automaton.

It can be computed in PTIME.

among layered automata

✦ THEOREM: All layered automata are history deterministic and 0-1 probabilistic.

THEOREM

Each ω-regular language admits a canonical minimal layered automaton.

It can be computed in PTIME.

among layered automata

from a given layered automaton

- THEOREM: All layered automata are history deterministic and 0-1 probabilistic.

THEOREM

Each ω-regular language admits a canonical minimal layered automaton.

It can be computed in PTIME.

among layered automata

from a given layered automaton

- Congruence-based characterisation of the minimal layered automaton.

Layered Automata

well-behaved

Formalism to represent a subclass of alternating ω-automata

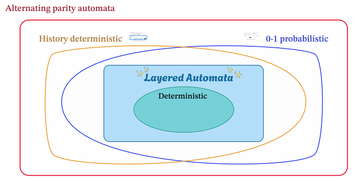
Non deterministic: w is accepted if ***some*** run is accepting.
An existential player tries to build an accepting run.

Universal: w is accepted if ***all*** run are accepting $=$ w is rejected if ***some*** run is rejecting.
A universal player tries to build a rejecting run.

Alternating automata: $p \xrightarrow{a} \{q_1, q_2, q_3\}$ $p \xrightarrow{a} \{q_1, q_2, q_3\}$
existential choices *universal choices*

A word w induces a parity game. $w \in L(\mathcal{A})$ if the existential player wins this game.

Some subclasses are usable in verification!!



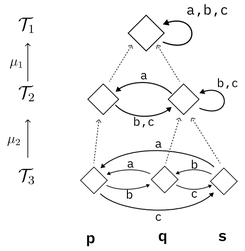
Definition

incomplete

Layered automaton: A sequence of deterministic transition systems with morphisms

$$\mathcal{T}_1 \xleftarrow{\mu_1} \mathcal{T}_2 \xleftarrow{\mu_2} \cdots \mathcal{T}_{d-1} \xleftarrow{\mu_{d-1}} \mathcal{T}_d$$

$\mu : V \to V'$: $s \xrightarrow{a} s' \implies \mu s \xrightarrow{a} \mu s'$



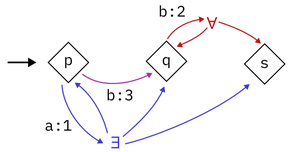
Odd → existential choice

Even → universal choice

Alternating Parity Automaton

States: leaves

Transitions: given by the deepest $\mathcal{T}_i$ where the transition exists



\+ $\mathcal{T}_1$ must correspond to residuals

Canonicity properties

✦ THEOREM: All layered automata are history deterministic and 0-1 probabilistic.

THEOREM

Each ω-regular language admits a canonical minimal layered automaton. (among layered automata)

It can be computed in PTIME. (from a given layered automaton)

✦ Congruence-based characterisation of the minimal layered automaton.

Everybody loves automata

- Finite automata ⟷ MSO-logic *(Büchi, Elgot, Trakhtenbrot '60)*

Beautiful mathematical theory!

A minimal, canonical DFA for each regular language

Efficiency in applications!

- Minimisation in $O(n \log n)$ *(Hopcroft '71)*
- Automata learning in PTIME *(Gold '67, Angluin 87')*
- Simple characterization of FO-definability

⋮

Let's generalize to infinite words!

- ω-automata ⟷ MSO-logic *(Büchi '62)*
- Applications in verification and reactive synthesis

No unique minimal deterministic automata

Use of ω-automata very costly in applications!

- × Minimisation is NP-complete *(Schewe '10, AbuRadi-Ehlers '25)*
- × Costly determinization and boolean operations
- × No efficient learning of ω-automata

The theory of deterministic parity automata is not satisfactory...

Instead, use ✨ *Layered Automata* ✨

Formalism to represent a subclass (*well-behaved*) of alternating ω-automata

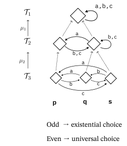
Some subclasses are usable in verification!!

Definition

incomplete

Layered automaton: A sequence of deterministic transition systems with morphisms

$$\mathcal{T}_1 \xleftarrow{\rho_2} \mathcal{T}_2 \xleftarrow{\rho_3} \cdots \mathcal{T}_{d-1} \xleftarrow{\rho_d} \mathcal{T}_d$$



Alternating Parity Automaton

States: leaves

Transitions: given by the deepest $\mathcal{T}_i$ where the transition exists

Odd → existential choice

Even → universal choice

+ $\mathcal{T}_i$ must correspond to residuals

Canonicity properties

+ THEOREM: All layered automata are history deterministic and 0-1 probabilistic.

THEOREM

Each ω-regular language admits a canonical (*among layered automata*) minimal layered automaton.

It can be computed in PTIME. (*from a given layered automaton*)

+ Congruence-based characterisation of the minimal layered automaton.

Conclusions

Layered automata admit canonical, minimal automata, computable in polynomial time

Conclusions

Layered automata admit canonical, minimal automata, computable in polynomial time

Related work

- Minimal HD coBüchi automata *(Abu Radi-Kupferman '19)*

Conclusions

Layered automata admit canonical, minimal automata, computable in polynomial time

Related work

- Minimal HD coBüchi automata *(Abu Radi-Kupferman '19)*
- Chains of coBüchi automata *(Schewe-Ehlers '22, Ehlers-Khalimov '24)*
- Rerailing automata *(Ehlers '25)*

Future work

Conjecture

The minimal layered automaton has a minimal number of states among all equivalent HD alternating parity automata.

Future work

Conjecture

Proved for Büchi and coBüchi!

The minimal layered automaton has a minimal number of states among all equivalent HD alternating parity automata.

Future work

Conjecture

Proved for Büchi and coBüchi!

The minimal layered automaton has a minimal number of states among all equivalent HD alternating parity automata.

- Passive and active learning of layered automata

Future work

Conjecture

Proved for Büchi and coBüchi!

The minimal layered automaton has a minimal number of states among all equivalent HD alternating parity automata.

- Passive and active learning of layered automata
- Implementation, boolean operations, translations from logic...

Future work

Conjecture

Proved for Büchi and coBüchi!

The minimal layered automaton has a minimal number of states among all equivalent HD alternating parity automata.

- Passive and active learning of layered automata
- Implementation, boolean operations, translations from logic...

Future work

Conjecture

Proved for Büchi and coBüchi!

The minimal layered automaton has a minimal number of states among all equivalent HD alternating parity automata.

- Passive and active learning of layered automata
- Implementation, boolean operations, translations from logic...

Thanks for your attention!

Everybody loves automata

- Finite automata ⟷ MSO-logic *(Büchi, Elgot, Trakhtenbrot '60)*

Beautiful mathematical theory!

A minimal, canonical DFA for each regular language

Efficiency in applications!

- Minimisation in $O(n \log n)$ *(Hopcroft '71)*
- Automata learning in PTIME *(Gold '67, Angluin 87')*
- Simple characterization of FO-definability

⋮

Let's generalize to infinite words!

- ω-automata ⟷ MSO-logic *(Büchi '62)*
- Applications in verification and reactive synthesis

No unique minimal deterministic automata

Use of ω-automata very costly in applications!

- ✗ Minimisation is NP-complete *(Schewe '10, AbuRadi-Ehlers '25)*
- ✗ Costly determinization and boolean operations
- ✗ No efficient learning of ω-automata

The theory of deterministic parity automata is not satisfactory...

Instead, use ✨ Layered Automata ✨

well-behaved

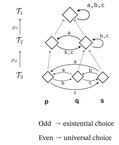
Formalism to represent a subclass of alternating ω-automata

Some subclasses are usable in verification!!

Definition

Layered automaton: A sequence of deterministic transition systems with morphisms

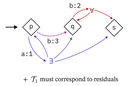
$$\mathcal{T}_1 \xleftarrow{\rho_2} \mathcal{T}_2 \xleftarrow{\rho_3} \cdots \mathcal{T}_{d-1} \xleftarrow{\rho_d} \mathcal{T}_d$$



Alternating Parity Automaton

States: leaves

Transitions: given by the deepest $\mathcal{T}_i$ where the transition exists



Odd → existential choice

Even → universal choice

+ $\mathcal{T}_i$ must correspond to residuals

Canonicity properties

- THEOREM: All layered automata are history deterministic and 0-1 probabilistic.

THEOREM

Each ω-regular language admits a canonical minimal layered automaton. *among layered automata*

It can be computed in PTIME. *from a given layered automaton*

- Congruence-based characterisation of the minimal layered automaton.

Conclusions

Layered automata admit canonical, minimal automata, computable in polynomial time

Related work

- Minimal HD coBüchi automata *(Abu Radi-Kupferman '19)*
- Chains of coBüchi automata *(Schewe-Ehlers '22, Ehlers-Khalimov '24)*
- Rerailing automata *(Ehlers '25)*

Future work

CONJECTURE *Proved for Büchi and coBüchi!*

The minimal layered automaton has a minimal number of states among all equivalent HD alternating parity automata.

- Passive and active learning of layered automata
- Implementation, boolean operations, translations from logic...

Thanks for your attention!

Layered Automata

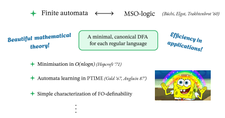
A canonical model for ω-regular languages

Antonio Casares · RPTU Kaiserslautern

Joint work with: Christof Löding and Igor Walukiewicz

Everybody loves automata

Let's generalize to infinite words!



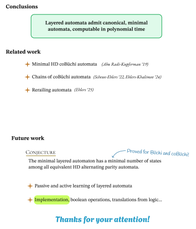
The theory of deterministic parity automata is not satisfactory...

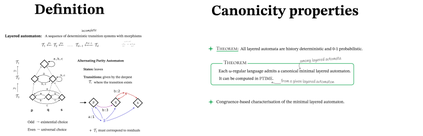
Instead, use Layered Automata

well-behaved

Formalism to represent a subclass of alternating ω-automata

Some subclasses are usable in verification!!





GT Vérif, 7 Novembre 2025