

From logic to games

Igor Walukiewicz
CNRS, LaBRI Bordeaux

Plan

- Understanding the mu-calculus (parity games).
- Advances in the mu-calculus.
- Hierarchy problems.
- Parity games on infinite graphs: pushdown graphs, . . .
- More complicated winning conditions.

Part I

Understanding the mu-calculus: parity games

- Syntax: $P \mid \neg P \mid X \mid \alpha \mid \alpha \vee \beta \mid \alpha \wedge \beta \mid \langle a \rangle \alpha \mid [a] \alpha \mid \mu X. \alpha \mid \nu X. \alpha$
- Semantics in a transition system $\mathcal{M} = \langle V, \{E_a\}_{a \in Act}, P^{\mathcal{M}}, \dots \rangle$;
we need $Val : Var \rightarrow \mathcal{P}(V)$

$$\llbracket P \rrbracket_{Val}^{\mathcal{M}} = P^{\mathcal{M}}$$

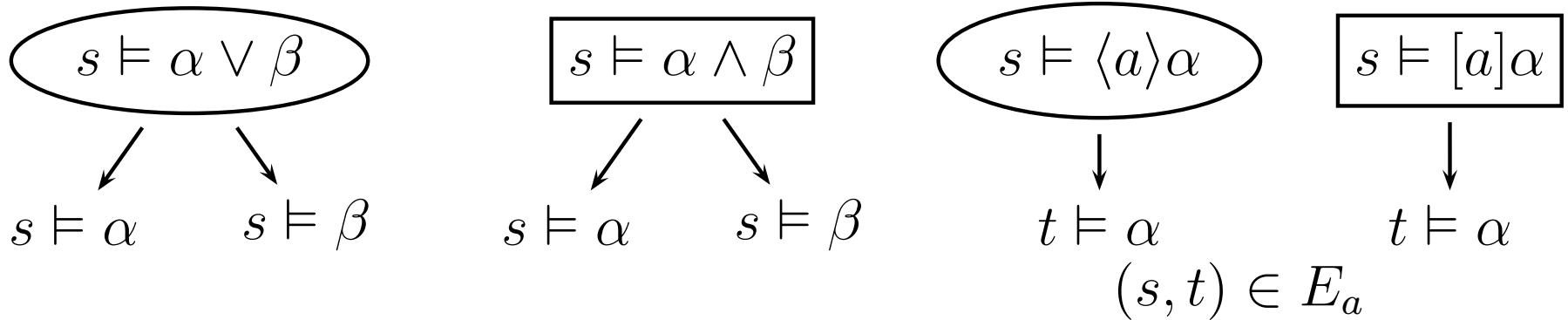
$$\llbracket X \rrbracket_{Val}^{\mathcal{M}} = Val(X)$$

$$\llbracket \langle a \rangle \alpha \rrbracket_{Val}^{\mathcal{M}} = \{v : \exists v'. E_a(v, v') \wedge v' \in \llbracket \alpha \rrbracket_{Val}^{\mathcal{M}}\}$$

$$\llbracket \mu X. \alpha(X) \rrbracket_{Val}^{\mathcal{M}} = \bigcap \{S \subseteq V : \llbracket \alpha(S) \rrbracket_{Val}^{\mathcal{M}} \subseteq S\}$$

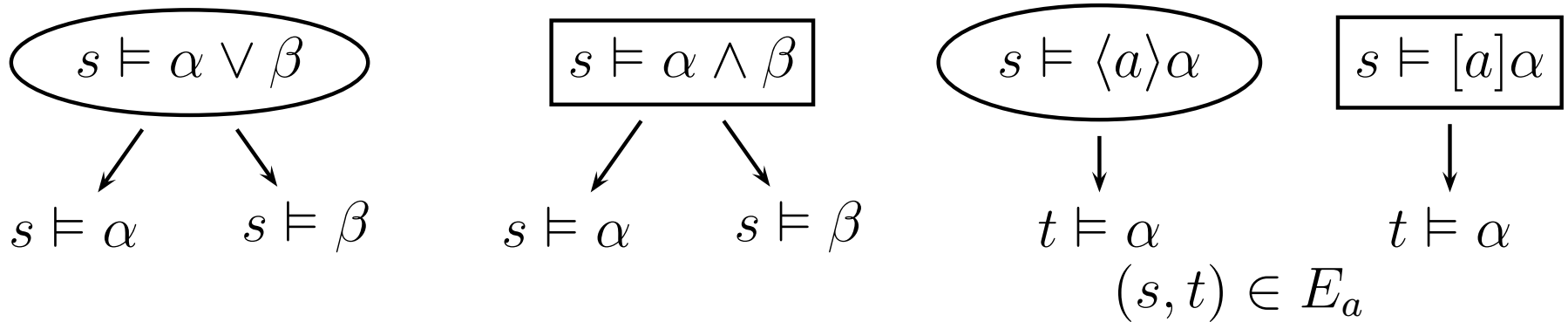
We will give a characterization of the semantics in terms of two player games.

- We are given a transition system \mathcal{M} and a formula α_0 .
- Player **Eva** wants to show that α_0 holds in a state s : $s \models \alpha_0$.
Player **Adam** wants to show that it does not hold.
- Game rules

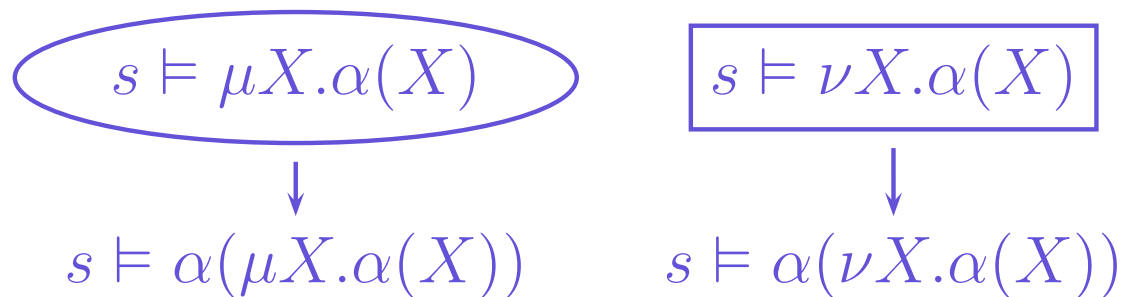


$s \models P$ Eve wins if $s \in P^{\mathcal{M}}$; $s \models \neg P$ Eve wins if $s \notin P^{\mathcal{M}}$.

- We are given a transition system \mathcal{M} and a formula α_0 .
- Player **Eva** wants to show that α_0 holds in a state s : $s \models \alpha_0$.
Player **Adam** wants to show that it does not hold.
- Game rules



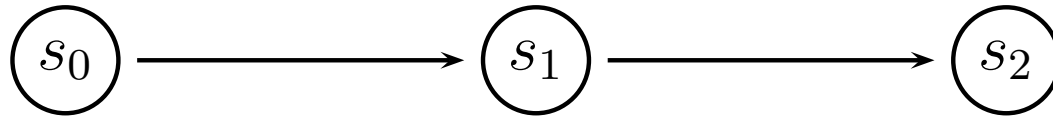
$s \models P$ Eve wins if $s \in P^M$; $s \models \neg P$ Eve wins if $s \notin P^M$.



- The last two rules may be a source of infinite plays.

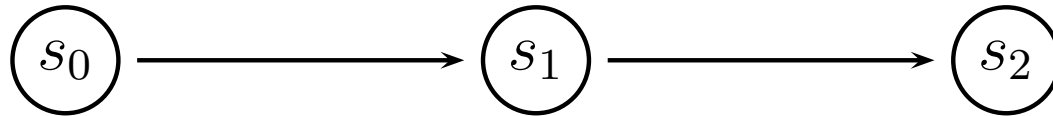
- Reachability: $\langle \cdot \rangle^* P \equiv \mu X. P \vee \langle \cdot \rangle X.$

- Reachability: $\langle \cdot \rangle^* P \equiv \mu X. P \vee \langle \cdot \rangle X$.



$$\alpha \equiv \mu X. P \vee \langle \cdot \rangle X$$

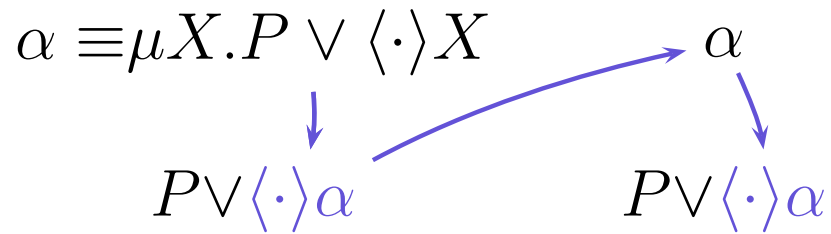
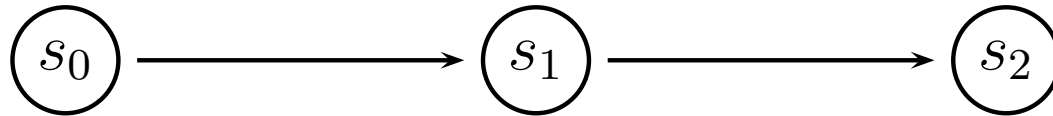
- Reachability: $\langle \cdot \rangle^* P \equiv \mu X. P \vee \langle \cdot \rangle X.$



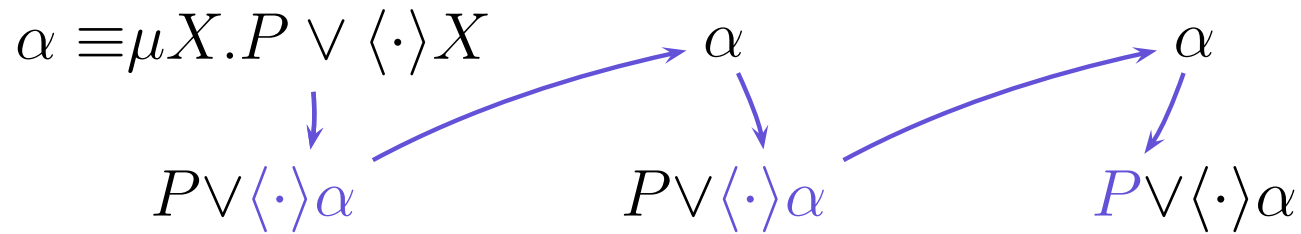
$$\alpha \equiv \mu X. P \vee \langle \cdot \rangle X$$

$$P \vee \langle \cdot \rangle \alpha$$

- Reachability: $\langle \cdot \rangle^* P \equiv \mu X. P \vee \langle \cdot \rangle X$.

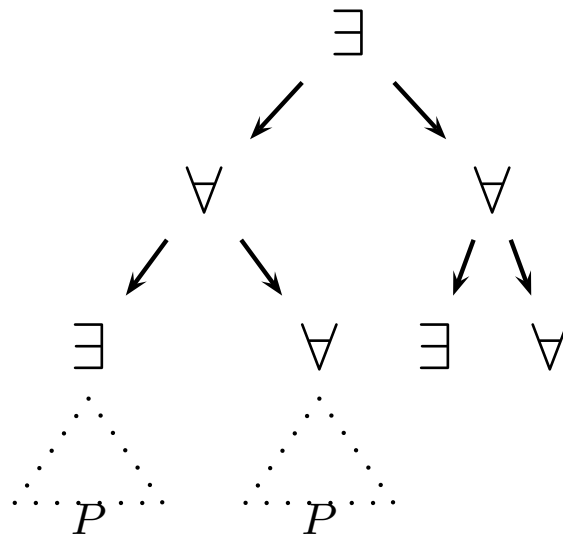


- Reachability: $\langle \cdot \rangle^* P \equiv \mu X. P \vee \langle \cdot \rangle X$.

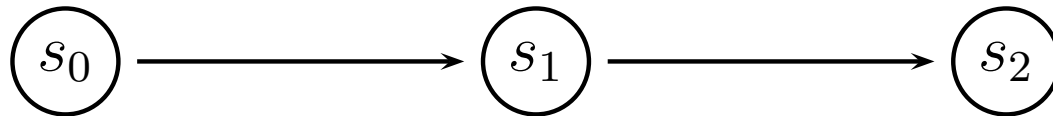


- Reachability: $\langle \cdot \rangle^* P \equiv \mu X. P \vee \langle \cdot \rangle X$.
- Existential until: $\exists(Q \cup P) \equiv \mu X. P \vee (Q \wedge \langle \cdot \rangle X)$.
- Universal until: $\forall(Q \cup P) \equiv \mu X. P \vee (Q \wedge [] X)$.
- Alternating reachability

$$\mu X. P \vee (Q_{\exists} \wedge \langle \cdot \rangle X) \vee (Q_{\forall} \wedge [] X)$$



● $\mu X. P \vee \langle \cdot \rangle X$ holds in s whenever from s one can reach a state where P holds.



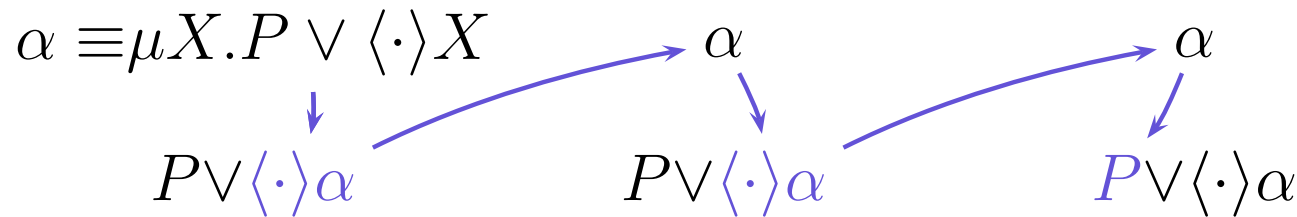
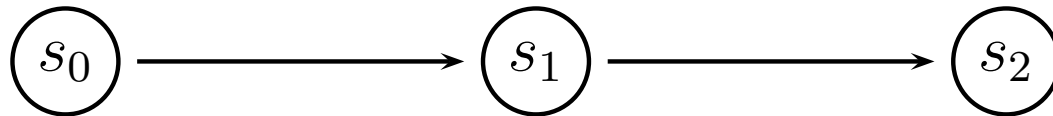
$$\alpha \equiv \mu X. P \vee \langle \cdot \rangle X$$

$$P \vee \langle \cdot \rangle \alpha$$

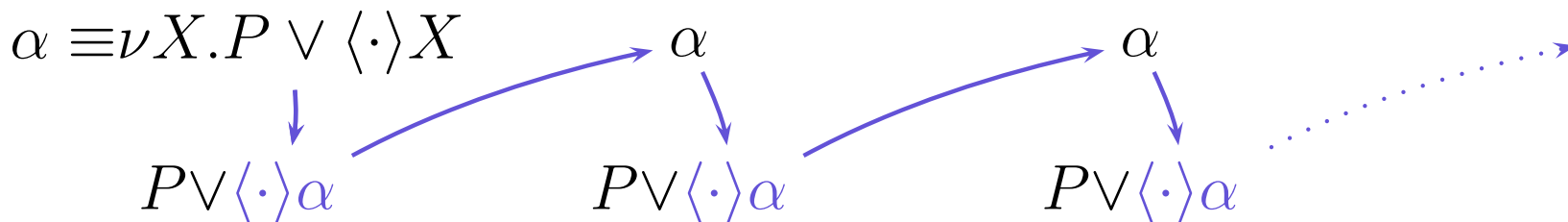
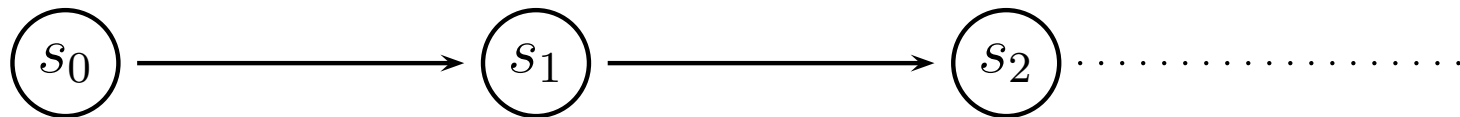
$$P \vee \langle \cdot \rangle \alpha$$

$$P \vee \langle \cdot \rangle \alpha$$

● $\mu X. P \vee \langle \cdot \rangle X$ holds in s whenever from s one can reach a state where P holds.



● $\nu X. P \vee \langle \cdot \rangle X$ holds in s also if there is an infinite path from s .



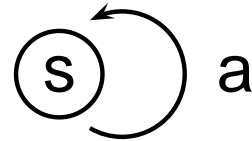
Examples (alternating fixpoints)

- Almost always P on some path

$$\mu Y. \nu X. (P \wedge \langle \cdot \rangle X) \vee \langle \cdot \rangle Y$$

- Infinitely often P on some path

$$\nu X. \mu Y. (P \wedge \langle \cdot \rangle X) \vee \langle \cdot \rangle Y$$



$$\begin{array}{c}
 s \models \mu X. \langle a \rangle X \\
 \downarrow \\
 s \models \langle a \rangle \mu X. \langle a \rangle X \\
 \downarrow \\
 s \models \mu X. \langle a \rangle X \\
 \downarrow \\
 \vdots
 \end{array}$$

$$\begin{array}{c}
 s \models \nu X. \langle a \rangle X \\
 \downarrow \\
 s \models \langle a \rangle (\nu X. \langle a \rangle X) \\
 \downarrow \\
 s \models \nu X. \langle a \rangle X \\
 \downarrow \\
 \vdots
 \end{array}$$

- Eve should win in the second game but not in the first.

$$\mu X.\beta(X) = \bigcup_{\tau \in \text{Ord}} \mu^\tau X.\beta(X)$$

$$\llbracket \mu^0 X.\beta(X) \rrbracket_{\text{Val}}^{\mathcal{M}} = \emptyset$$

$$\llbracket \mu^{\tau+1} X.\beta(X) \rrbracket = \llbracket \beta(X) \rrbracket_{\text{Val}}^{\mathcal{M}}[\llbracket \mu^\tau X.\beta(X) \rrbracket_{\text{Val}}^{\mathcal{M}}/X]$$

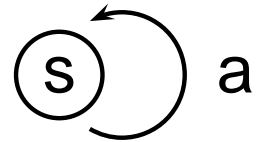
$$\llbracket \mu^\tau X.\beta(X) \rrbracket_{\text{Val}}^{\mathcal{M}} = \bigcup_{\tau' < \tau} \llbracket \mu^{\tau'} X.\beta(X) \rrbracket_{\text{Val}}^{\mathcal{M}} \quad \text{if } \tau \text{ is a limit ordinal}$$

$$\nu X.\beta(X) = \bigcap_{\tau \in \text{Ord}} \nu^\tau X.\beta(X)$$

$$\llbracket \nu^0 X.\beta(X) \rrbracket_{\text{Val}}^{\mathcal{M}} = V$$

$$\llbracket \nu^{\tau+1} X.\beta(X) \rrbracket = \llbracket \beta(X) \rrbracket_{\text{Val}}^{\mathcal{M}}[\llbracket \nu^\tau X.\beta(X) \rrbracket_{\text{Val}}^{\mathcal{M}}/X]$$

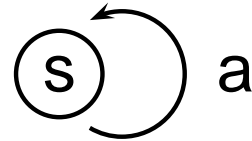
$$\llbracket \nu^\tau X.\beta(X) \rrbracket_{\text{Val}}^{\mathcal{M}} = \bigcap_{\tau' < \tau} \llbracket \nu^{\tau'} X.\beta(X) \rrbracket_{\text{Val}}^{\mathcal{M}} \quad \text{if } \tau \text{ is a limit ordinal}$$



$$\begin{array}{c}
 s \models \mu^\tau X.\langle a \rangle X \\
 \downarrow \\
 s \models \langle a \rangle (\mu^{\tau-1} X.\langle a \rangle X) \\
 \downarrow \\
 s \models \mu^{\tau-1} X.\langle a \rangle X \\
 \downarrow \\
 \vdots
 \end{array}$$

$$\begin{array}{c}
 s \models \nu^\tau X.\langle a \rangle X \\
 \downarrow \\
 s \models \langle a \rangle (\nu^\tau X.\langle a \rangle X) \\
 \downarrow \\
 s \models \nu^\tau X.\langle a \rangle X \\
 \downarrow \\
 \vdots
 \end{array}$$

- Eve should win in the second game but not in the first.



$$\begin{array}{c}
 s \models \mathbf{1} \mu^\tau X. \langle a \rangle X \\
 \downarrow \\
 s \models \mathbf{3} \langle a \rangle (\mu^{\tau-1} X. \langle a \rangle X) \\
 \downarrow \\
 s \models \mathbf{1} \mu^{\tau-1} X. \langle a \rangle X \\
 \downarrow \\
 \vdots
 \end{array}$$

$$\begin{array}{c}
 s \models \mathbf{2} \nu^\tau X. \langle a \rangle X \\
 \downarrow \\
 s \models \mathbf{3} \langle a \rangle (\nu^\tau X. \langle a \rangle X) \\
 \downarrow \\
 s \models \mathbf{2} \nu^\tau X. \langle a \rangle X \\
 \downarrow \\
 \vdots
 \end{array}$$

- Eve should win in the second game but not in the first.

Assign rank **1** to μ -regeneration and rank **2** to ν -regeneration.

- Make Eve win iff the smallest number appearing infinitely often is even.

Defining winning conditions

$$\mu X_1. \nu X_2. \mu X_3. \nu X_4 \dots \varphi(X_1, X_2, \dots)$$

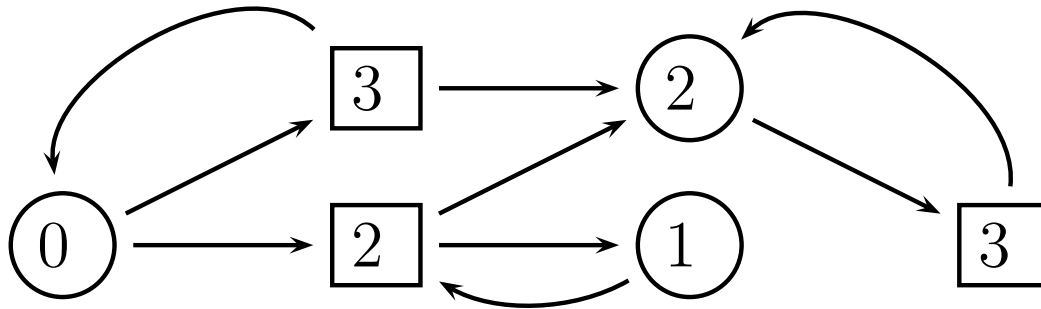
1 2 3 4 ...

- μ 's have odd ranks,
- ν 's have even ranks,
- if β is a subformula of α then β has bigger rank than α .

● The winning condition is the **parity condition**:
Eve wins if the smallest rank appearing infinitely often is even

$$\mu_1 Y. \nu_2 X. (P \wedge \langle \cdot \rangle X) \vee \langle \cdot \rangle Y \quad \nu_0 X. \mu_1 Y. (P \wedge \langle \cdot \rangle X) \vee \langle \cdot \rangle Y$$

- The model checking problem $(\mathcal{M}, s \models \alpha_0)$ is reduced to deciding if Eve wins in a **parity game**:



● The winning condition is the parity condition:
Eve wins if the smallest rank appearing infinitely often is even

- Winning a game means to have a strategy that guarantees a win no matter what the opponent does.

Part II

Advances in the mu-calculus

- Model checking.
- Satisfiability and completeness.
- Expressive power
- Alternation hierarchy.

● Model checking is linear time reducible to solving parity games.

- There is also linear time reduction in the opposite direction. (There is a formula defining the winning positions)
- So the two problems are equivalent up to linear time reductions.
- The problem is in $UP \cap co-UP$ [Jurdzinski].
We do not know if the problem is in P_{TIME} .
- There is an subexponential algorithm $n^{O(\sqrt{n})}$ [Jurdzinski, Paterson, Zwick]
- There are also algorithms working in time $n^{O(d/2+1)}$.

Satisfiability and complete axiomatization

- The satisfiability problem for the mu-calculus is EXPTIME-complete. [Emerson & Jutla]

- There is a complete axiomatization of the logic

modal logic + axiomatization of fixpoints:

$$\mu x.\alpha(X) \equiv \alpha(\mu X.\alpha(X)) \qquad \frac{\alpha(\beta) \Rightarrow \beta}{\mu X.\alpha(X) \Rightarrow \beta}$$

$$\nu X.\alpha(X) \equiv \neg\mu X.\neg\alpha(\neg X)$$

- The logic has also some other nice properties like, for example:
finite model property [Kozen]
interpolation property [Holdenbergl].

- MSOL (monadic second order logic) is an extension of FOL with set quantification.

$$R(x, y) \mid \varphi \vee \psi \mid \neg\varphi \mid \exists x. \varphi \mid y \in X \mid \exists X. \varphi$$

- Semantics: $M, V \models \varphi$:
 - $M, V \models y \in X$ if $V(y) \in V(X)$;
 - $M, V \models \exists X. \varphi$ if there is $S \subseteq M$, s.t., $M, V[S/X] \models \varphi$.

Thm [Niwiński]: Over binary trees μ -calc \equiv MSOL.

Fact: μ -calculus properties are bisimulation invariant (if $M, s \models \alpha$ and $(M, s) \approx (M, s')$ then $M, s' \models \alpha$.)

Thm [Janin & W.]: A property of transition systems is expressible in the μ -calculus iff it is expressible in MSOL and bisimulation invariant.

- $\Sigma_0^\mu = \Pi_0^\mu$ — formulas without fixpoints.
- Σ_{i+1}^μ — closure of Π_i^μ under conjunction, disjunction, substitutions and application of the least fixpoint operator μ :
$$\mu X. \alpha \in \Sigma_{i+1}^\mu \quad \text{if} \quad \alpha \in \Sigma_{i+1}^\mu$$
- Π_{i+1}^μ — similar closure but of Σ_i^μ (of course now it is closed under ν not μ).

The goal is to study the power of alternation of fixpoints

- Examples:

$$\begin{aligned} & \mu Y. \nu X. (P \wedge \langle \cdot \rangle X) \vee \langle \cdot \rangle Y \\ & \mu Y. (\nu X. (P \wedge \langle \cdot \rangle X)) \vee \langle \cdot \rangle Y \end{aligned}$$

- Understanding the power of fixpoint alternation.
- Application of diagonalization in a context when there is no pairing and when formalism is decidable.
- Related to complexity of model checking.
- Has many different characterizations: automata indices, Borel classes

Automata on infinite strings

- Instead of considering transition systems we focus on words.

- **Word:** $w : \mathbb{N} \rightarrow \Sigma$.

$$\mathcal{A} = \langle Q, \Sigma, q^0 \in Q, \delta : Q \times \Sigma \rightarrow \mathcal{P}(Q), \Omega : Q \rightarrow \mathbb{N} \rangle$$

- **Run** of \mathcal{A} on a word $w = a_0a_1 \dots$:

$$q^0 = q_0 \xrightarrow{a_0} q_1 \xrightarrow{a_1} q_2 \cdots \cdots \cdots \quad q_{i+1} \in \delta(q_i, a_i)$$

- Run is **successful** if $\Omega(q_0), \Omega(q_1), \Omega(q_2) \dots$ satisfies the parity condition:

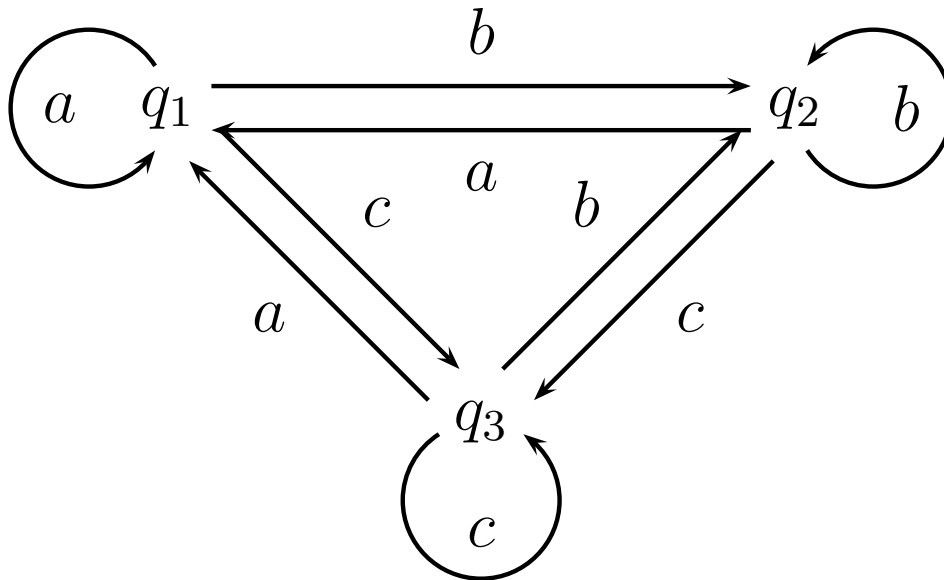
$$\liminf_{n \rightarrow \infty} \Omega(q_i) \quad \text{is even}$$

- $L(\mathcal{A})$ is the set of words accepted by \mathcal{A} .

Def: $L \subseteq \Sigma^\omega$ is **regular** iff it is the language of some automaton.

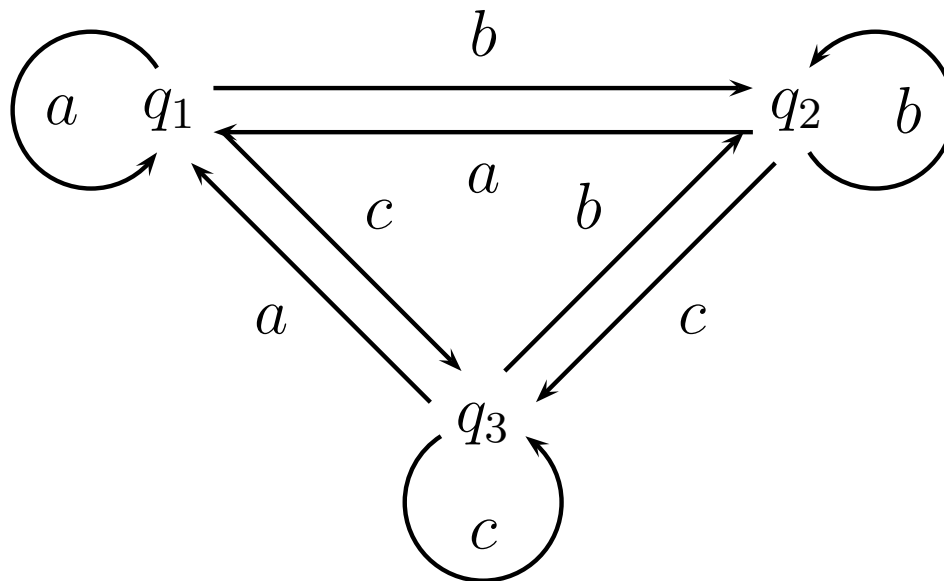
$\Sigma = \{a, b, c\}$ a appears finitely often and b inf often

- Parity automaton (deterministic)



$\Sigma = \{a, b, c\}$ a appears finitely often and b inf often

● Parity automaton (deterministic)



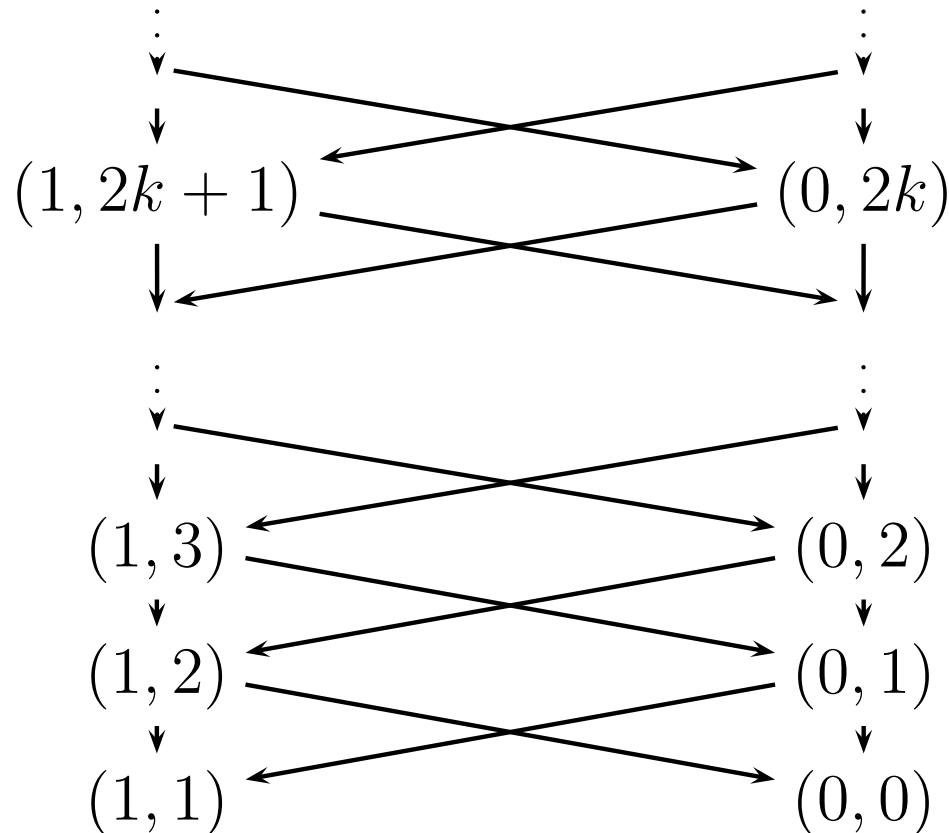
$q_1 \xrightarrow{a} q_1 \xrightarrow{b} q_2 \xrightarrow{c} q_3 \xrightarrow{b} q_2 \xrightarrow{a} q_1 \cdots$

The word is accepted if q_1 appears finitely often and q_2 appears infinitely often.

Hierarchy of acceptance conditions

Rem: If the range of Ω is $\{2, \dots, i\}$ then $\Omega'(x) = \Omega(x) - 2$ defines the same winning condition.

Cor: Interesting ranges are $\{0, \dots, i\}$ and $\{1, \dots, i + 1\}$.



Fact[Büchi]: Nondeterministic automata hierarchy collapses on $(0, 1)$ level (Büchi automata).

Fact[Wagner]: Deterministic automata hierarchy is strict.

● Let $\Sigma_n = \{1, \dots, n\}$ define:

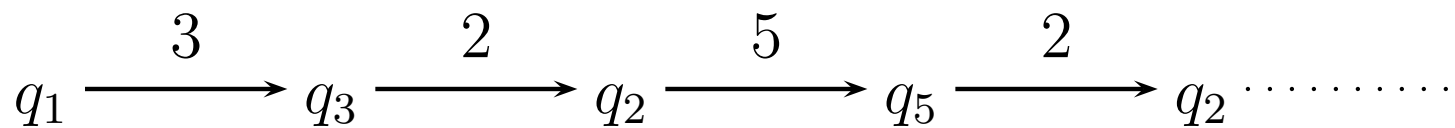
$$M_n = \{w \in \Sigma_n^\omega : \liminf_{n \rightarrow \infty} w(n) \text{ is even}\}$$

$$N_n = \{w \in \Sigma_n^\omega : \liminf_{n \rightarrow \infty} w(n) \text{ is odd}\}$$

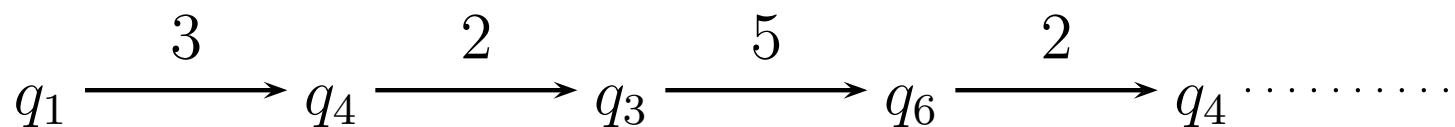
Fact: M_n is a $(1, n)$ language but not a $(0, n-1)$ language. Dually for N_n .

$$\langle Q = \{q_1, \dots, q_n\}, \Sigma_n, q_1, \delta, \Omega \rangle$$

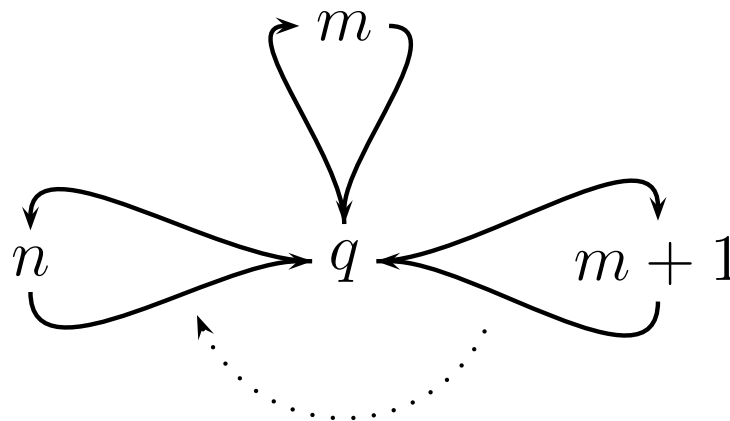
- $\Omega(q_i) = i$
- $\delta(*, i) = q_i$



- For N_n the same but $\delta(*, i) = q_{i+1}$.

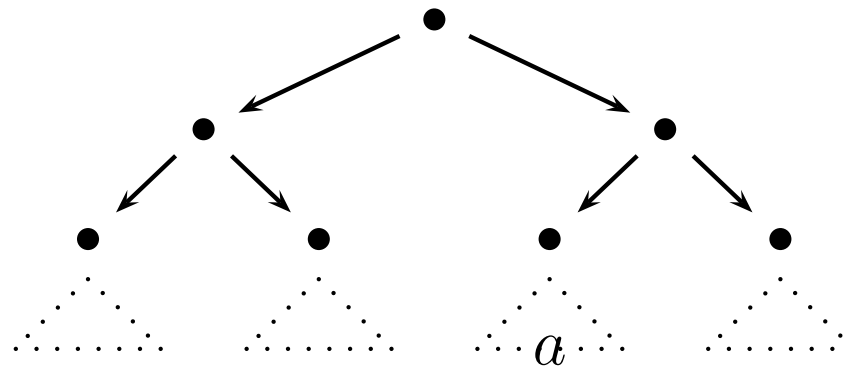


- Fix a deterministic automaton $\mathcal{A} = \langle Q, \Sigma, q, \delta, \Omega \rangle$.
- A **graph of an automaton** is $\langle Q, E \rangle$ with $(q, q') \in E$ iff $q' \in \delta(q, a)$ for some a .
- A **k -loop** is a nontrivial loop the minimal priority being k .
- A state q is a **m - n flower** if for every $k \in \{m, \dots, n\}$ there is a k -loop containing q .



Thm [Niwiński & W.]: The complexity of $L(\mathcal{A})$ is determined by the biggest size of a flower in \mathcal{A} .

- Hierarchy of nondeterministic automata on words collapses on level $(0, 1)$.
- Hierarchy of deterministic automata on words is strict (M_n languages).
- Over trees $(t : \{0, 1\}^* \rightarrow \Sigma)$ nondeterministic automata are more powerful than deterministic ones.



Alternating automata on trees


● Tree $t : \{0, 1\}^* \rightarrow \Sigma$.

$\mathcal{A} = \langle Q, Q_{\exists}, Q_{\forall}, \Sigma, q^0, \delta : Q \times \Sigma \rightarrow \mathcal{P}(Q \times \{0, 1, \varepsilon\}), \Omega : Q \rightarrow \mathbb{N} \rangle$

+ Q_{\exists}, Q_{\forall} — partition of states

+ Move (q', d) — go to direction d changing state to q' .

● Acceptance game $G_{\mathcal{A}, t}$:

$$(q_0, \varepsilon) \leftarrow \delta(q_0, t(\varepsilon)) = \{(q_1, 0), (q_2, 1)\}$$


Alternating automata on trees

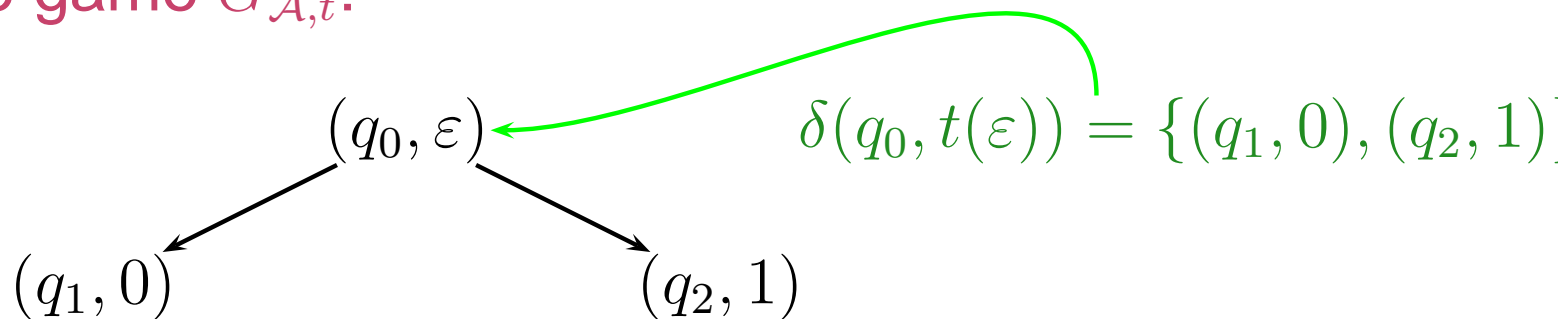
● Tree $t : \{0, 1\}^* \rightarrow \Sigma$.

$\mathcal{A} = \langle Q, Q_{\exists}, Q_{\forall}, \Sigma, q^0, \delta : Q \times \Sigma \rightarrow \mathcal{P}(Q \times \{0, 1, \varepsilon\}), \Omega : Q \rightarrow \mathbb{N} \rangle$

+ Q_{\exists}, Q_{\forall} — partition of states

+ Move (q', d) — go to direction d changing state to q' .

● Acceptance game $G_{\mathcal{A}, t}$:



Alternating automata on trees

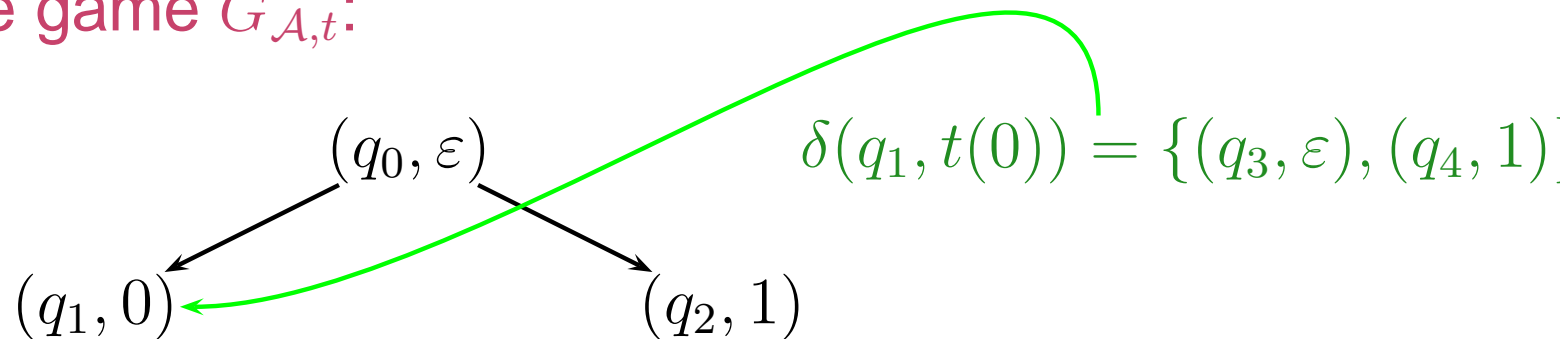
● Tree $t : \{0, 1\}^* \rightarrow \Sigma$.

$\mathcal{A} = \langle Q, Q_{\exists}, Q_{\forall}, \Sigma, q^0, \delta : Q \times \Sigma \rightarrow \mathcal{P}(Q \times \{0, 1, \varepsilon\}), \Omega : Q \rightarrow \mathbb{N} \rangle$

+ Q_{\exists}, Q_{\forall} — partition of states

+ Move (q', d) — go to direction d changing state to q' .

● Acceptance game $G_{\mathcal{A}, t}$:



Alternating automata on trees

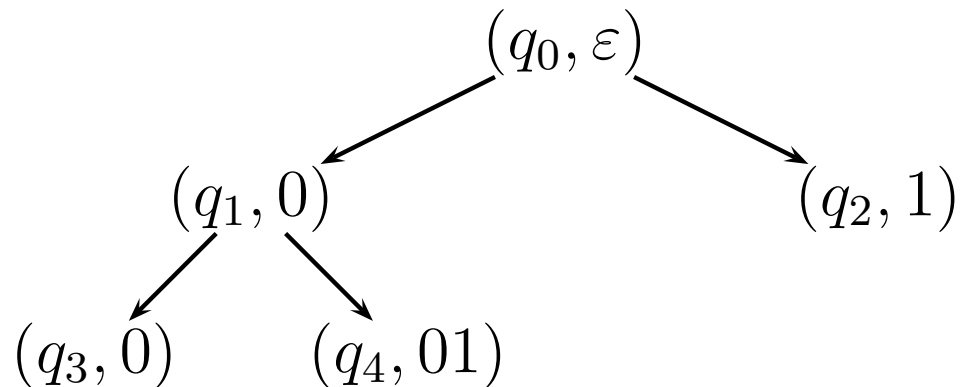
• Tree $t : \{0, 1\}^* \rightarrow \Sigma$.

$\mathcal{A} = \langle Q, Q_{\exists}, Q_{\forall}, \Sigma, q^0, \delta : Q \times \Sigma \rightarrow \mathcal{P}(Q \times \{0, 1, \varepsilon\}), \Omega : Q \rightarrow \mathbb{N} \rangle$

+ Q_{\exists}, Q_{\forall} — partition of states

+ Move (q', d) — go to direction d changing state to q' .

• Acceptance game $G_{\mathcal{A}, t}$:



Alternating automata on trees

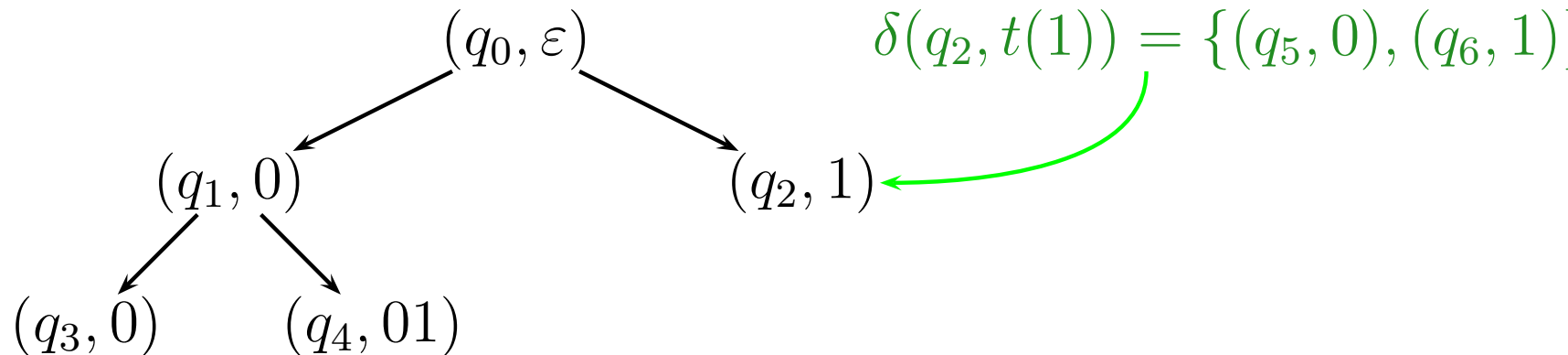
● Tree $t : \{0, 1\}^* \rightarrow \Sigma$.

$\mathcal{A} = \langle Q, Q_{\exists}, Q_{\forall}, \Sigma, q^0, \delta : Q \times \Sigma \rightarrow \mathcal{P}(Q \times \{0, 1, \varepsilon\}), \Omega : Q \rightarrow \mathbb{N} \rangle$

+ Q_{\exists}, Q_{\forall} — partition of states

+ Move (q', d) — go to direction d changing state to q' .

● Acceptance game $G_{\mathcal{A}, t}$:



Alternating automata on trees

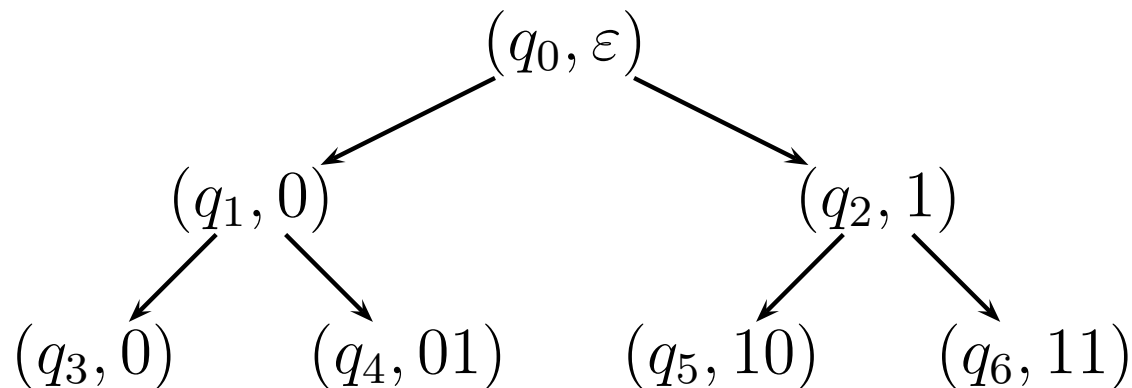
• Tree $t : \{0, 1\}^* \rightarrow \Sigma$.

$\mathcal{A} = \langle Q, Q_{\exists}, Q_{\forall}, \Sigma, q^0, \delta : Q \times \Sigma \rightarrow \mathcal{P}(Q \times \{0, 1, \varepsilon\}), \Omega : Q \rightarrow \mathbb{N} \rangle$

+ Q_{\exists}, Q_{\forall} — partition of states

+ Move (q', d) — go to direction d changing state to q' .

• Acceptance game $G_{\mathcal{A}, t}$:



Alternating automata on trees

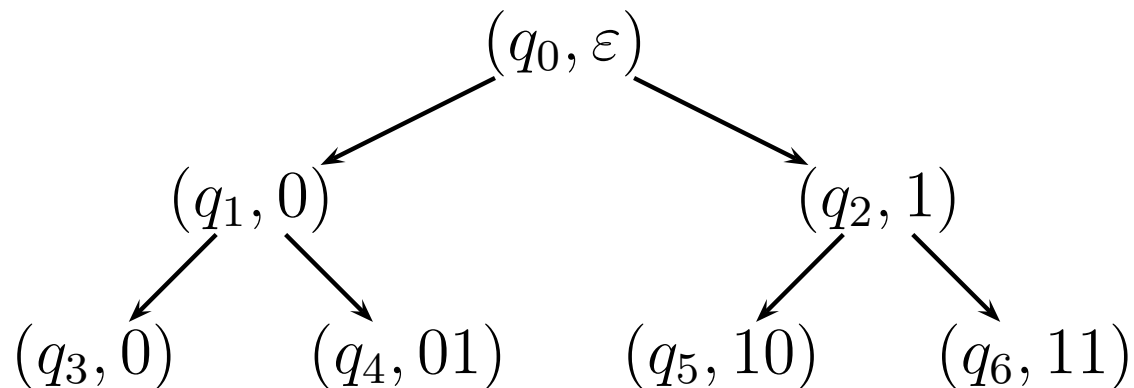
● Tree $t : \{0, 1\}^* \rightarrow \Sigma$.

$\mathcal{A} = \langle Q, Q_{\exists}, Q_{\forall}, \Sigma, q^0, \delta : Q \times \Sigma \rightarrow \mathcal{P}(Q \times \{0, 1, \varepsilon\}), \Omega : Q \rightarrow \mathbb{N} \rangle$

+ Q_{\exists}, Q_{\forall} — partition of states

+ Move (q', d) — go to direction d changing state to q' .

● Acceptance game $G_{\mathcal{A}, t}$:



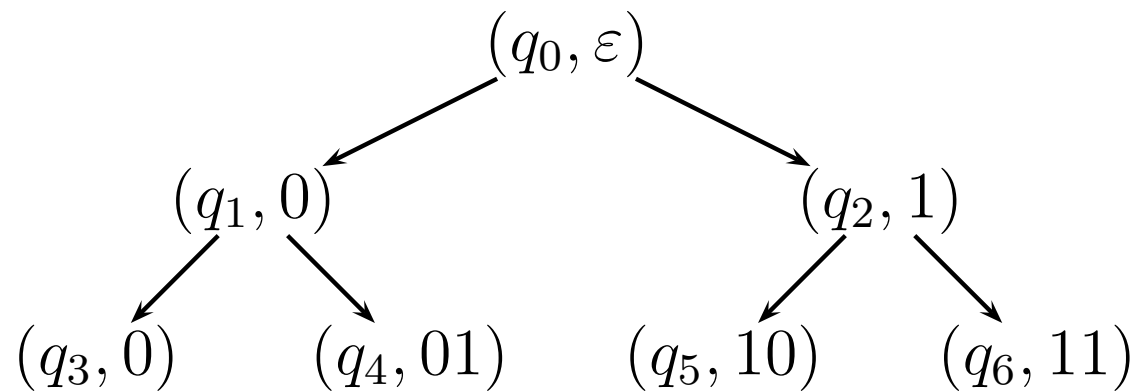
● This game is uniquely determined by \mathcal{A} and t .

• Tree $t : \{0, 1\}^* \rightarrow \Sigma$.

$\mathcal{A} = \langle Q, Q_{\exists}, Q_{\forall}, \Sigma, q^0, \delta : Q \times \Sigma \rightarrow \mathcal{P}(Q \times \{0, 1, \varepsilon\}), \Omega : Q \rightarrow \mathbb{N} \rangle$

+ Q_{\exists}, Q_{\forall} — partition of states

+ Acceptance game $G_{\mathcal{A}, t}$:

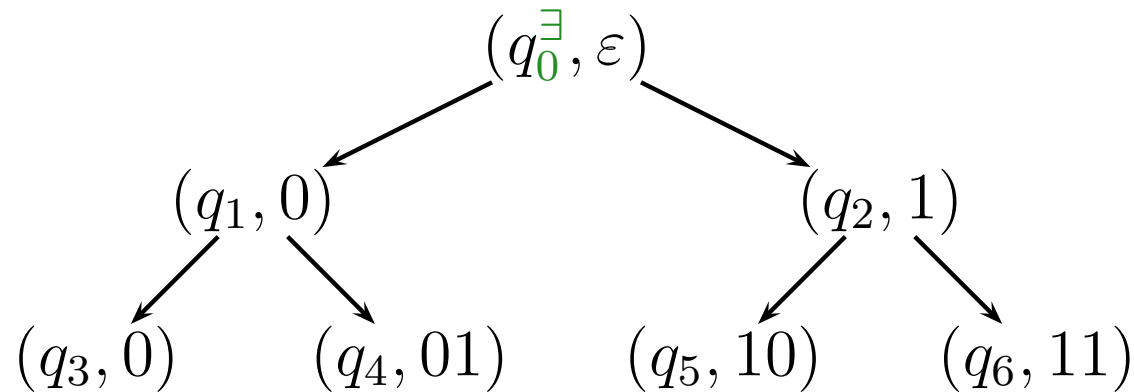


• Tree $t : \{0, 1\}^* \rightarrow \Sigma$.

$\mathcal{A} = \langle Q, Q_{\exists}, Q_{\forall}, \Sigma, q^0, \delta : Q \times \Sigma \rightarrow \mathcal{P}(Q \times \{0, 1, \varepsilon\}), \Omega : Q \rightarrow \mathbb{N} \rangle$

+ Q_{\exists}, Q_{\forall} — partition of states

+ Acceptance game $G_{\mathcal{A}, t}$:

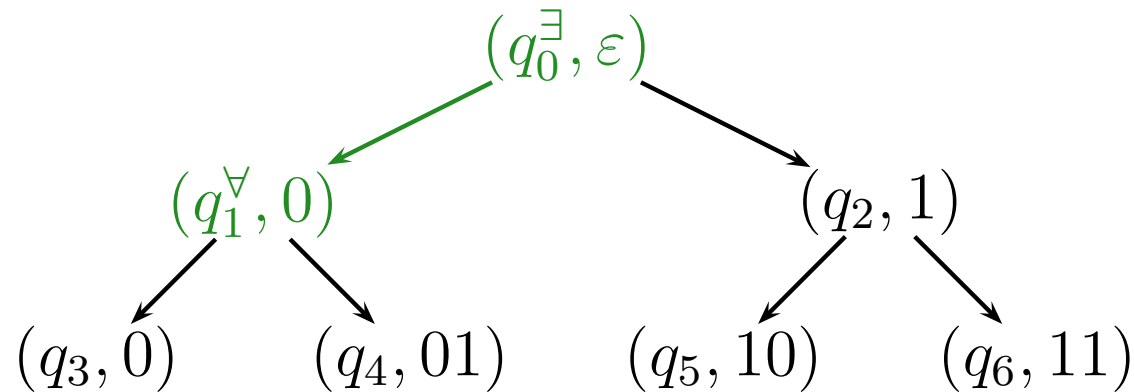


• Tree $t : \{0, 1\}^* \rightarrow \Sigma$.

$\mathcal{A} = \langle Q, Q_{\exists}, Q_{\forall}, \Sigma, q^0, \delta : Q \times \Sigma \rightarrow \mathcal{P}(Q \times \{0, 1, \varepsilon\}), \Omega : Q \rightarrow \mathbb{N} \rangle$

+ Q_{\exists}, Q_{\forall} — partition of states

+ Acceptance game $G_{\mathcal{A}, t}$:

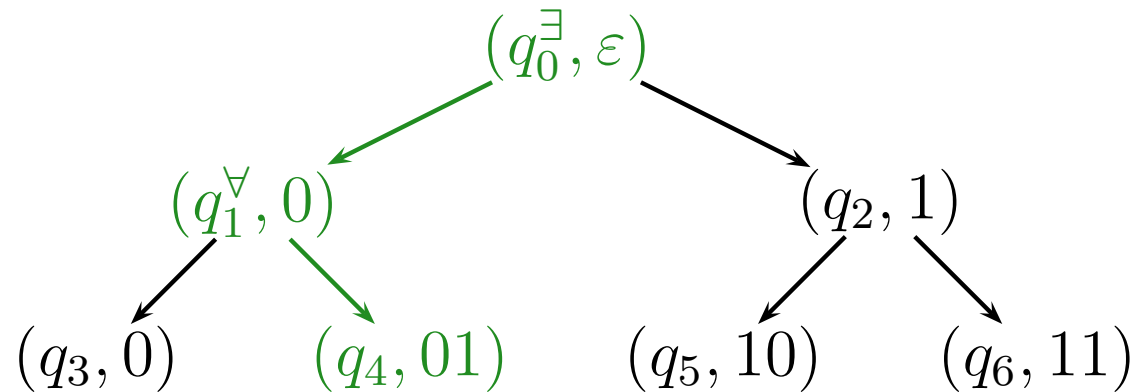


• Tree $t : \{0, 1\}^* \rightarrow \Sigma$.

$\mathcal{A} = \langle Q, Q_{\exists}, Q_{\forall}, \Sigma, q^0, \delta : Q \times \Sigma \rightarrow \mathcal{P}(Q \times \{0, 1, \varepsilon\}), \Omega : Q \rightarrow \mathbb{N} \rangle$

+ Q_{\exists}, Q_{\forall} — partition of states

+ Acceptance game $G_{\mathcal{A}, t}$:

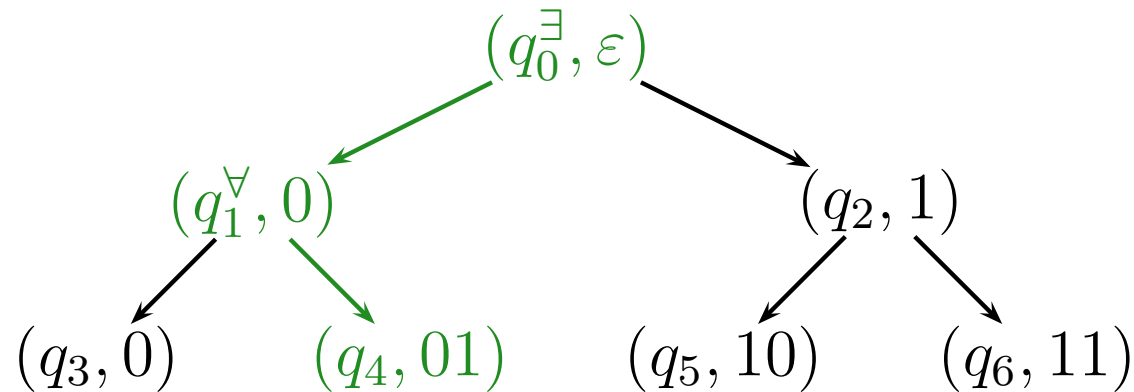


• Tree $t : \{0, 1\}^* \rightarrow \Sigma$.

$\mathcal{A} = \langle Q, Q_{\exists}, Q_{\forall}, \Sigma, q^0, \delta : Q \times \Sigma \rightarrow \mathcal{P}(Q \times \{0, 1, \varepsilon\}), \Omega : Q \rightarrow \mathbb{N} \rangle$

+ Q_{\exists}, Q_{\forall} — partition of states

+ Acceptance game $G_{\mathcal{A}, t}$:



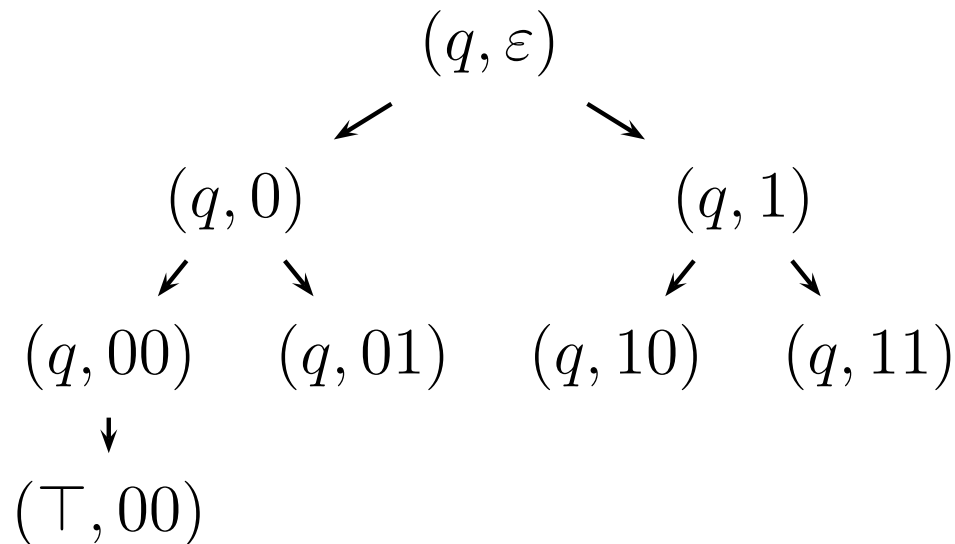
+ Eve wins a play $(q_0, v_0), (q_1, v_1), \dots$ if
 $\liminf_{n \rightarrow \infty} \Omega(q_i)$ is even

Def: $t \in L(\mathcal{A})$ iff Eve has a winning strategy from (ε, q^0) .

- There is a descendant labeled by b :

$$(q, a) \mapsto \{(q, 0), (q, 1)\} \quad (q, b) \mapsto \{(\top, \varepsilon)\}$$

Both states are existential.



- There is b on every path of the tree:

The same automaton but now states are universal.

- Automata hierarchy:

$A(1, i)$ alternating automata with $\Omega : Q \rightarrow \{1, \dots, i\}$

$A(0, i - 1)$ alternating automata with $\Omega : Q \rightarrow \{0, \dots, i - 1\}$

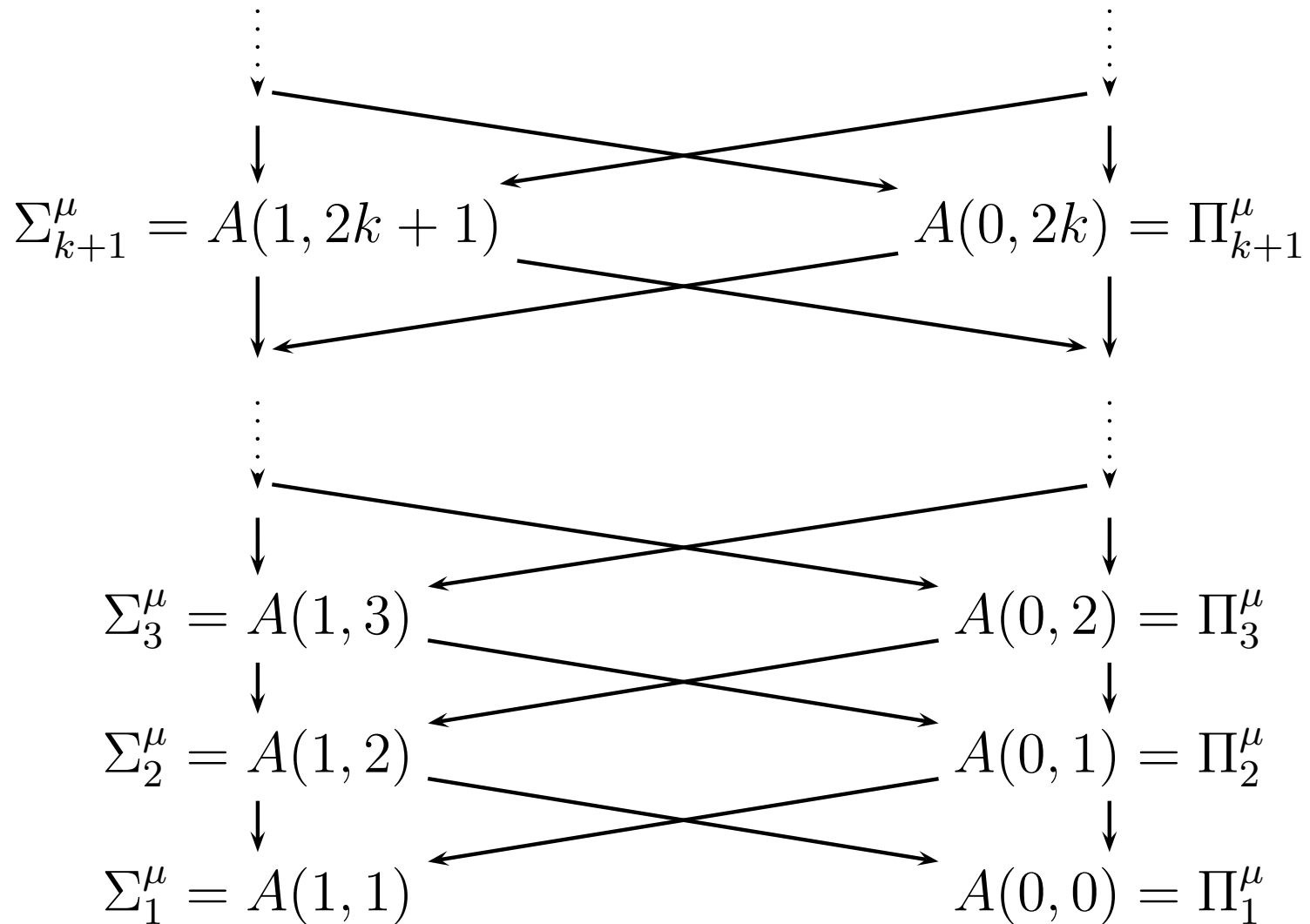
- Languages definable by Σ_i^μ formulas are exactly the languages recognizable by $A(1, i)$ automata.

Thm [Niwiński]: $\Sigma_i^\mu = A(1, i), \quad \Pi_i^\mu = A(0, i - 1).$

Rem: $\Pi_2^\mu = \nu\mu = A(0, 1) = \text{Büchi}$

Thm [Bradfield, Arnold]: The μ -calculus hierarchy is infinite.

Rem: Over words the hierarchy collapses on $\nu\mu$ -level.



- Along with the hierarchy of alternating automata:

$$\mathcal{A} = \langle Q, Q_{\exists}, Q_{\forall}, \Sigma, q^0, \delta : Q \times \Sigma \rightarrow \mathcal{P}(Q \times \{0, 1, \varepsilon\}), \Omega : Q \rightarrow \mathbb{N} \rangle$$

- There is also an index hierarchy of **nondeterministic automata**:

$$\mathcal{A} = \langle Q = Q_{\exists}, Q_{\forall}, \Sigma, q^0, \delta : Q \times \Sigma \rightarrow \mathcal{P}(Q \times Q), \Omega : Q \rightarrow \mathbb{N} \rangle$$

$$N(0, i) \quad \text{or} \quad N(1, i)$$

- And of **deterministic automata**:

$$\mathcal{A} = \langle Q = Q_{\exists}, Q_{\forall}, \Sigma, q^0, \delta : Q \times \Sigma \rightarrow Q \times Q, \Omega : Q \rightarrow \mathbb{N} \rangle$$

$$D(0, i) \quad \text{or} \quad D(1, i)$$

Thm [Niwiński & W.]: The levels $A(0, 0)$ and $A(1, 1)$ of the hierarchy are decidable.

Thm [Niwiński & W.]: The level $A(0, 1) \cap A(1, 2)$ is decidable.

Thm [Urbański]: The level $A(0, 1) = N(0, 1)$ is decidable for deterministic automata.

Thm [Niwiński & W.]: All levels of N hierarchy are decidable for deterministic automata.

Rem: For deterministic automata A hierarchy collapses on $A(1, 2)$ level.

Rem: Deterministic automata are weaker than nondeterministic. It is decidable if a regular language is accepted by a deterministic automaton.

Part III

Games on infinite graphs

- Games on pushdown graphs.
- Higher order pushdowns.
- Recursive program schemes.

Graphs of pushdown machines

- Pushdown machine (deterministic):

$\langle Q, \Sigma, \Gamma, q_0 \in Q, \delta : Q \times \Sigma \times \Gamma \rightarrow Q \times \{pop, push(z) : z \in \Gamma\}, F \subseteq Q \rangle$.

- Configuration: $(q, w) \in Q \times \Gamma^*$.

- Configuration graph

- nodes: configurations

- transitions:

$(q, zw) \rightarrow (q', w)$ if there is $a \in \Sigma$ and $\delta(q, a, z) = (q', pop)$

$(q, zw) \rightarrow (q', z'zw)$ if there is $a \in \Sigma$ and $\delta(q, a, z) = (q', push(z'))$

- **Rem:** The input alphabet and accepting states do not play any role. Determinism is also not important.

Pushdown system: $P = (Q, \Gamma, \Delta)$

Rewrite rules: $\Delta \subseteq Q \times \Gamma \times Q \times (\{\varepsilon\} \cup \Gamma^2)$
 $qz \rightsquigarrow q' \quad qz \rightsquigarrow q'z'z$

Pushdown graph: $G(P)$

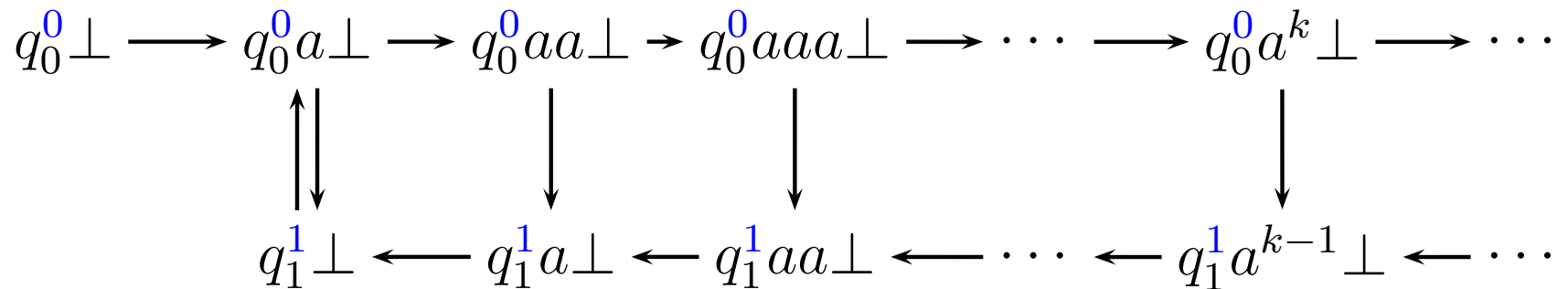
Vertices: $Q \times \Gamma^*$

Edges: $qw \rightarrow q'w'$ according to the rules **applied to prefixes**.

● q_0 is always the initial state and \perp is the initial stack symbol.

TM graph: rules of the form $aqb \rightsquigarrow q'a'b$ or $aqb \rightsquigarrow ab'q'$ without restrictions on the place of application.

Pushdown game: an example



- We have that:

q_0 is a vertex of Adam and q_1 of Eve;

$\Omega(q_0) = 0$ and $\Omega(q_1) = 1$.

- Eve has a winning strategy in this game.

● The game solving problem: Given P with a partition (Q_E, Q_A) of states, and a function $\Omega : Q \rightarrow \mathbb{N}$ decide who has a winning strategy from the initial vertex of $G(P)$.

Thm: The problem of solving parity pushdown games is EXPTIME-complete.

● EF logic

$$p \mid \neg\alpha \mid \alpha \wedge \beta \mid \exists\langle \rangle a \mid \exists F\alpha$$

!No $\exists G\alpha$!

● CTL

$$\text{EF} + (\exists(\alpha_1 U \alpha_2) \mid \exists\neg(\alpha_1 U \alpha_2))$$

● $M, v \models \exists F\alpha$ iff there is v' reachable from v with $M, v' \models \alpha$

● $M, v \models \exists G\alpha$ iff there is a path from v s.t. for every v' on it we have $M, v' \models \alpha$.

● μ -calculus

$$P \mid \neg P \mid X \mid \alpha \mid \alpha \vee \beta \mid \alpha \wedge \beta \mid \langle a \rangle \alpha \mid [a] \alpha \mid \mu X. \alpha \mid \nu X. \alpha$$

Thm: Model checking problem for the μ -calculus is EXPTIME-complete.

Thm: The model checking problem for EF-logic is in PSPACE. It is PSPACE-hard [Bouajjani, Esparza, Maler]

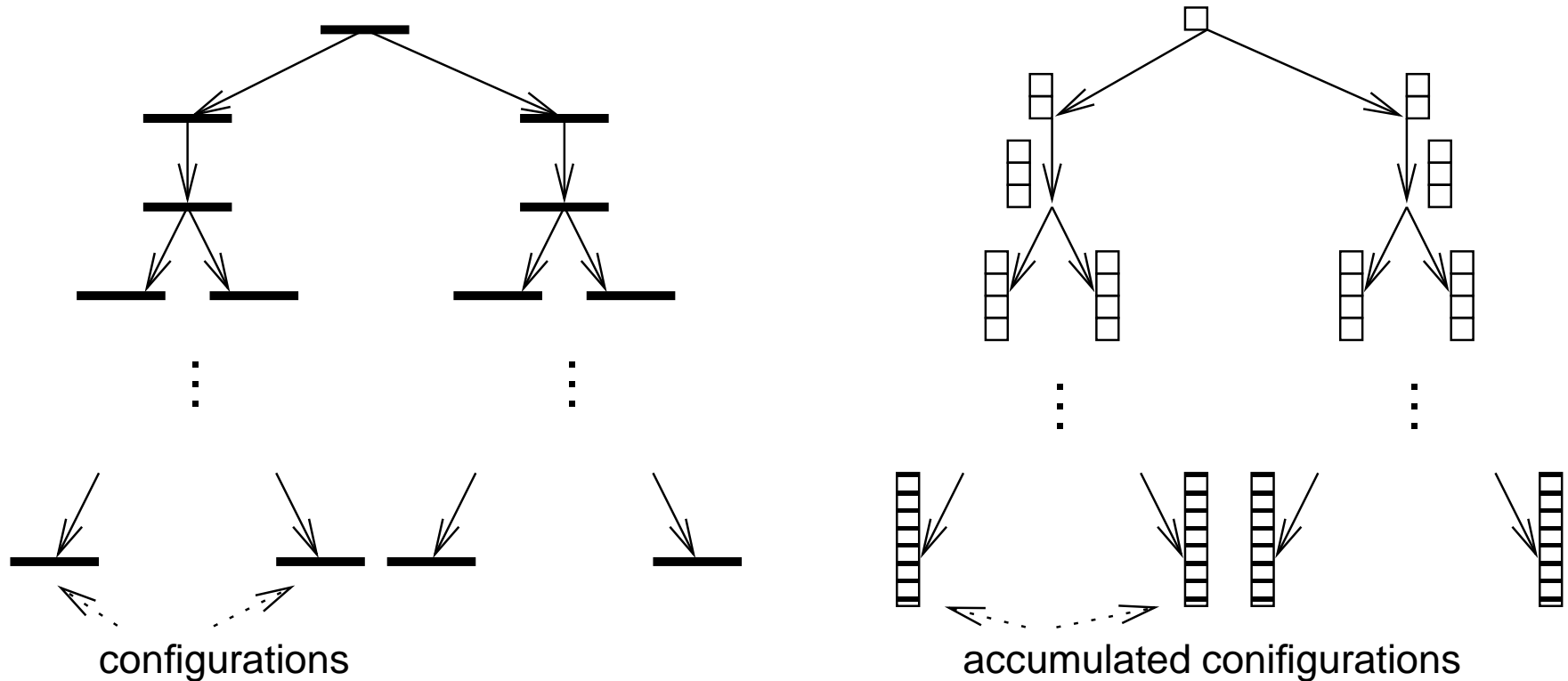
Thm: The same problem for CTL formulas is EXPTIME-complete.

Rem: The problem is with $\exists \alpha U \beta$.

Alt reachability: EXPTIME-hard

- Take $\text{ASPACE}(n)$ machine M and input w . Construct P_w :

P_w has alt. reach. prop. iff $w \in L(M)$.



- How to check that a sequence of accumulated configurations is correct?

Checking consecutive configurations

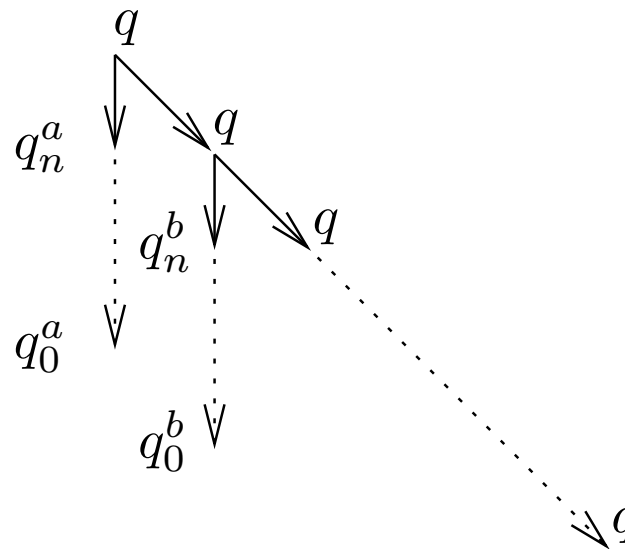
- Simpler problem: Given a word w of length n decide if the stack is of the form $w^k \perp$ for some k .

$$qa \rightarrow q, q_n^a$$

$$q_n^a b \rightarrow q_{n-1}^a$$

$$q \perp \rightarrow q_F$$

$$q_0^a a \rightarrow q_F$$



Summary of pushdown model checking

μ -calc	EXPTIME-compl
Alt reach	EXPTIME-compl
CTL	EXPTIME-compl
LTL	EXPTIME-compl
EF	PSPACE-compl
reach	PTIME

● **2-stack** is a stack of stacks. There are operations on top-most stack and of copying the top-most stack.

● A system where all paths are of the form $q_1^k q_2^k q_3^k$

$q_1[a] \rightarrow q_1[aa] \rightarrow \dots \rightarrow q_1[a^k] \rightarrow$

$q_2[a^k] \rightarrow q_2[a^{k-1}][a^k] \rightarrow \dots \rightarrow q_2[a][aa] \dots [a^k] \rightarrow$

$q_3[a][aa] \dots [a^k] \rightarrow q_3[aa] \dots [a^k] \rightarrow \dots \rightarrow q_3[a^k] \rightarrow q_3 \perp$

● 2-stack gives additional power. If considered as an accepting device 2-store automaton would recognize $\{a^k b^k c^k : k \in \mathbb{N}\}$.

2nd order pushdown system

- 2nd order pushdown system $\langle Q, \Gamma, Inst \rangle$
- A configuration is a sequence $[s_1][s_2] \dots [s_k] \in (\Gamma^*)^*$
- Instructions:

$$q'[w]v \xrightarrow{push_1(a)} q'[aw]v$$

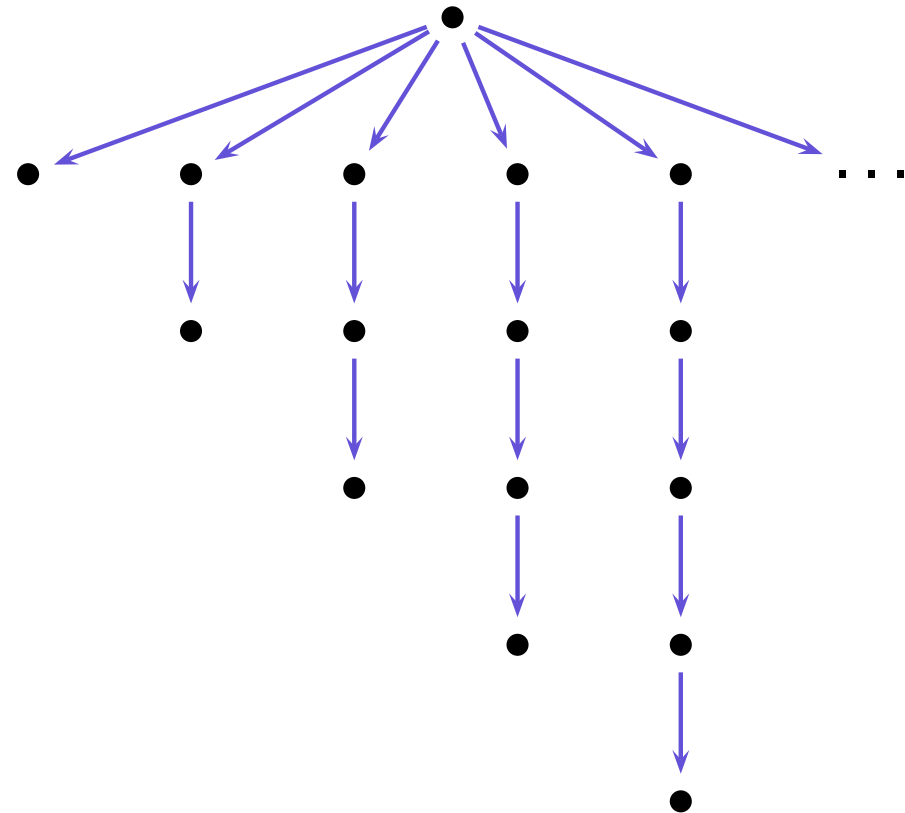
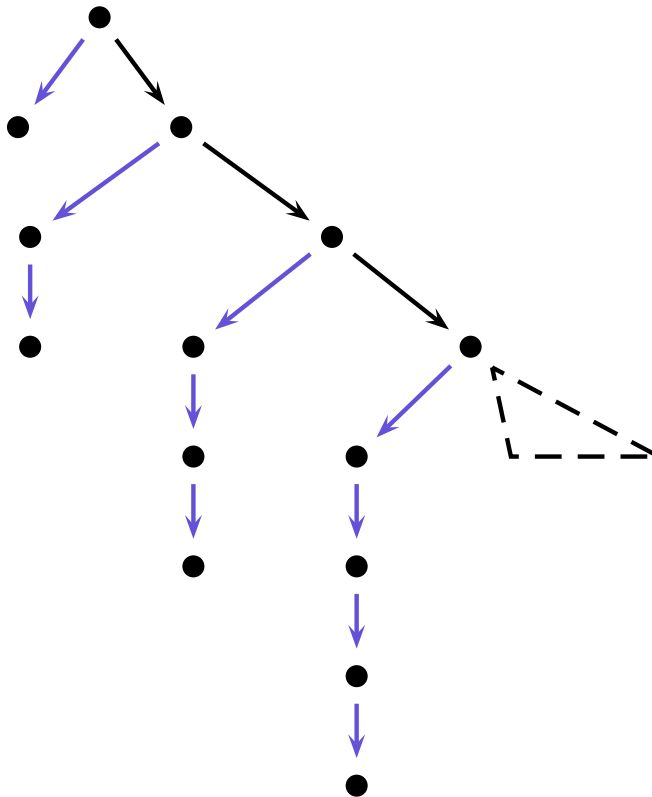
$$q'[aw]v \xrightarrow{pop_1} q'[w]v$$

$$q'[w]v \xrightarrow{push_2} q'[w][w]v$$

$$q'[w]v \xrightarrow{pop_2} q'v$$

- Similarly for higher orders: stacks become of type $((\Gamma^*)^{*\dots*})$

- The n -th level of Caucal hierarchy consists of ε -closures of n -level pushdown graphs.



● The n -th level of **Caucal hierarchy** consists of ε -closures of n -level pushdown graphs.

Rem: 1-st level graphs are prefix-recognizable graphs.

Thm [Carayol & Wöhrle]: n -th level Caucal graphs are precisely those that are MSOL interpretable in n -tree.

- A 2-tree over the set Γ is $\langle (\Gamma^*)^*, \{succ_z : z \in \Gamma\}, succ_2 \rangle$ where
- $(v[w], v[wz]) \in succ_z$
 - $(v[w], v[w][w]) \in succ_2$

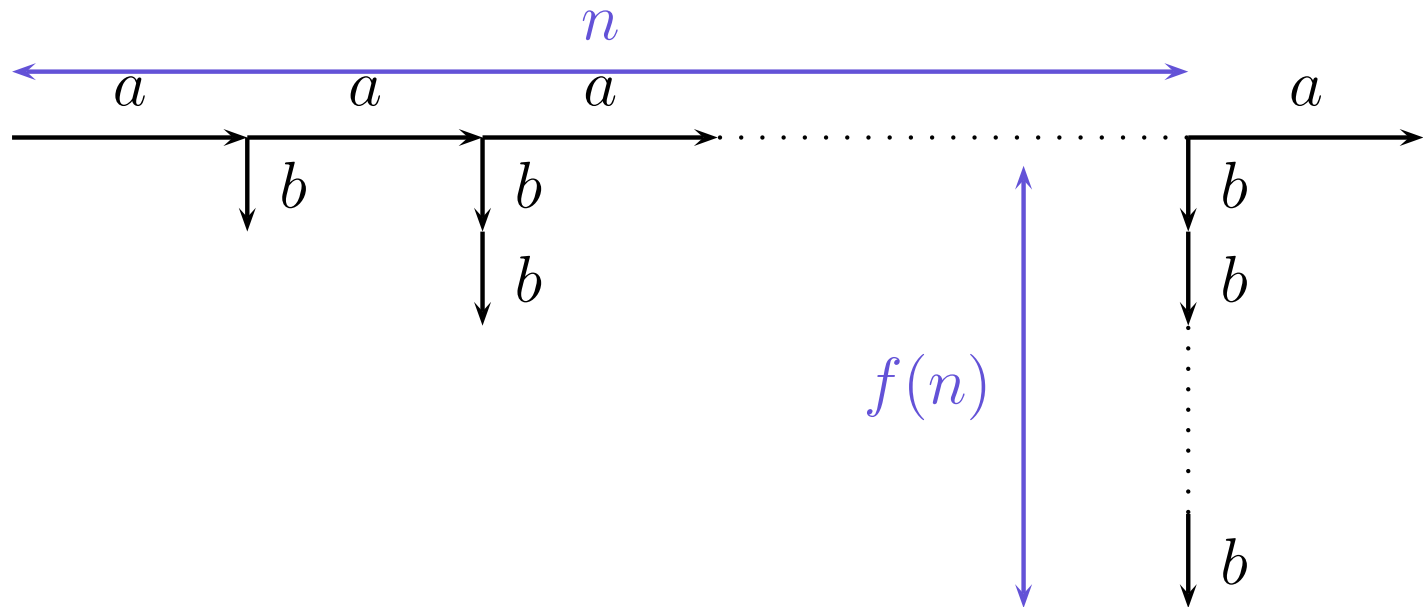
Thm[Engelfriet]: Alternating reachability problem for n -th level pushdown-graphs is n -EXPTIME-hard.

Thm[Cachat]: The μ -calculus model-checking problem for n -th level pushdown graphs is solvable in n -EXPTIME.

Rem: Not clear what is the complexity for other logics of programs.

Infiniteness of Caucal hierarchy

- For a function $f : \mathbb{N} \rightarrow \mathbb{N}$ we define a tree T_f :



- Define $Tower_0(n) = n$ and $Tower_{k+1}(n) = 2^{Tower_k(n)}$. Moreover $Tower_\omega(n) = Tower_n(n)$.

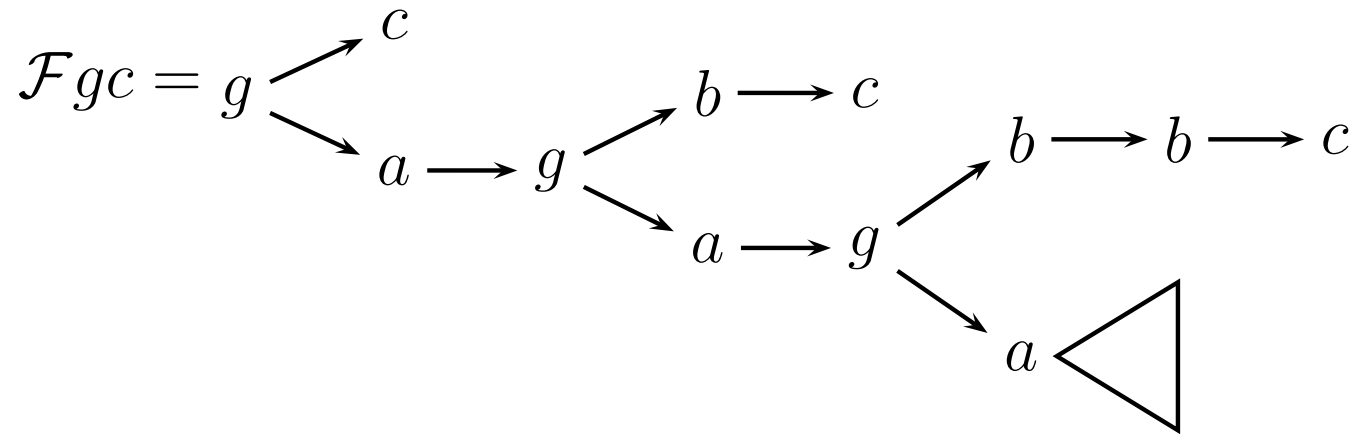
- T_{Tower_k} tree is an k -level graph that is not $(k - 1)$ -level.

- T_{Tower_ω} tree is not in the Caucal hierarchy. It has decidable MSO theory (follows from morphic predicates of Carton and Thomas).

- We do not know how to decide the levels of the hierarchy.

- $\mathcal{F}x \Rightarrow g (a(\mathcal{F}(bx))) x$

- Equations generate infinite trees (infinite terms).



Thm[Courcelle]: The meanings of (1st order) recursive schemes \equiv pushdown graphs.

- Example of a second order scheme:

$$\mathcal{F}\psi x \Rightarrow f (\mathcal{F}(\mathcal{D}\psi)x) (\psi x)$$

$$\mathcal{D}\psi y \Rightarrow \psi(\psi y)$$

Thm[Knapik, Niwiński, Urzyczyn, W.]: Meanings of 2nd order schemes \equiv trees of 2nd order pushdown systems with panic.

Thm[Knapik, Niwiński, Urzyczyn, W.]: For every 2nd order scheme, the tree defined by this scheme has decidable MSO theory.

Question: Does panic increase expressive power?

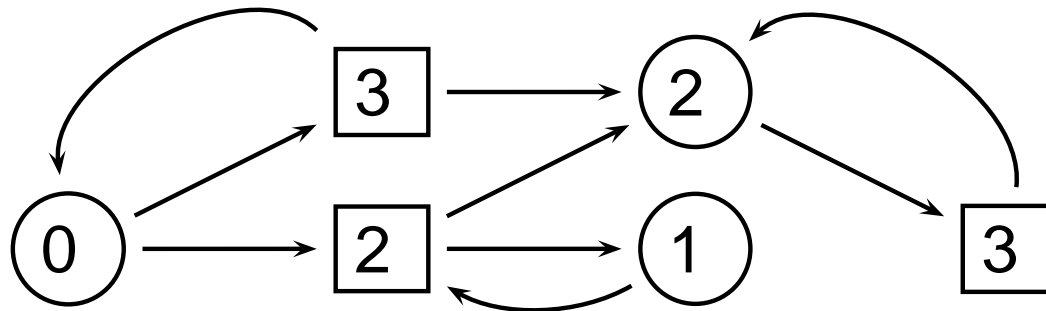
Question: Decidability of the equivalence between schemes.

Part IV

Complicating wining conditions

- Stack height conditions.
- Stack height and parity conditions.
- Visibly pushdown conditions.
- Quantitative conditions.

In contrast to the finite-state case, there are now natural winning conditions which are no more monadic second-order definable and occur at higher Borel levels than $\mathcal{B}(\Sigma_2^0)$. [Thomas, STACS'95]



$$\mathcal{G} = \langle V_E, V_A, R, \lambda : V \rightarrow C, Acc \subseteq C^\omega \rangle$$

- $ht(v)$ the height of the stack at position v

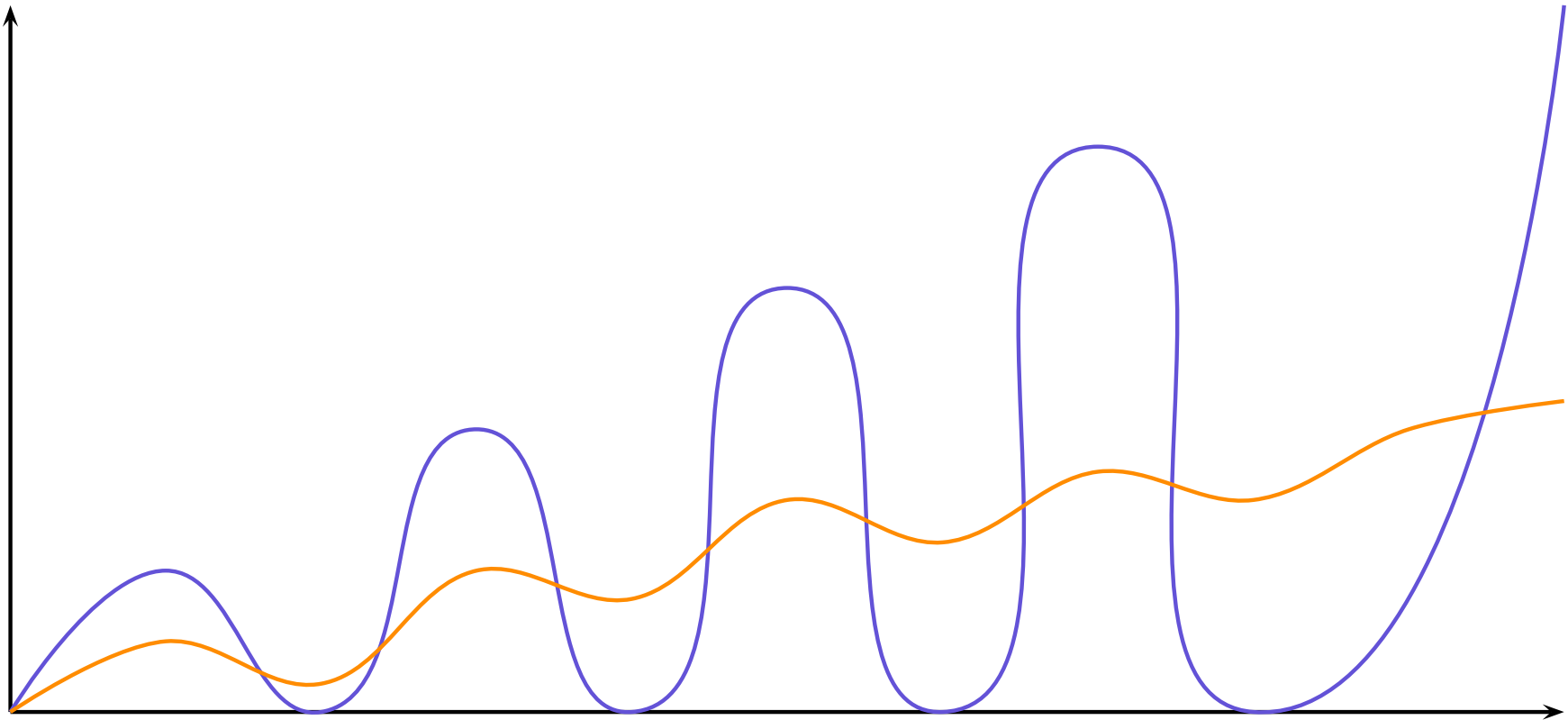
Unboundedness conditions

● **Unboundedness condition** The height of the stack is

unbounded: $\vec{v} \in Acc$ iff $\forall n. \exists i. ht(v_i) > n$.

● **Strict unboundedness condition** The liminf of the stack height is

infinity: $\vec{v} \in Acc$ iff $\forall n. \exists i. \forall j > i. ht(v_j) > n$.

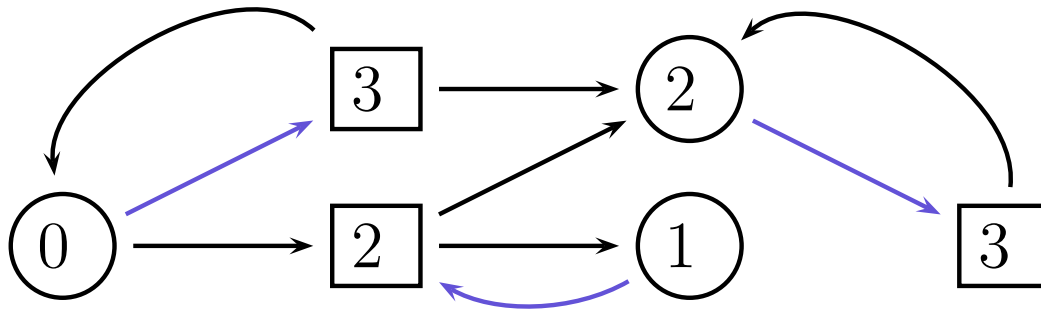


Memoryless strategies for unboundedness

$$\mathcal{G} = \langle V_E, V_A, R, \lambda : V \rightarrow C, Acc \subseteq C^\omega \rangle$$

● **Strategy** for player 0 is $\sigma : V^* \times V_0 \rightarrow V$ such that $\sigma(\vec{v}v_0) \in R(v_0)$

● A strategy σ for Eve is **winning from** v if all plays from v respecting the strategy are winning for Eve.



● **Positional/memoryless strategy** for Eve is a function $\sigma : V_0 \rightarrow V$ such that $\sigma(v) \in R(v)$.

Thm: In the game with unboundedness conditions Eve has a memoryless winning strategy.

Unboundedness \equiv strict unboundedness

● If there is a winning strategy for Eve in unboundedness game there is a memoryless winning strategy. This strategy σ never visits a vertex twice.

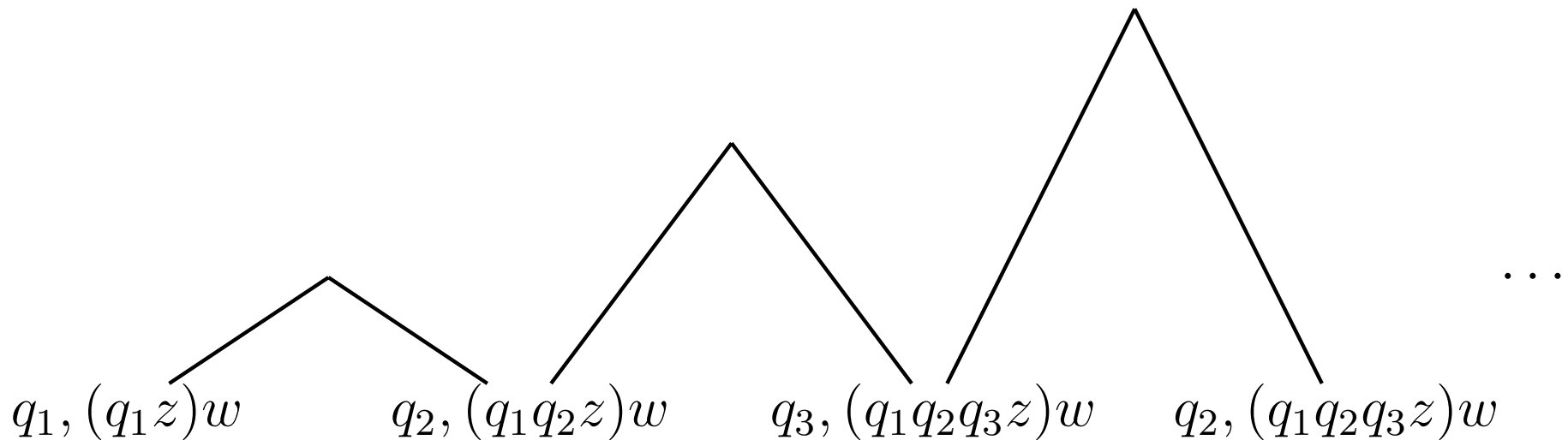
● If σ is winning for unboundedness then it is also winning for strict unboundedness.

Cor: The game is winning for unboundedness iff it is winning for strict unboundedness.

Rem: The same for disjunction of unboundedness and a parity condition.

Reduction: unboundedness to safety

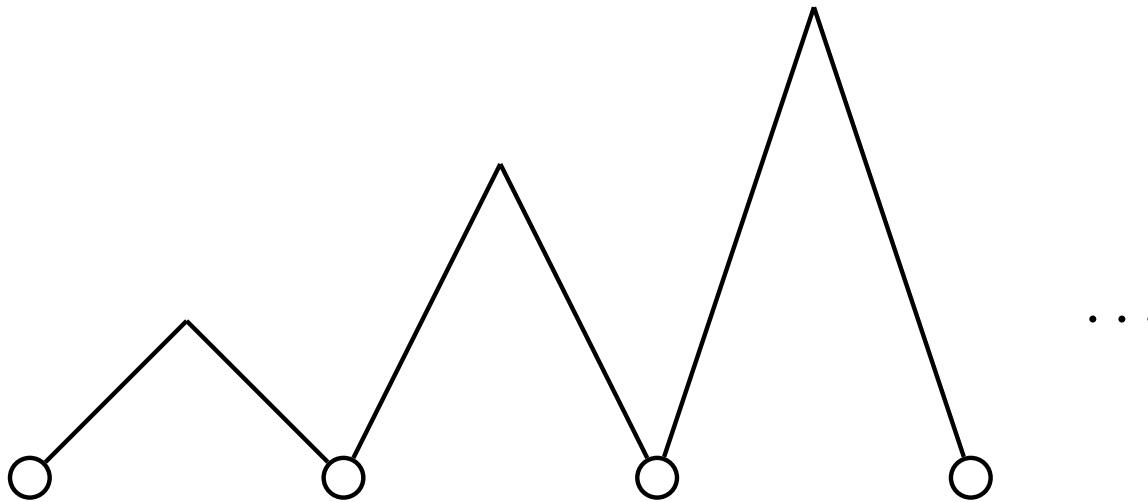
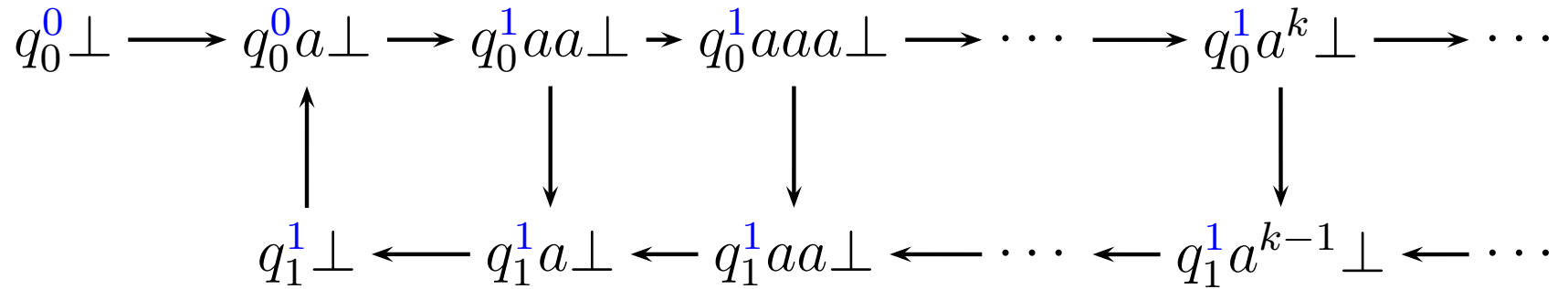
- There is a winning strategy for Eve in unboundedness game iff there is a memoryless winning strategy.
- Modify a pushdown system P in such a way that states seen with the current stack contents are recorded in the top of the stack.



- Make a position losing if the current state is the same as recorded on in the top of the stack

unboundedness in $P \equiv$ safety in P'

Unboundedness and parity



- Infinite memory is need to win.

Cor: Winning conditions that are unions of explosion and parity conditions admit memoryless strategies. Intersection of Büchi and explosion conditions may need infinite memory.

Thm[Cachat & Duparc & Thomas]: Strict unboundedness condition is Σ_3 -complete in the Borel hierarchy.

Thm[Cachat & Duparc & Thomas, Bouquet & Serre & W., Gimbert]: Games with winning conditions that are combinations of parity and explosion conditions can be solved in EXPTIME.

Thm[Serre]: For every finite level of the Borel hierarchy there is a winning condition complete for this level and such that games with this conditions are decidable.

- Winning conditions can be defined by pushdown automata: the sequence of visited states must be accepted by the automaton
- Games with such winning conditions are undecidable (universality of a context-free language).
- Visibly pushdown automata: input letters determine stack actions.

Thm [Löding, Madhusudan, Serre]: Pushdown games with winning conditions given by visibly pushdown automata are decidable. Strict explosion condition is expressible in this framework.

- Mean pay-off games: $\limsup_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n \Omega(v_i)$
- The result of the play is a real number and not win/lose. The objective is to maximize/minimize the value.

Thm [Ehrenfeucht & Mycielski]: Finite mean pay-off games are solvable.

- Priority mean pay-off games:
 - vertices labeled by $(m, r) \in \mathbb{N} \times \mathbb{R}$
 - For the sequence $(m_0, r_0), (m_1, r_1), \dots$ calculate $k = \liminf_{n \rightarrow \infty} m_n$.
 - Let i_0, i_1, \dots the positions j where $m_j = k$.
 - The result is $\limsup_{n \rightarrow \infty} \frac{1}{n} \sum_{i=1}^n r_{i_n}$.

Thm [Gimbert & Zielonka]: Finite priority mean pay-off games are solvable.

- The main motivation is the verification problem.

$$\begin{array}{c} \mathcal{M} \stackrel{?}{\models} \alpha \\ \updownarrow \\ G \leftarrow \text{parity game} \end{array}$$

- The research splits into two tracks:
 - Study of the properties of the logic.
algorithmic and expressive properties
 - Study of games.
game presentations and winning conditions.