A landscape with games in the background

Igor Walukiewicz
Bordeaux University

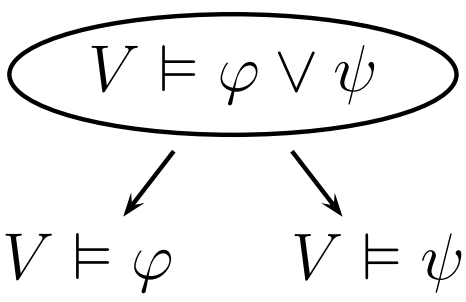
Plan

- Basic games.
- Games behind model-checking.
- Games behind synthesis.
- Extensions of the basic game model.
- Distributed synthesis.

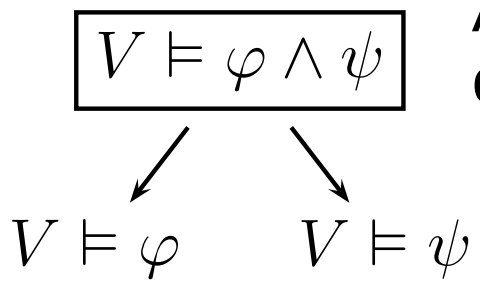
Abstraction

Propositional logic (model checking)

- $\mathrm{Prop} \mid \neg\mathrm{Prop} \mid \varphi \vee \psi \mid \varphi \wedge \psi$
- Valuation: $V : \mathrm{Prop} \to \{0, 1\}$
- Model checking rules



Eve chooses



Adam chooses

$V \vDash P$ Eve wins if $V(P) = 1$.
$V \vDash \neg P$ Eve wins if $V(P) = 0$.

- Eve has a winning strategy from $V \vDash \varphi$ iff φ is true in V.

Propositional logic (satisfiability)

- We want to design a game for satisfiability checking.
- We work with sets of formulas.

$$\frac{\varphi \vee \psi, \Gamma}{\varphi, \Gamma} \qquad \frac{\varphi \vee \psi, \Gamma}{\psi, \Gamma} \qquad \text{Eve chooses}$$

$$\frac{\varphi \wedge \psi, \Gamma}{\varphi, \psi, \Gamma} \qquad \text{Adam chooses}$$

if Γ-irreducible then Eve wins if no $P, \neg P \in \Gamma$.

- Eve has a winning strategy from $\{\varphi\}$ iff φ is satisfiable.

$$P_1, \neg P_2, P_3$$

Propositional logic (satisfiability)

- We want to design a game for satisfiability checking.
- We work with sets of formulas.

$$\frac{\varphi \vee \psi, \Gamma}{\varphi, \Gamma} \qquad \frac{\varphi \vee \psi, \Gamma}{\psi, \Gamma}$$ Eve chooses

$$\frac{\varphi \wedge \psi, \Gamma}{\varphi, \psi, \Gamma}$$ Adam chooses

if Γ-irreducible then Eve wins if no $P, \neg P \in \Gamma$.

- Eve has a winning strategy from $\{\varphi\}$ iff φ is satisfiable.
- Every model of φ can be obtained from a winning strategy in the satisfiability game for $\{\varphi\}$.

Remarks

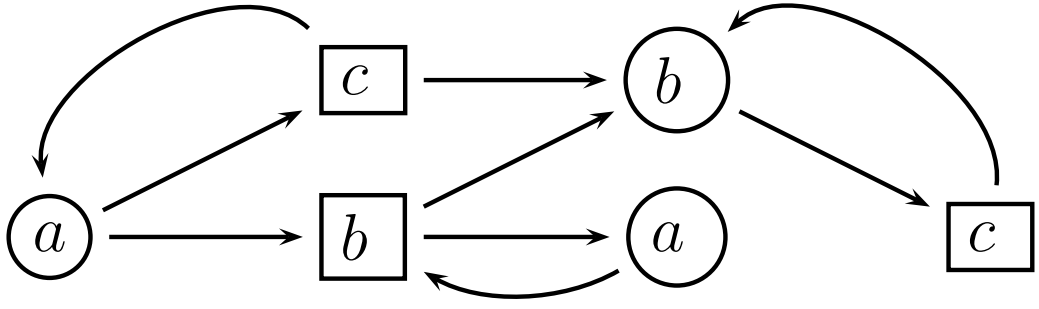
- In the satisfiability game Adam has nothing to say.
  (This is a peculiarity of the simple case).
- Satisfiability games are related to synthesis.
- MC games are related to model-checking.
- In the MC game we work with formulas while in the satisfiability game we work with sets of formulas. (Boolean algebra).
- Satisfiability games are constructed from MC games by a kind of power-set construction.
  (A position in sat game is like a set of positions in MC game)

Part Ib

- Games: basic definitions.
- Games behind model-checking.
- Games behind synthesis.
- Extensions of the basic game model.
- Distributed synthesis.

Path forming games

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

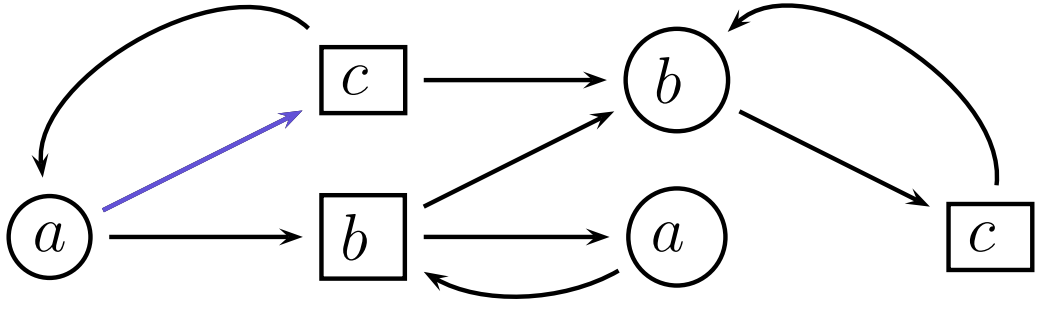


- Eve wins a play if the labeling of it is in Acc.

  (There is an edge from every node.)

Path forming games

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

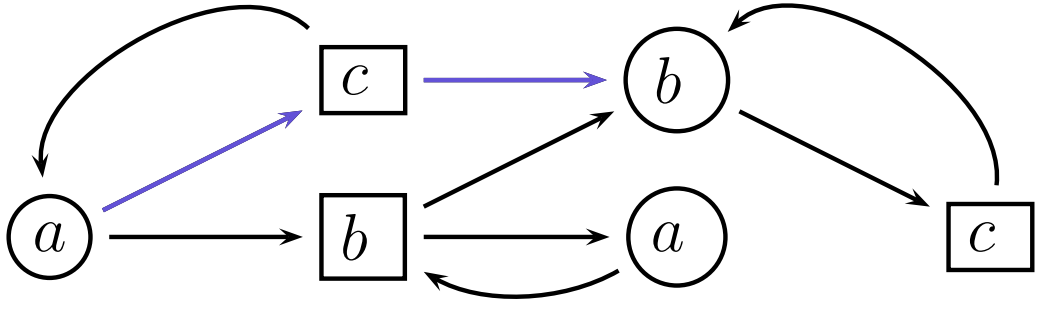


- Eve wins a play if the labeling of it is in Acc.

  (There is an edge from every node.)

Path forming games

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

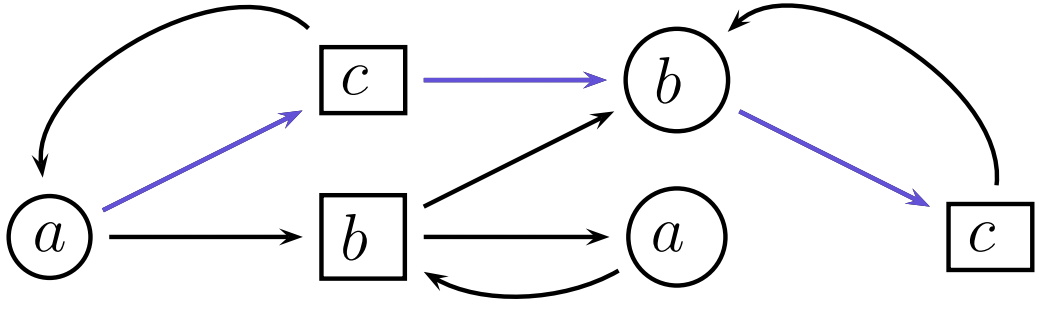


- Eve wins a play if the labeling of it is in Acc.

  (There is an edge from every node.)

Path forming games

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

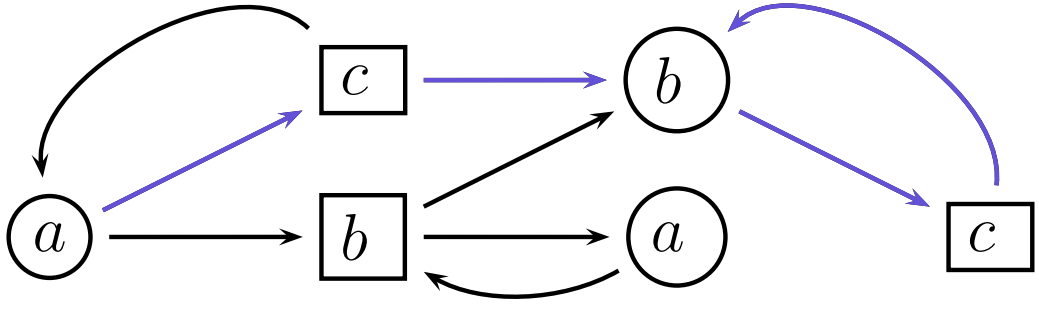


- Eve wins a play if the labeling of it is in Acc.

  (There is an edge from every node.)

Path forming games

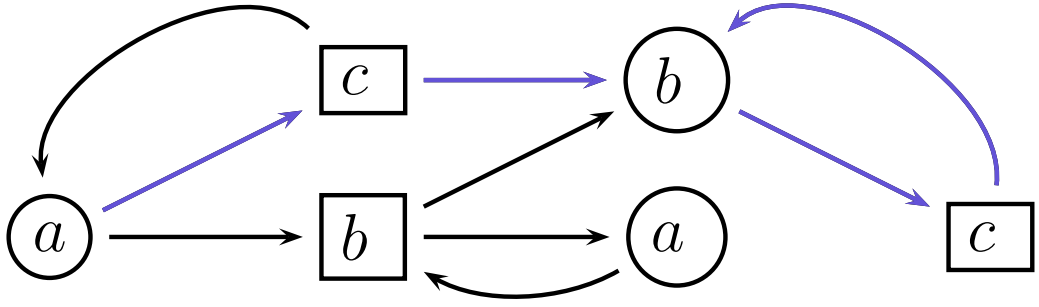
$$\mathcal{G} = \langle V_E,\ V_A,\ R,\ \lambda : V \to C,\ Acc \subseteq C^\omega \rangle$$



- Eve wins a play if the labeling of it is in Acc.

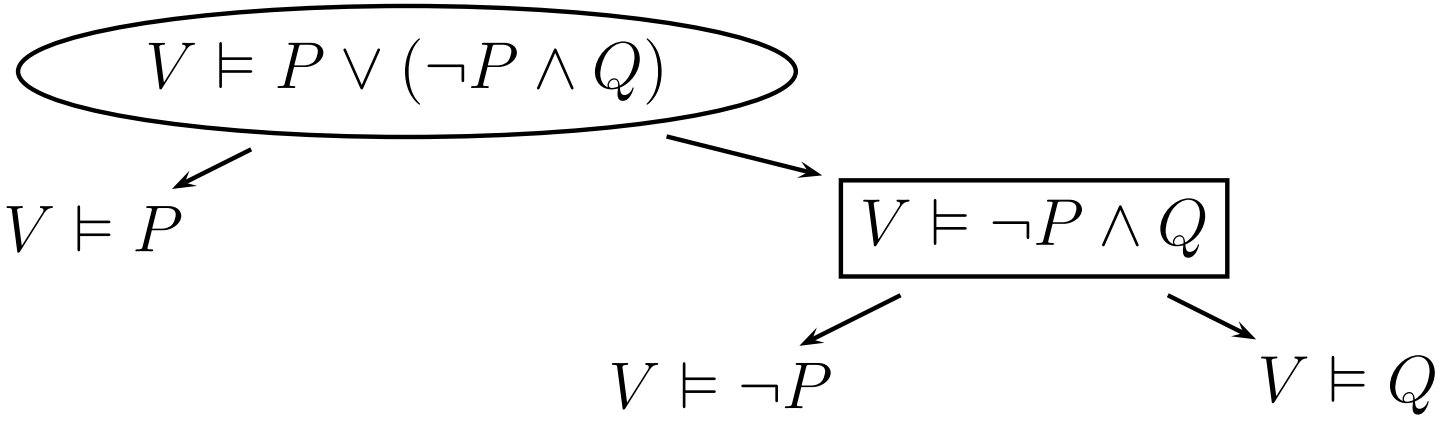
  (There is an edge from every node.)

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



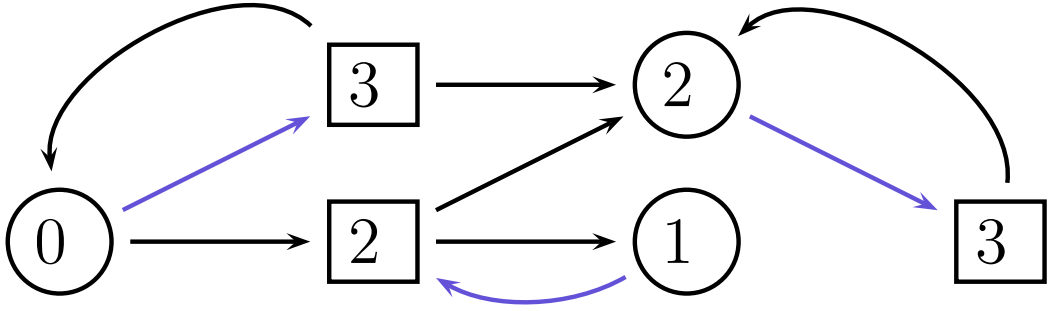
- Eve wins a play if the labeling of it is in Acc.

  (There is an edge from every node.)



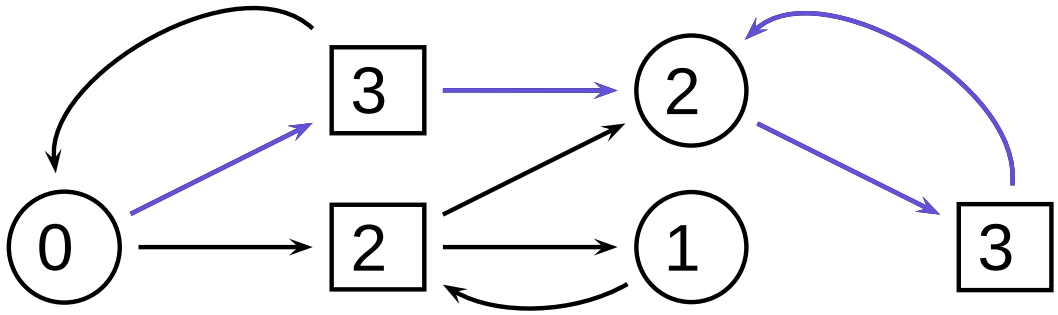
$$\mathcal{G} = \langle V_E,\ V_A,\ R,\ \lambda : V \to C,\ Acc \subseteq C^\omega \rangle$$

- Strategy for Eve is $\sigma : V^* \times V_E \to V$ 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 $\sigma : V_E \to V$ such that $\sigma(v) \in R(v)$.

Winning conditions



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

- $\mathrm{Inf}_\lambda(\vec{v})$: the set of colours appearing infinitely often on a path $\vec{v}$.
- Muller condition: given by a partition of $\mathcal{P}(C)$ into $(\mathcal{F}_E, \mathcal{F}_A)$.

$$\vec{v} \in Acc \quad \text{iff} \quad \{\vec{v} : \mathrm{Inf}_\lambda(\vec{v}) \in \mathcal{F}_E\}$$

- Parity condition colours are numbers $\{0, \ldots, d\}$ and:

$$\vec{v} \in Acc \quad \text{iff} \quad \min(\mathrm{Inf}_\lambda(\vec{v})) \text{ is even.}$$

Basic results

Thm: Every game with a Muller winning condition is determined, i.e., from every vertex one of the players has a winning strategy.

Thm: In a parity game a player has a memoryless winning strategy from each of his winning vertices.

Def: To solve a game is to determine for each position who has a winning strategy from this position.

Thm: There is an algorithm for solving finite Muller games.

[Martin, Emerson & Jutla, Mostowski]

Part II

- Games: basic definitions.
- **Games behind model-checking.**
- Games behind synthesis.
- Extensions of the basic game model.
- Distributed synthesis.

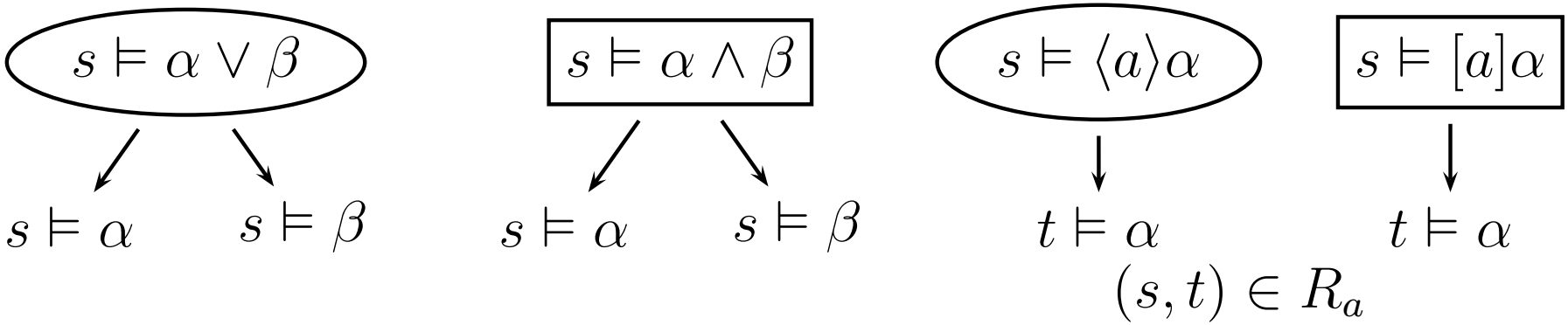
- Syntax: $P \mid \neg P \mid X \mid \alpha \vee \beta \mid \alpha \wedge \beta \mid \langle a \rangle \alpha \mid [a]\alpha \mid \mu X.\alpha \mid \nu X.\alpha$
- Semantics in a transition system $\mathcal{M} = \langle V, \{R_a\}_{a \in Act}, P^{\mathcal{M}}, \ldots \rangle$; we need $Val : Var \to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

$$
\begin{aligned}
[\![P]\!]^{\mathcal{M}}_{Val} &= P^{\mathcal{M}} \\
[\![X]\!]^{\mathcal{M}}_{Val} &= Val(X) \\
[\![\langle a \rangle \alpha]\!]^{\mathcal{M}}_{Val} &= \{v : \exists v'.\ R_a(v, v') \wedge v' \in [\![\alpha]\!]^{\mathcal{M}}_{Val}\} \\
[\![\mu X.\alpha(X)]\!]^{\mathcal{M}}_{Val} &= \bigcap \{S \subseteq V : [\![\alpha(S)]\!]^{\mathcal{M}}_{Val} \subseteq S\}
\end{aligned}
$$

- The model-checking problem: given a sentence α, a finite transition system $\mathcal{M}$, and a state s_0, check if $s_0 \in [\![\alpha]\!]^{\mathcal{M}}$. (Notation $\mathcal{M}, s_0 \models \alpha$)

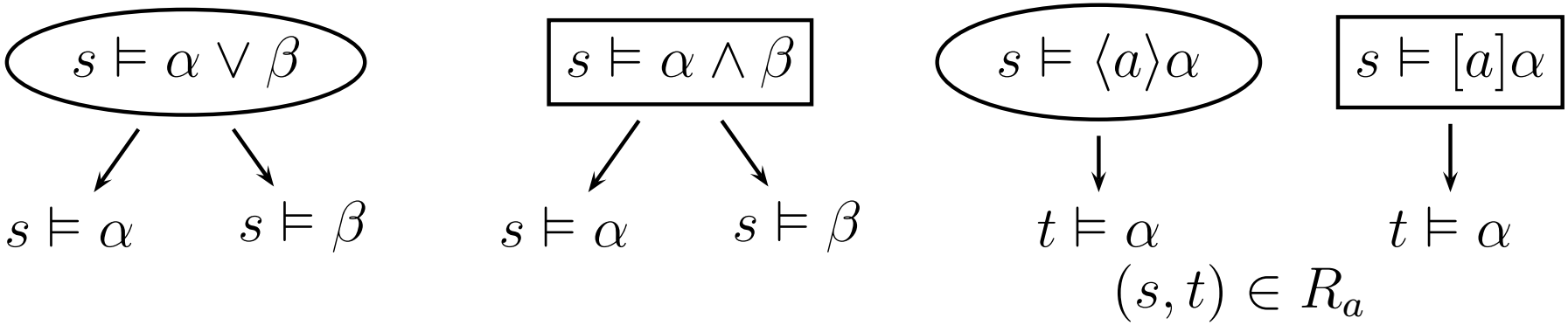
Model-checking

- We are given a transition system $\mathcal{M}$.
- Model checking rules

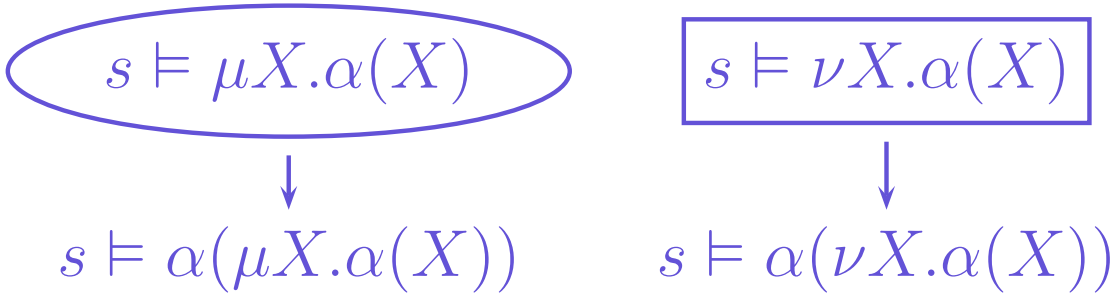


$s \models P$ Eve wins if $s \in P^{\mathcal{M}}$; $s \models \neg P$ Eve wins if $s \notin P^{\mathcal{M}}$.

- We are given a transition system $\mathcal{M}$.
- Model checking rules

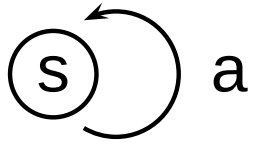


$s \models P$ Eve wins if $s \in P^{\mathcal{M}}$; $s \models \neg P$ Eve wins if $s \notin P^{\mathcal{M}}$.



- The last two rules may be a source of infinite plays.
- Wanted: Eve wins in $G(\mathcal{M}, \alpha)$ from $s_0 \models \alpha$ iff $\mathcal{M}, s_0 \models \alpha$.

Infinite plays



$$s \models \mu X.\langle a \rangle X \quad\quad s \models \nu X.\langle a \rangle X$$

$$\downarrow \quad\quad \downarrow$$

$$s \models \langle a \rangle \mu X.\langle a \rangle X \quad\quad s \models \langle a \rangle (\nu X.\langle a \rangle X)$$

$$\downarrow \quad\quad \downarrow$$

$$s \models \mu X.\langle a \rangle X \quad\quad s \models \nu X.\langle a \rangle X$$

$$\downarrow \quad\quad \downarrow$$

$$\vdots \quad\quad \vdots$$

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

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

$$[\![\mu^0 X.\beta(X)]\!]^{\mathcal{M}}_{Val} = \emptyset$$

$$[\![\mu^{\tau+1} X.\beta(X)]\!] = [\![\beta(X)]\!]^{\mathcal{M}}_{Val[[\![\mu^{\tau} X.\beta(X)]\!]^{\mathcal{M}}_{Val}/X]}$$

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

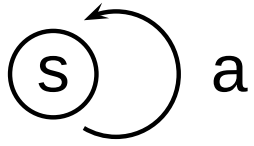
$$\nu X.\beta(X) = \bigcap_{\tau \in Ord} \nu^{\tau} X.\beta(X)$$

$$[\![\nu^0 X.\beta(X)]\!]^{\mathcal{M}}_{Val} = V$$

$$[\![\nu^{\tau+1} X.\beta(X)]\!] = [\![\beta(X)]\!]^{\mathcal{M}}_{Val[[\![\nu^{\tau} X.\beta(X)]\!]^{\mathcal{M}}_{Val}/X]}$$

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

Infinite plays

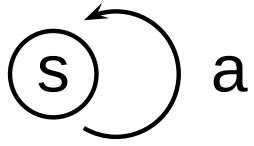


$$s \vDash \mu^{\tau} X.\langle a \rangle X \quad \downarrow \quad s \vDash \langle a \rangle(\mu^{\tau-1} X.\langle a \rangle X) \quad \downarrow \quad s \vDash \mu^{\tau-1} X.\langle a \rangle X \quad \downarrow \quad \vdots$$

$$s \vDash \nu^{\tau} X.\langle a \rangle X \quad \downarrow \quad s \vDash \langle a \rangle(\nu^{\tau} X.\langle a \rangle X) \quad \downarrow \quad s \vDash \nu^{\tau} X.\langle a \rangle X \quad \downarrow \quad \vdots$$

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

Infinite plays



$$s \vDash {}_3\mu^{\tau} X.\langle a\rangle X \qquad s \vDash {}_3\nu^{\tau} X.\langle a\rangle X$$

$$\downarrow \qquad\qquad \downarrow$$

$$s \vDash {}_1\langle a\rangle(\mu^{\tau-1} X.\langle a\rangle X) \qquad s \vDash {}_2\langle a\rangle(\nu^{\tau} X.\langle a\rangle X)$$

$$\downarrow \qquad\qquad \downarrow$$

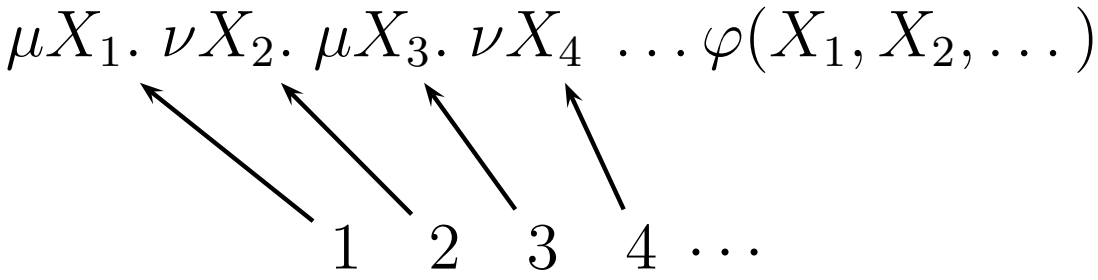
$$s \vDash {}_3\mu^{\tau-1} X.\langle a\rangle X \qquad s \vDash {}_3\nu^{\tau} X.\langle a\rangle X$$

$$\downarrow \qquad\qquad \downarrow$$

$$\vdots \qquad\qquad \vdots$$

- Eve should win in the second game but not in the first.
- Assign rank 1 to μ-regeneration and rank 2 to ν-regeneration.

Defining winning conditions



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

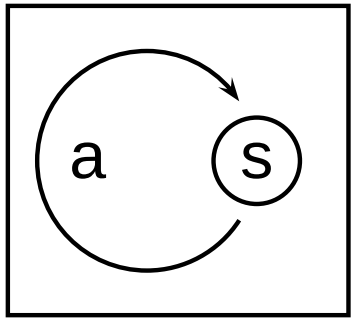
- With such acceptance conditions we have:

Thm [Emerson & Jutla, Stirling]:

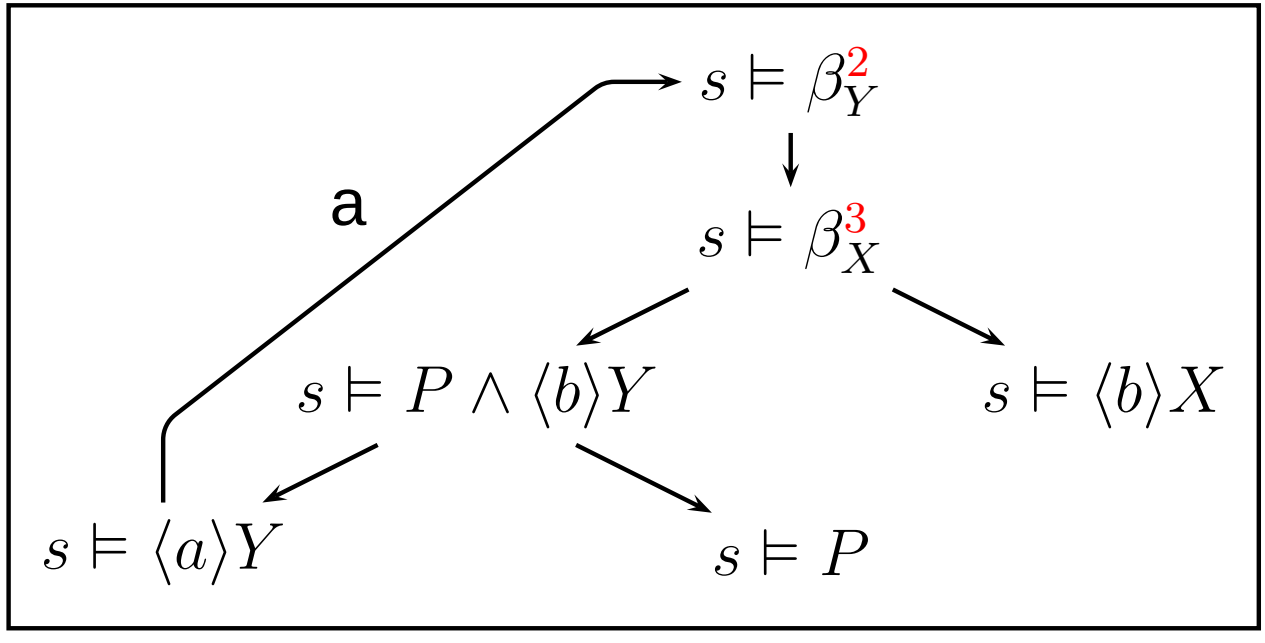
$\mathcal{M}, s_0 \vDash \alpha$ iff Eve wins in $G(\mathcal{M}, \alpha)$ from $s_0 \vDash \alpha$.

Example

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

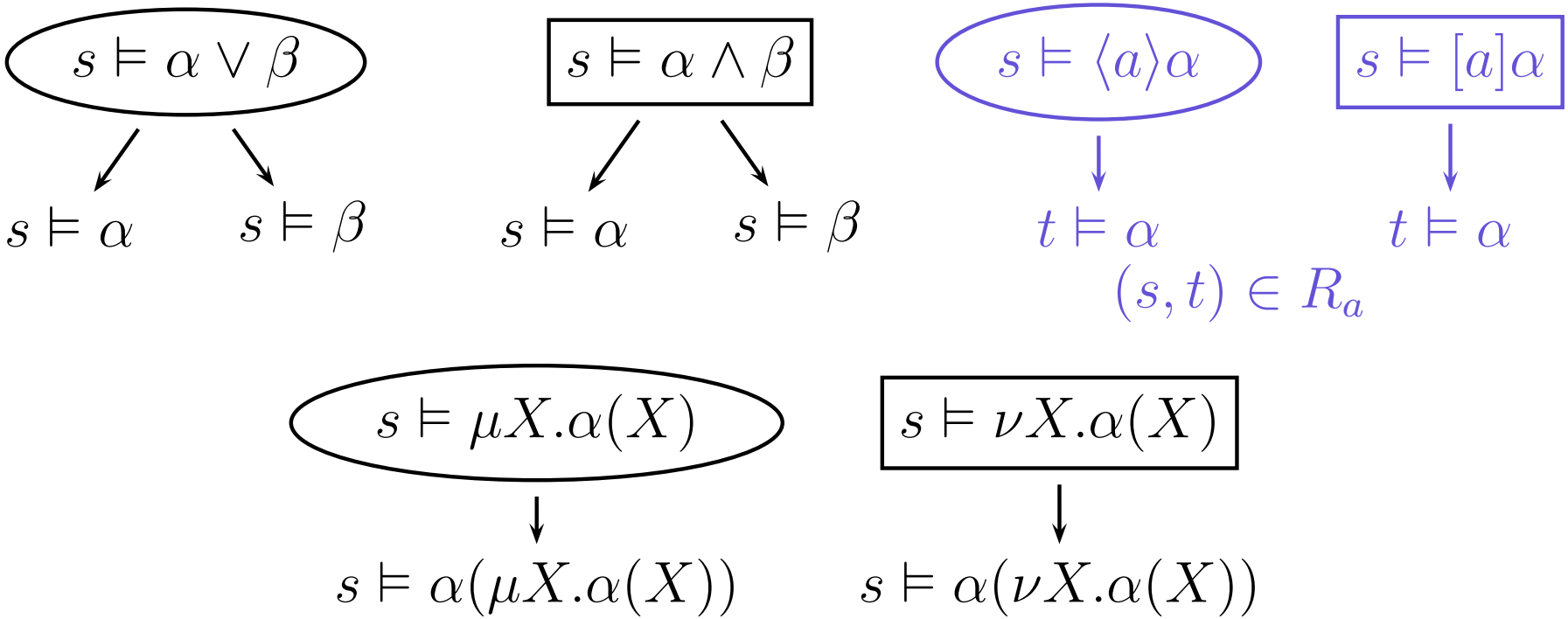


$\mathcal{M}$



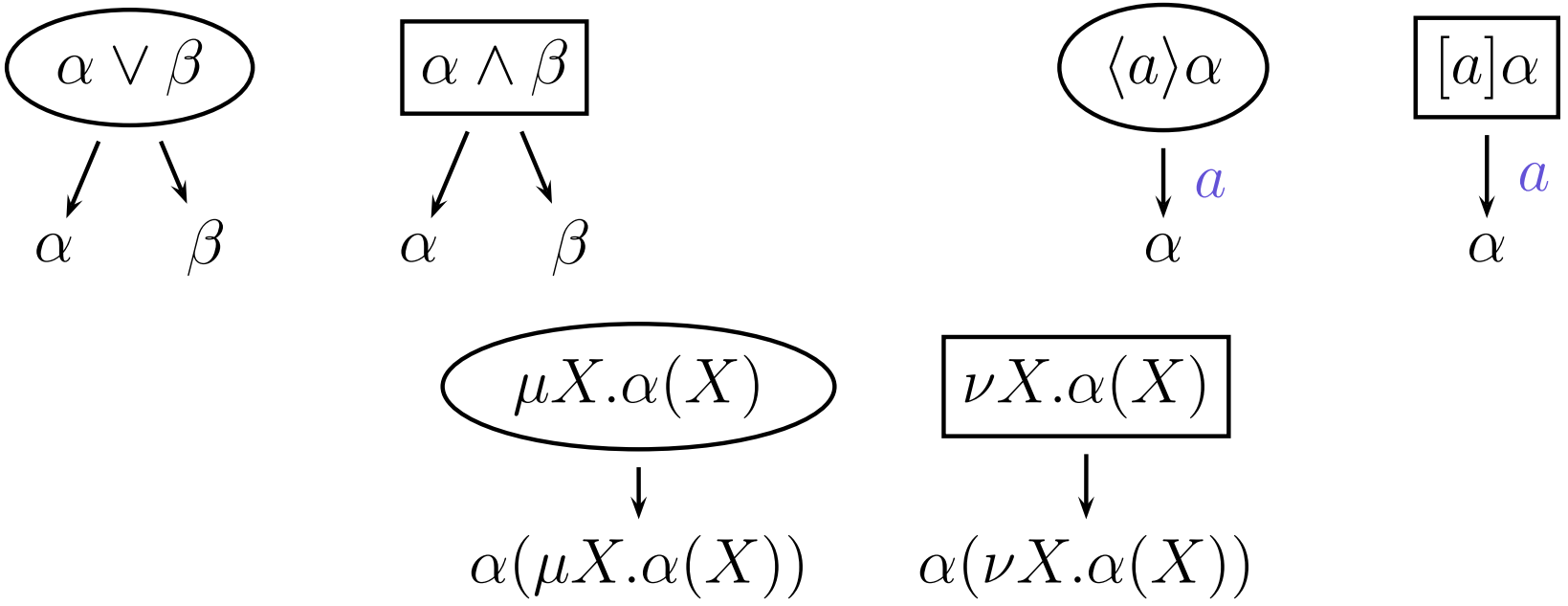
$G(\mathcal{M}, \alpha)$

- Model checking rules



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

- Tableaux rules

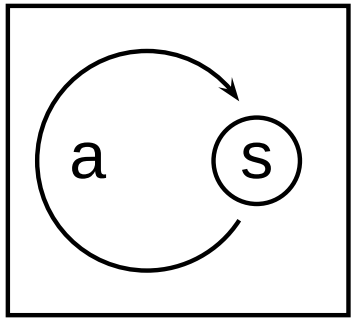


- These rules define a tableau $\mathcal{T}_\alpha$ for a formula α.
- Operation $\mathcal{M} \otimes \mathcal{T}_\alpha$ of "synchronized product" of a transition system and a tableau that gives the MC game.

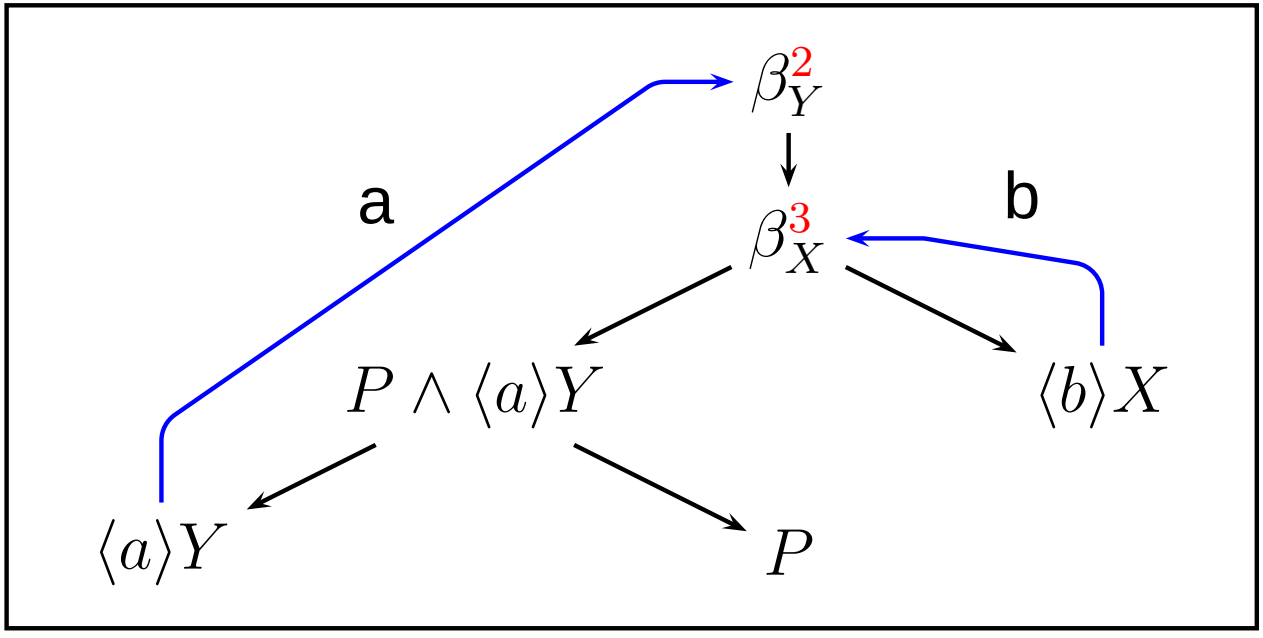
Obs: $\mathcal{M}, s_0 \vDash \alpha$ iff Eve wins from (s_0, α) in $\mathcal{M} \otimes \mathcal{T}_\alpha$.

Example

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



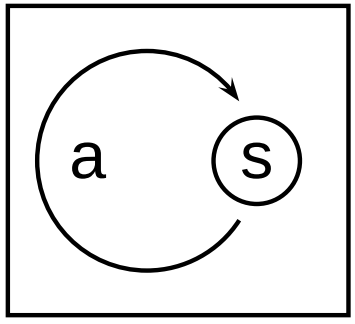
$\mathcal{M}$



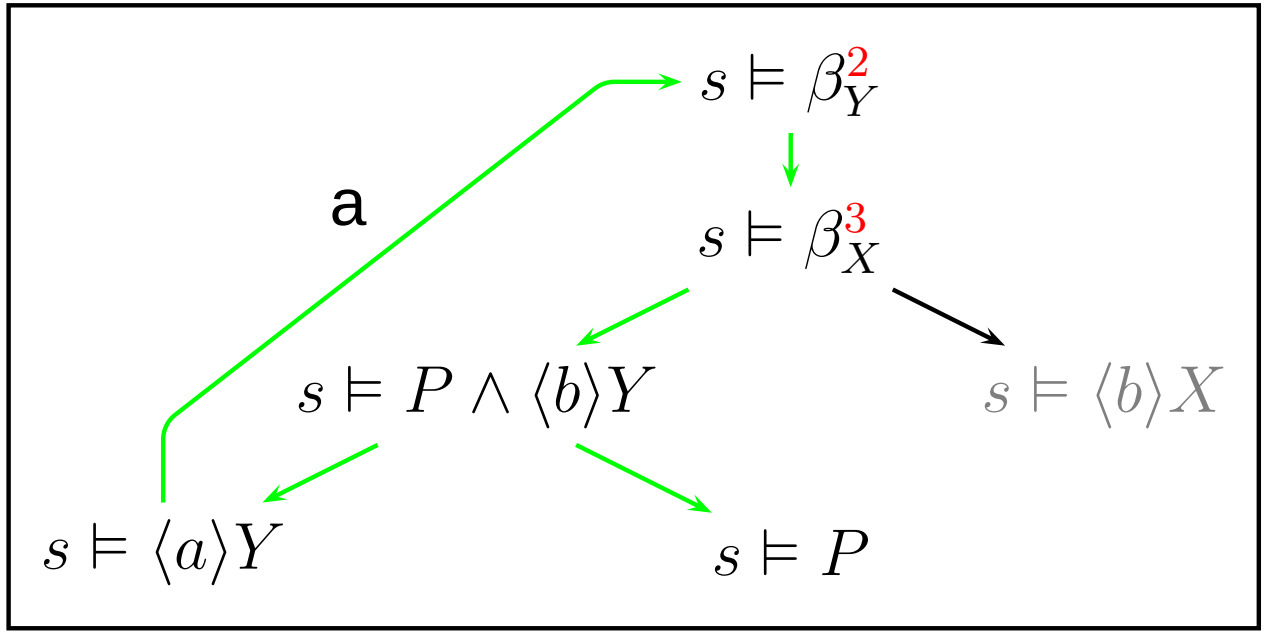
$\mathcal{T}_{\beta_Y}$

Example

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



$\mathcal{M}$



$\mathcal{M} \otimes \mathcal{T}_{\beta_Y}$

From MC to games

- Given a structure $\mathcal{M}$ and a formula α we construct the game $G(\mathcal{M}, \alpha)$ such that:

  $\mathcal{M}, s \models \alpha$ iff Eve wins from $(s \models \alpha)$ in $G(\mathcal{M}, \alpha)$

- The winning condition in $G(\mathcal{M}, \alpha)$ is a parity condition which size is the depth of alternation of fixpoints in α.

- One can defined a tableau $\mathcal{T}_\alpha$ and a synchronized product $\mathcal{M} \otimes \mathcal{T}_a$ so that $G(\mathcal{M}, \alpha) = \mathcal{M} \otimes \mathcal{T}_\alpha$.

- In particular the size of $|\mathcal{M}| \otimes |\mathcal{T}_\alpha|$ is $|\mathcal{M}| \cdot |\alpha|$.

- This works also for infinite transition systems.

- A game can be represented as a transition system where
  - proposition P_E designates Eve's positions,
  - propositions $P_0, \ldots, P_d$ define $\lambda : V \to \{0, \ldots, d\}$.

Thm [Emerson & Jutla]: There is a formula of the mu-calculus ε_d such that

$$\mathcal{M}_G, v \models \varepsilon_d \quad \text{iff} \quad \text{Eve wins from } v \text{ in } G.$$

$$\gamma(Z_0, \ldots, Z_d) =$$

$$\big(P_E \wedge \bigwedge_{i=0,\ldots,d} (P_i \Rightarrow \langle\,\rangle Z_i)\big) \vee \big(\neg P_E \wedge \bigwedge_{i=0,\ldots,d} (P_i \Rightarrow [\,] Z_i)\big)$$

$$\varepsilon_d = \nu Z_0 . \mu Z_1 \ldots \sigma Z_d .\ \gamma(Z_0, \ldots, Z_d)$$

In summary

- Parity games and model-checking for the mu-calculus are very close to each other (inter-reducible in linear time).

$$\mathcal{M}, s \models \alpha \quad \text{iff} \quad \text{in } \mathcal{M} \otimes \mathcal{T}_\alpha \text{ Eve wins from } (s, \alpha)$$

- Because of this translation it is enough to consider the games solving problem instead of MC problem.

- The tableau construction gives an alternating automaton accepting models of the formula.
- The $\mathcal{M} \otimes \mathcal{T}_\alpha$ operation defines the space of runs of the automaton $\mathcal{T}_\alpha$ on the structure $\mathcal{M}$.
- As $\mathcal{T}_\alpha$ accepts all models of α, the satisfiability problem reduces to the emptiness test of $\mathcal{T}_\alpha$.
- Indeed, the satisfiability game is obtained from converting $\mathcal{T}_\alpha$ into a nondeterministic automaton.

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

Transitions rules: $\Delta \subseteq Q \times \Gamma \times Q \times Op$

$$(q, a) \rightarrowtail (q', pop) \qquad (q, a) \rightarrowtail (q', push_b)$$

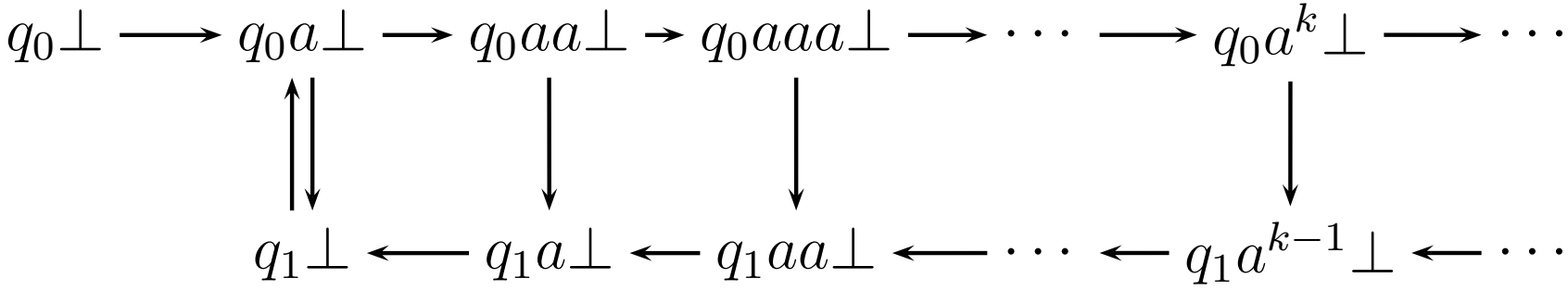
Pushdown graph: $G(P)$

Vertices: $Q \times \Gamma^*$

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

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

Pushdown graph: an example



- This is (a part of) the graph of the system:

$$(q_0, a) \rightarrowtail (q_0, push_a) \qquad (q_0, a) \rightarrowtail (q_1, pop)$$
$$(q_1, a) \rightarrowtail (q_1, pop)$$
$$(q_0, \bot) \rightarrowtail (q_0, push_a) \qquad (q_1, \bot) \rightarrowtail (q_0, push_a)$$

- The push-down model checking problem:

  Given P and α decide if α holds in the initial vertex of $G(P)$.

Model checking pushdown systems

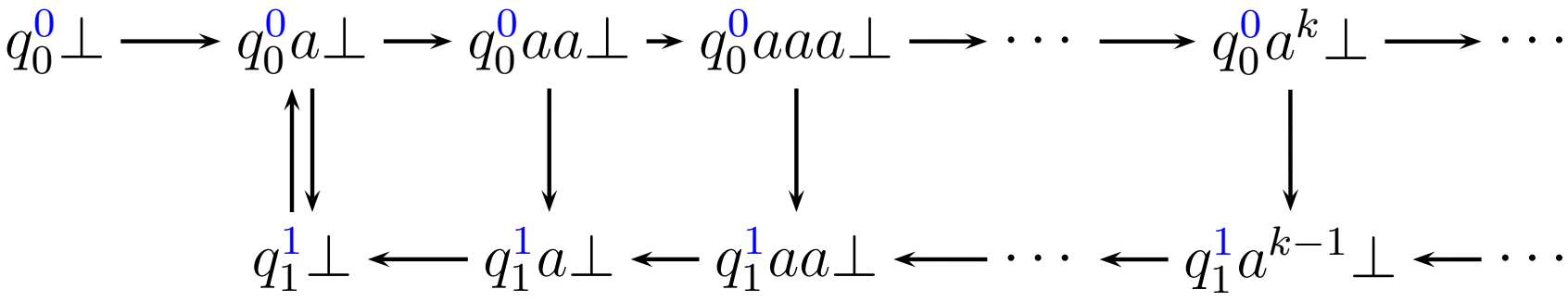
- Given P and α decide if α holds in the initial vertex of $G(P)$.
- Construct $\mathcal{T}_\alpha$ and the product $G(P) \otimes \mathcal{T}_\alpha$.
- This gives an infinite pushdown game:

$$P = \langle Q, \Gamma, \Delta, Q_E, Q_A, \Omega : Q \to \mathbb{N} \rangle$$

  - pushdown system with states partitioned between Eve and Adam
  - where each state is assigned a rank ($\Omega : Q \to \mathbb{N}$).

- α holds in the initial vertex of $G(P)$ iff
  Eve has a winning strategy from the initial vertex in the game.

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 $\Omega : Q \to \mathbb{N}$ decide who has a winning strategy from the initial vertex of $G(P)$.

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

Higher-order pushdown systems

- 1-store: a sequence $a_l \dots a_1$ over an alphabet Γ.
- n-store: a sequence $[s_l] \cdots [s_1]$ of $(n-1)$-stores.
- We have standard operations $push_a^1$ and pop_a^1.
- Additionally we have $push^k$ and pop^k operations:

$$push^k([s_l] \cdots [s_1]) = \begin{cases} [s_l][s_l] \cdots [s_1] & \text{stack order} = k \\ [push^k(s_l)] \cdots [s_1] & \text{stack order} > k \end{cases}$$

$$pop^n([s_l][s_{l-1}] \cdots [s_1]) = \begin{cases} [s_{l-1}] \cdots [s_1] & \text{stack order} = k \\ [s_{l-1}] \cdots [s_1] & \text{stack order} > k \end{cases}$$

- Pushdown system of order n: $P = \langle Q, \Gamma, \Delta \rangle$ where

$$\Delta \subseteq Q \times \Gamma \times Q \times Op_n.$$

Higher-order example

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

$$
\begin{array}{lllllll}
q_1[a] & \rightarrow & q_1[aa] & \rightarrow & \cdots & \rightarrow & q_1[a^k] \rightarrow \\
q_2[a^k][a^k] & \rightarrow & q_2[a^{k-1}][a^k] & \rightarrow & \cdots & \rightarrow & q_2[][a^k] \rightarrow \\
q_3[a^k] & \rightarrow & q_3[a^{k-1}] & \rightarrow & \cdots & \rightarrow & q_3[]
\end{array}
$$

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

MC for higher-order automata

- Once again the model checking problem reduces to solving games. This time higher-order pushdown games.

- Such a game is given by a higher-order pushdown automaton with states partitioned into Adam's and Eve's states and a function $\Omega : Q \to \mathbb{N}$.

Thm[Engelfreit, Cachat]: Solving n-order pushdown games is n-EXPTIME complete.

- Higher-order pushdown automata "implement" higher-order (safe) program schemes.

- The graphs of configurations of n-order pushdown automata are the graphs of n-th level of the Caucal hierarchy.

Part III

- Games: basic definitions.
- Games behind model-checking.
- **Games behind synthesis.**
- Extensions of the basic game model.
- Distributed synthesis.

Synthesis via satisfiability checking

- Synthesis problem I: Given a specification find a system satisfying it.
- Specification: propositional formula;
  System: valuation of variables.

$$\frac{\varphi \vee \psi, \Gamma}{\varphi, \Gamma} \qquad \frac{\varphi \vee \psi, \Gamma}{\varphi, \Gamma}$$ Eve chooses

$$\frac{\varphi \wedge \psi, \Gamma}{\varphi, \psi, \Gamma}$$ Adam chooses

when Γ-irreducible then Eve wins if no $P, \neg P \in \Gamma$.

- Eve has a winning strategy from $\{\varphi\}$ iff φ is satisfiable.
- Every model of φ can be obtained from a winning strategy in the satisfiability game for $\{\varphi\}$.

Extension to the μ-calculus

$$\frac{\Gamma}{\{\alpha, \{\beta : [a]\beta \in \Gamma\} : \langle a\rangle\alpha \in \Gamma\}}$$

Adam chooses

$$\frac{\Gamma, \mu X.\alpha(X)}{\Gamma, \alpha(\mu X.\alpha(X))} \qquad \frac{\Gamma, \nu X.\alpha(X)}{\Gamma, \alpha(\nu X.\alpha(X))}$$

- There are now infinite paths and we need a rule to decide the winner there.

Path condition

$$\dfrac{\dfrac{\mu X.\langle a\rangle X}{\langle a\rangle(\mu X.\langle a\rangle X)}}{\mu X.\langle a\rangle X}\quad\vdots \qquad \dfrac{\dfrac{\nu X.\langle a\rangle X}{\langle a\rangle(\nu X.\langle a\rangle X)}}{\nu X.\langle a\rangle X}\quad\vdots$$

- On the left Adam should win on the right it should be Eve.

$$\dfrac{\dfrac{\mu X.[a]X,\ \nu X.\langle a\rangle X}{[a](\mu\ldots),\ \langle a\rangle(\nu\ldots)}}{\mu X.[a]X,\ \nu X.\langle a\rangle X}$$

- The conditions should talk about traces inside the path.

Extension to the μ-calculus

$$\frac{\Gamma}{\{\alpha, \{\beta : [a]\beta \in \Gamma\} : \langle a\rangle\alpha \in \Gamma\}}$$

Adam chooses

$$\frac{\Gamma, \mu X.\alpha(X)}{\Gamma, \alpha(\mu X.\alpha(X))} \qquad \frac{\Gamma, \nu X.\alpha(X)}{\Gamma, \alpha(\nu X.\alpha(X))}$$

- The rule for infnite paths says that Eve wins if there is no bad trace inside the path. (It can be converted to a parity condition).
- Eve wins from $\{\alpha\}$ iff α is satisfiable.
- Every model (transition system) for α comes from some winning strategy in this game.

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

Control problem for a given plant

- A plant is a deterministic transition system over Σ.

$$P = \langle S^p,\ \Sigma,\ s_I^p,\ e^p : S \times A \dot{\rightarrow} S \rangle$$

Given P and α, find a controller C (deterministic transition system) s.t. $P \times C \models \alpha$.

$$P \times C = \langle S = S^p \times S^c,\ \Sigma,\ (s_I^p, s_I^c),\ e : S \times \Sigma \dot{\rightarrow} S \rangle$$
$$e((s_p, s_c), a) = (e^p(s_p, a),\ e^c(s_c, a))$$

- Solution:
- Define an operation α/P such that:

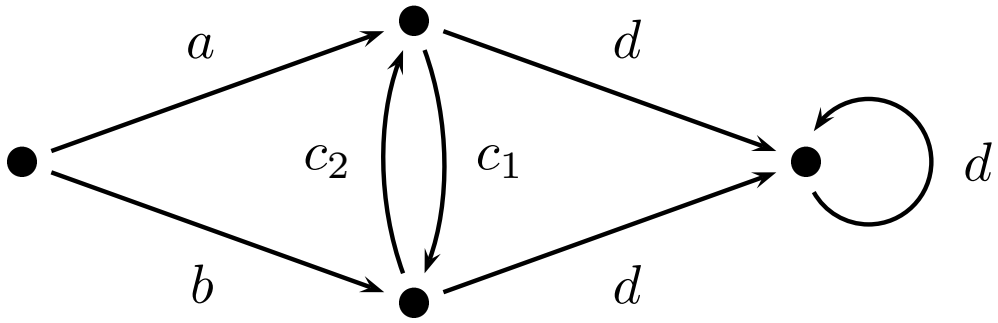
$$C \models \alpha/P \quad \text{iff} \quad P \times C \models \alpha$$

- Find a model C of α/P.

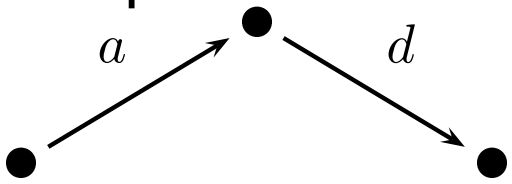
Example

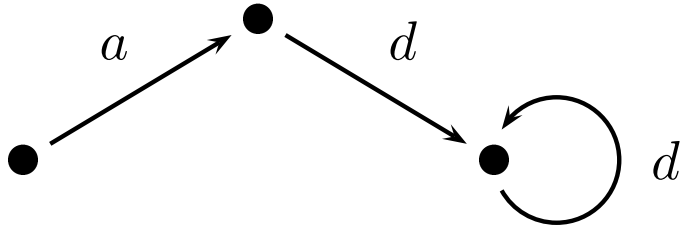
$\alpha \equiv$ execute d action

P:



- Some possible controllers:





- The second solution is non-blocking.

$$C \models (\alpha / P) \wedge \beta_{nonblock}$$

- So we can require additional properties from the controller.

Uncontrollable/unobseravable actions

- Divide Σ into:
- Σ_{con} and Σ_{ucon} of controllable and uncontrollable actions.
- Σ_{obs} and Σ_{uobs} of observable and unobservable actions.

- Additional conditions:

θ_{ucon} : C cannot forbid actions from Σ_{ucon}

$$\equiv \forall s \in S. \quad \forall a \in \Sigma_{ucon}. \quad e(s,a) \text{ defined}$$

$$\equiv \nu X. \quad \Big(\bigwedge_{a \in \Sigma} [a]X\Big) \quad \wedge \quad \Big(\bigwedge_{a \in \Sigma_{ucon}} \langle a \rangle tt\Big)$$

θ_{uobs} : C cannot observe actions from Σ_{uobs}

$$\equiv \forall s \in S. \quad \forall a \in \Sigma_{uobs}. \quad e(s,a) = s$$

$$\equiv \nu X. \quad \Big(\bigwedge_{a \in \Sigma} [a]X\Big) \quad \wedge \quad \Big(\bigwedge_{a \in \Sigma_{uobs}} \circlearrowleft_a\Big)$$

Uncontrollable/unobservable actions

- Divide Σ into:
- Σ_{con} and Σ_{ucon} of controllable and uncontrollable actions.
- Σ_{obs} and Σ_{uobs} of observable and unobservable actions.

- Additional conditions:

θ_{ucon} : C cannot forbid actions from Σ_{ucon}

$$\equiv \nu X. \quad \Big(\bigwedge_{a\in\Sigma}[a]X\Big) \quad \wedge \quad \Big(\bigwedge_{a\in\Sigma_{ucon}}\langle a\rangle tt\Big)$$

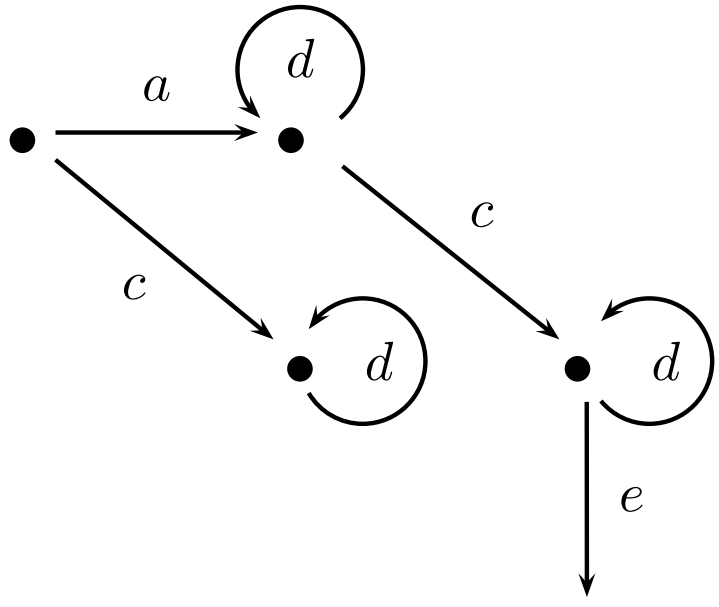
θ_{uobs} : C cannot observe actions from Σ_{uobs}

$$\equiv \nu X. \quad \Big(\bigwedge_{a\in\Sigma}[a]X\Big) \quad \wedge \quad \Big(\bigwedge_{a\in\Sigma_{uobs}}\circlearrowleft_a\Big)$$

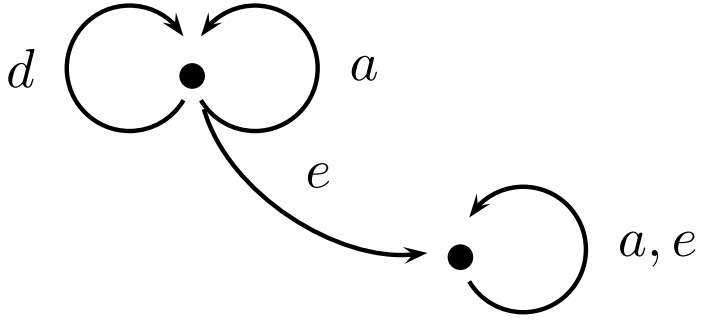
- Solution: Find $C \vDash (\alpha/P) \wedge \theta_{ucon} \wedge \theta_{uobs}$

Example

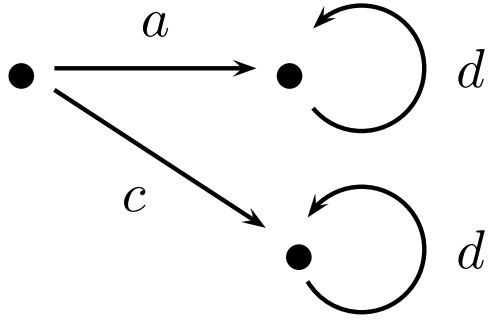
$\Sigma_{uobs} = \{a\}$, $\Sigma_{ucon} = \{e\}$ and the goal is to avoid e



- A solution:



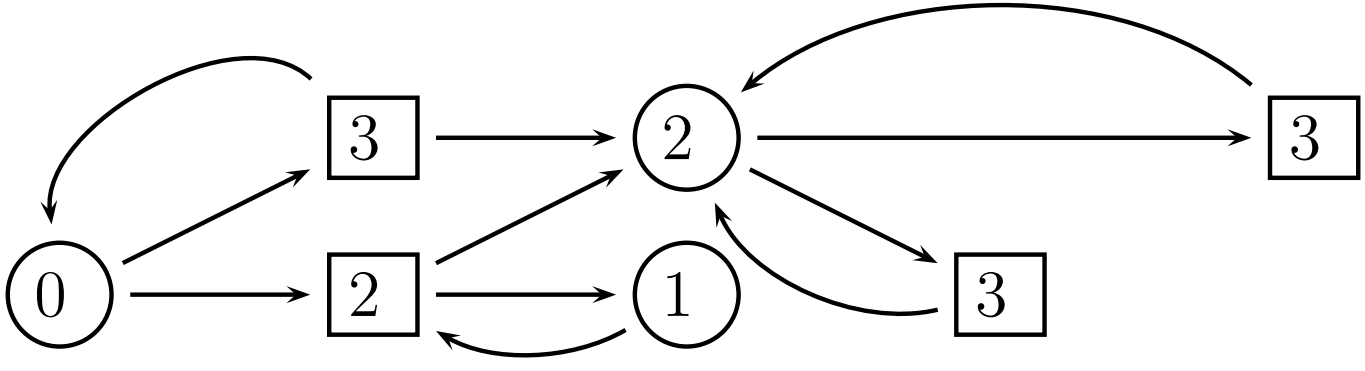
Wrong:



Games as specifications

- Till now we have reduced various synthesis problems to games.

- Games themselves can be considered as specifications and strategies as programs.



- In this kind of setting we can vary only the shape of a graph and the rest of a specification is fixed.

All centralized synthesis problems are reducible to this one.

Part IV

- Games: basic definitions.
- Games behind model-checking.
- Games behind synthesis.
- **Extensions.**
- Distributed synthesis.

More elaborate winning conditions

- The mu-calculus specifications translate into parity winning conditions. Similarly for other standard program logics.

- In the context of push-down games we have phenomena not expressible in these logics:

  explosion: the height of the stack is unbounded.

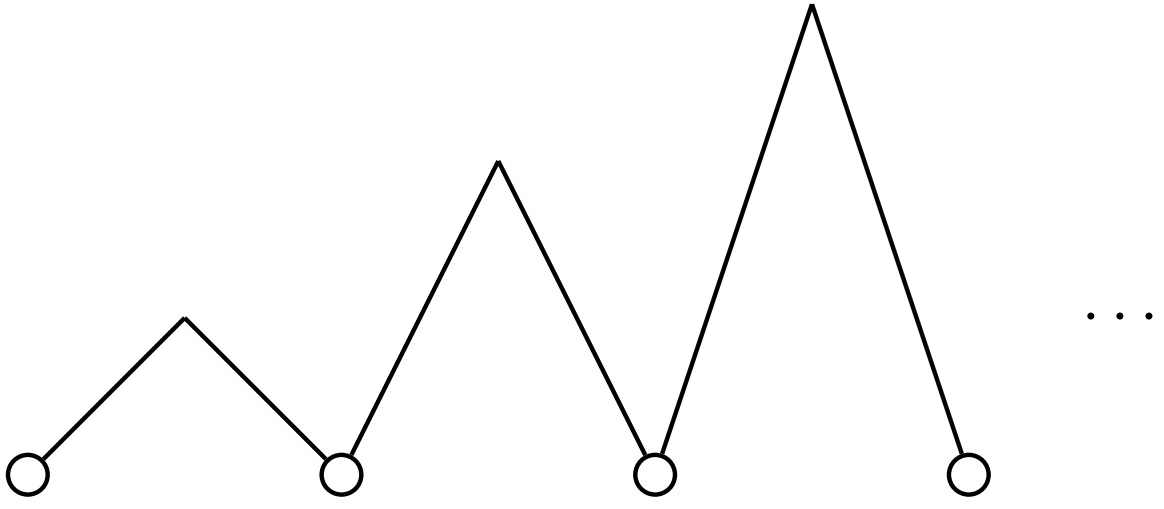
Thm[Cachat & Duparc & Thomas, Bouquet & Serre & W., Gimbert]: Games with winning conditions that are boolean combinations of parity and explosion conditions can be solved in EXPTIME.

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

Example

$$q_0^0\perp \longrightarrow q_0^0 a\perp \longrightarrow q_0^1 aa\perp \rightarrow q_0^1 aaa\perp \longrightarrow \cdots \longrightarrow q_0^1 a^k\perp \longrightarrow \cdots$$

$$q_1^1\perp \longleftarrow q_1^1 a\perp \longleftarrow q_1^1 aa\perp \longleftarrow \cdots \longleftarrow q_1^1 a^{k-1}\perp \longleftarrow \cdots$$



Conditions admitting positional strategies

- Memoryless strategies are interesting as the size of memory influences:
  - the size of controllers,
  - the size of counterexamples,
  - the complexity of the algorithms.

- A winning condition admits positional determinacy iff all the games with this condition are positionally determined.

Thm [McNaughton]: Parity condition is the only Muller condition admitting positional determinacy.

Rem [Zielonka]: If all nodes need to be coloured then the class is a bit bigger.

Other types of conditions

- Muller conditions with infinite number of colours.
$$G = \{V_E, V_A, R, \lambda : V \to \omega\}$$

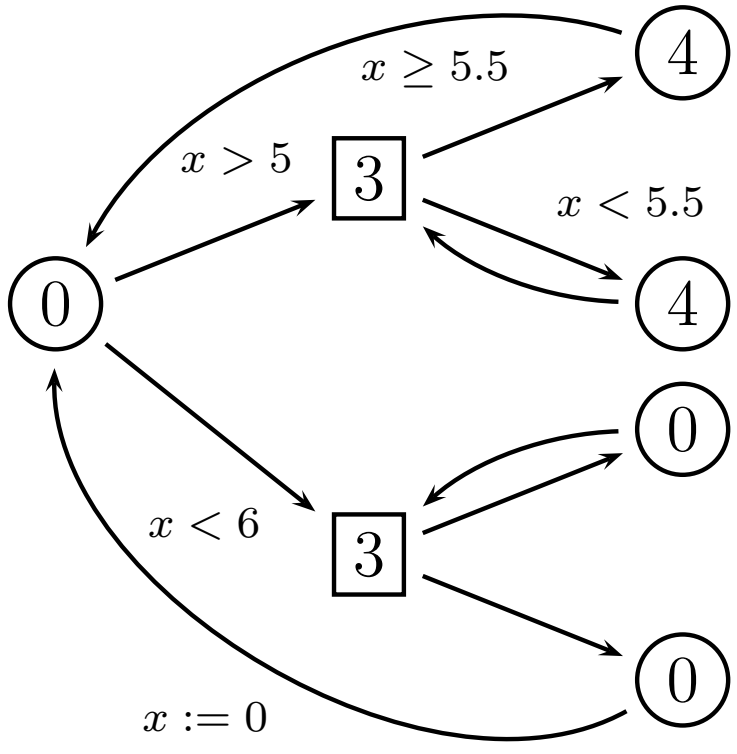
- Infinite parity condition:
Eve wins iff $\min(\mathrm{Inf}(p))$ is even or $\mathrm{Inf}(p) = \emptyset$.

Thm[Graedel & W.]: Games with infinite parity condition admit memoryless determinacy. All other conditions need infinite memory.

Thm[Graedel & W.]: The conditions given by $\lambda : V \to (\omega + 1)$ admit positional determinacy over graphs of bounded out-degree.

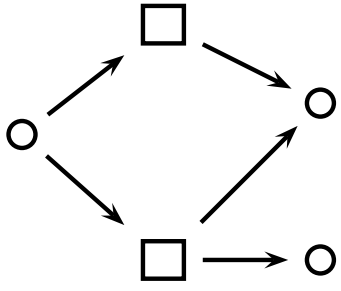
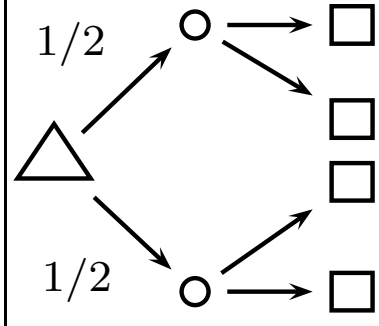
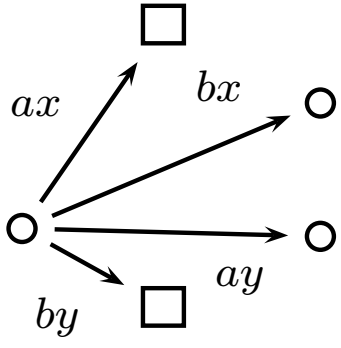
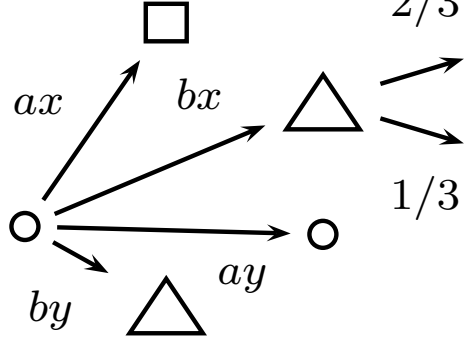
Thm [Colcombet & Niwiński]: If partial colouring functions are allowed then only finite parity conditions admit positional determinacy.

Games with time



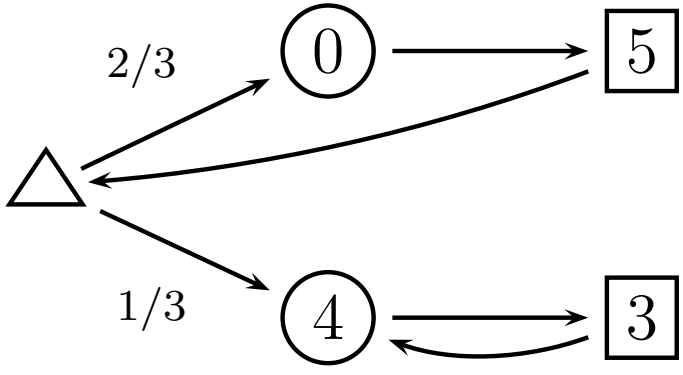
- Like in timed automata, game has clocks and restrictions on when transitions can be taken.

- A taxonomy of the types of rules:

	Deterministic	Probabilistic
Turn-based		
Concurrent		

Perfect information stochastic games

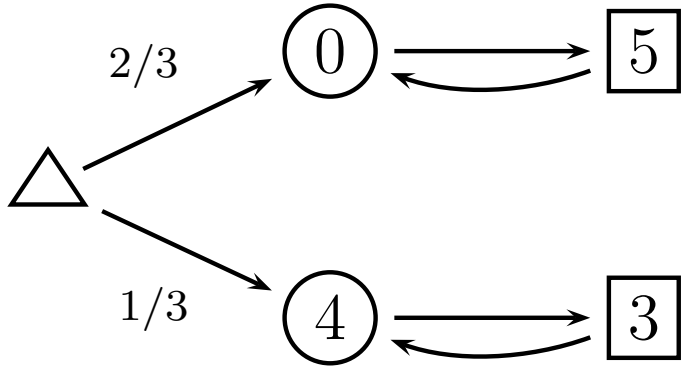
- We add randomized positions: Δ.
- In such a vertex we a have a probability distribution on outgoing edges.



- Adam wins in this game

Perfect information stochastic games

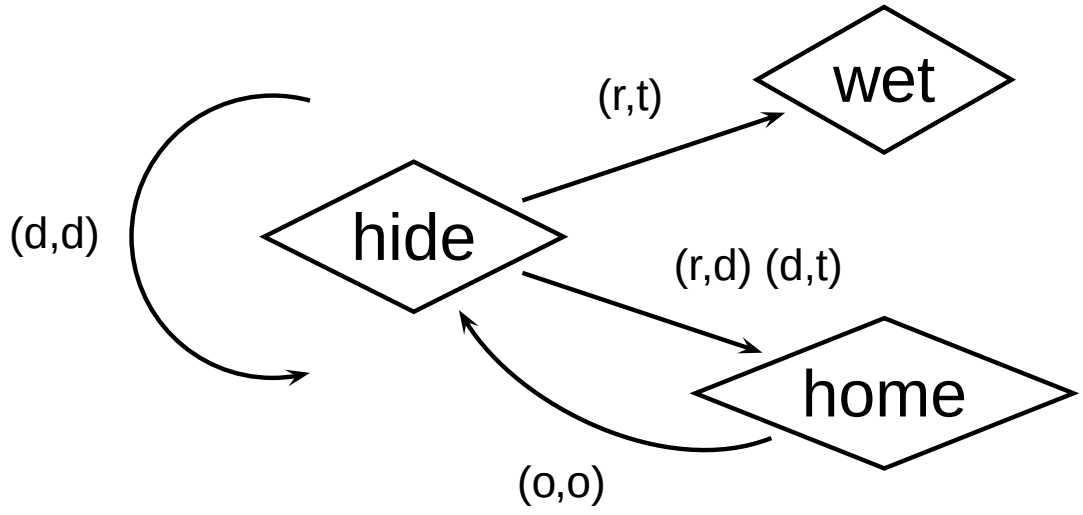
- None of the players may be sure to win.



- Eve wins with the probability $2/3$ and Adam with the probability $1/3$.

Thm [de Alfaro & Majumdar, Chatterjee & Jurdziński & Henzinger, Zielonka] :
In a finite game each state has a value and each player has an positional, pure and optimal strategy.

Concurrent games



Eve	Adam
d	d
r	t
o	o

- Two players choose their moves concurrently. Their joint choice determines the successor.
- None of the players may have a pure winning strategy.
- There exists randomized strategies, but they may require infinite memory [de Alfaro, Henzinger].

Part V

- Games: basic definitions.
- Games behind model-checking.
- Games behind synthesis.
- Extensions of the basic game model.
- **Distributed synthesis.**

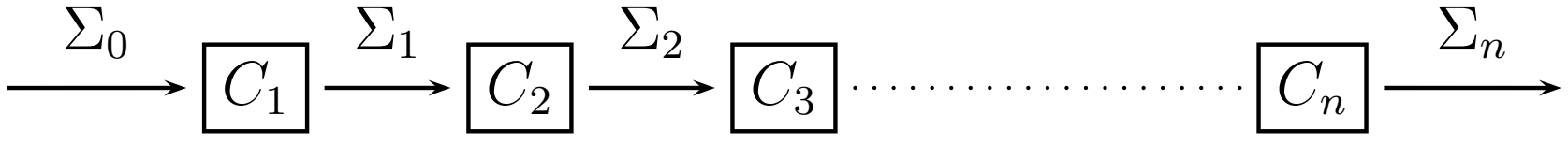
Distributed synthesis

Given a plant P and formulas α, β_1, β_2 do there exist controllers C_1, C_2 such that:

$$C_1 \models \beta_1, \quad C_2 \models \beta_2 \quad \text{and} \quad P \times C_1 \times C_2 \models \alpha.$$

(Each controller has its own Σ^i_{ucon} and Σ^i_{uobs}.)

Synthesis for a given architecture:



Given a plant P and formulas α, β_1, β_2 do there exist controllers C_1, C_2 such that:

$$C_1 \models \beta_1, \quad C_2 \models \beta_2 \quad \text{and} \quad P \times C_1 \times C_2 \models \alpha.$$

(Each controller has its own Σ^i_{ucon} and Σ^i_{uobs}.)

- Define new operation α/β with the property:

  $P \models \alpha/\beta$ iff there is C such that $C \models \beta$ and $P \times C \models \alpha$

- The operation α/β works only if β does not use $\circlearrowleft$.

- We have:

  $P \models (\alpha/\beta_1)/\beta_2$ iff there is C_2 with $P \times C_2 \models \alpha/\beta_1$
  iff there are C_1, C_2 with $P \times C_1 \times C_2 \models \alpha$.

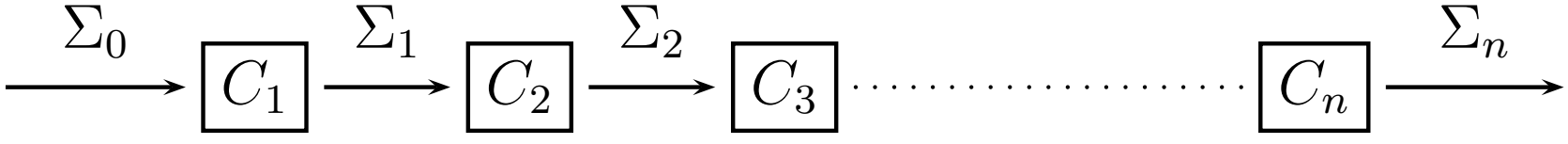
Distributed synthesis is difficult

Fact: The following problem is undecidable:
Given α, β_1, β_2 are there C_1, C_2 such that $C_1 \times C_2 \models \alpha$ and $C_1 \models \beta_1$, $C_2 \models \beta_2$.

Thm[Pnueli & Rosner]: The problem:

> For a fixed architecture, given α are there controllers that make the system satisfy α.

is decidable only for pipelines.



Distributed synthesis is difficult

Fact: The following problem is undecidable:
Given α, β_1, β_2 are there C_1, C_2 such that $C_1 \times C_2 \models \alpha$ and $C_1 \models \beta_1$, $C_2 \models \beta_2$.

Thm[Pnueli & Rosner]: The problem:

> For a fixed architecture, given α are there controllers that make the system satisfy α.

is decidable only for pipelines.

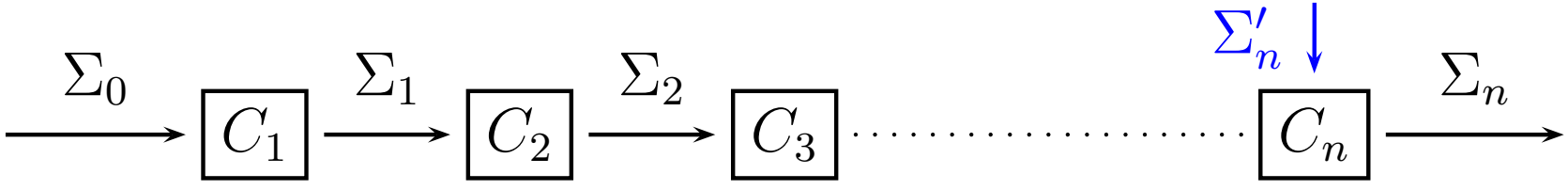
Distributed synthesis is difficult (2)

- A specification is local if it is a conjunction of requirements on each controller.

Thm[Madhusudan]: The problem:

> For a fixed architecture given a local specification, are there controllers that make the system satisfy the specification.

is decidable only for doubly flanked pipelines.

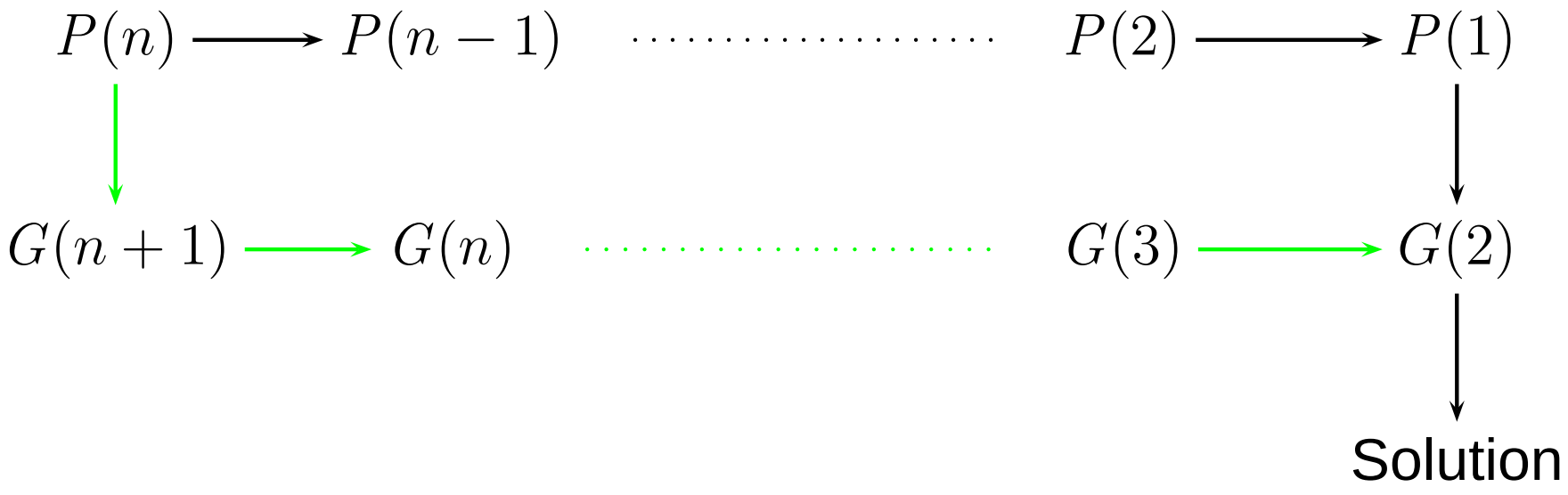


Summary

- For most architectures there are specifications that make the problem undecidable.
- It may be more fruitful to take a specification into account and look for which pairs (architecture, specification) the problem is decidable.
- Idea: Compile (architecture, specification) pair into a game and use tools developed there.
- Problem: Compiling to two player games does not make much sense.
- We want a setting with a coalition of players against the environment.
- Distributed games to distributed strategies as standard games to centralized strategies.

Solving distributed synthesis

Solving distributed synthesis

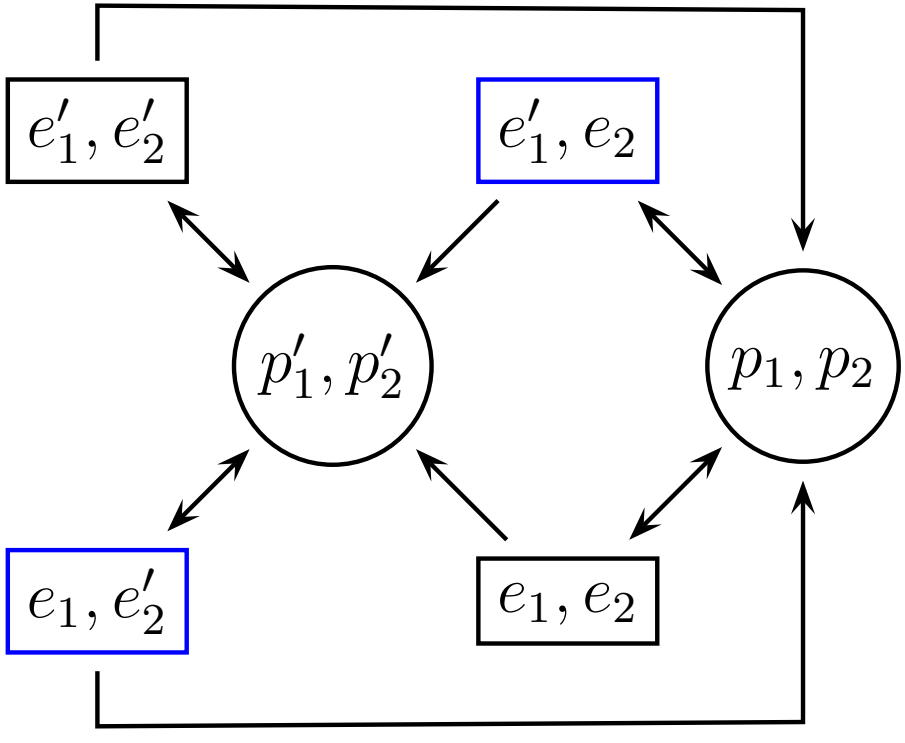


- The game setting can be:
  - more general,
  - combinatorially easier to handle.

Distributed game

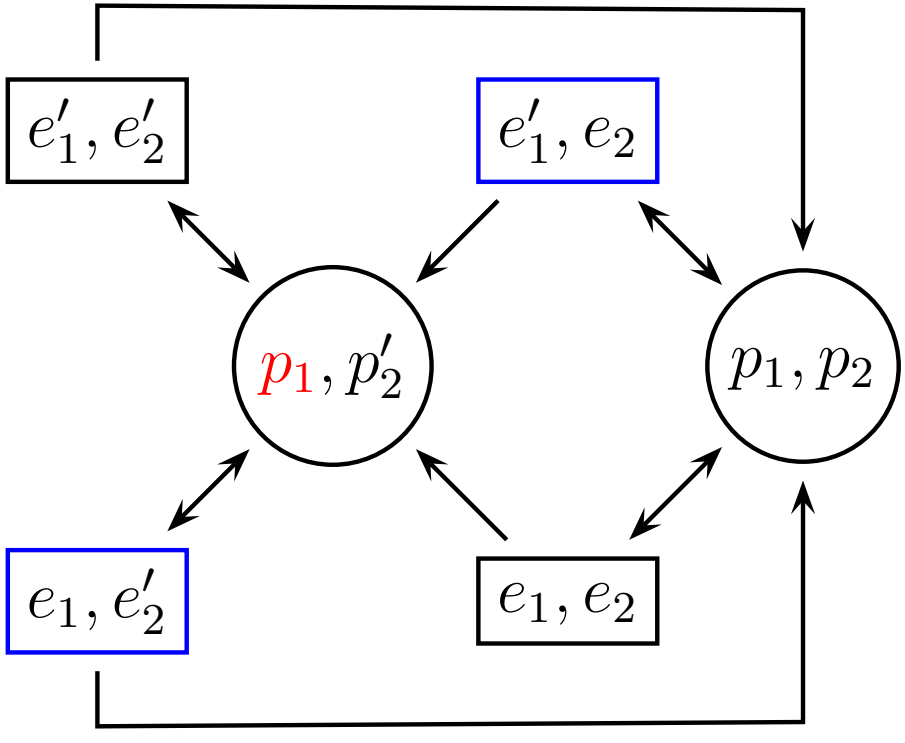
- Take n "local" games $G_i = \langle A_i, R_i, T_i \rangle$. (bipartite)
- Distributed game $\mathcal{G} = \langle A, E, R, Acc \subseteq (E \cup P)^\omega \rangle$.
- $E = E_1 \times \cdots \times E_n$,
- $A \subseteq (A_1 \cup E_1) \times \cdots \times (A_n \cup E_n) \setminus E$.
- Eve's (environment) moves: $[e_1, \ldots, e_n] \to (x_1, \ldots, x_n)$ with
  $$x_i = e_i \text{ or } e_i \to x_i.$$
  Some of these transitions can be suppressed.
- Adam (system) moves: $(x_1, \ldots, x_n) \to [e_1, \ldots, e_n]$ with
  $$x_i = e_i \text{ or } x_i \to e_i.$$
  Every such transition must be present.

Example



- Goal: Avoid blue positions.

Example 2



- Goal: Avoid blue positions.

Distributed strategies

- Given a play $\vec{v}$ in $\mathcal{G}$, a view of Adam i is $view_i(v) \in (E_i \cdot P_i)^\omega$.

$(e_1, e_2, \ldots, e_n)$	e_1
$(e_1, p_2, \ldots, e_n)$	
$(e_1, e'_2, \ldots, e_n)$	
$(p_1, e_2, \ldots, e_n)$	p_1
$(e'_1, e'_2, \ldots, e_n)$	e_1

- An i-local strategy is a strategy in the game G_i.

- Distributed strategy is a tuple $\langle \sigma_1, \ldots, \sigma_n \rangle$ of local strategies.

$$\sigma(\vec{v} \cdot (x_1, \ldots, x_n)) = (e_1, \ldots, e_n)$$

where $e_i = x_i$ or $e_i = \sigma_i(view_i(\vec{v} \cdot x_i))$.

Observations

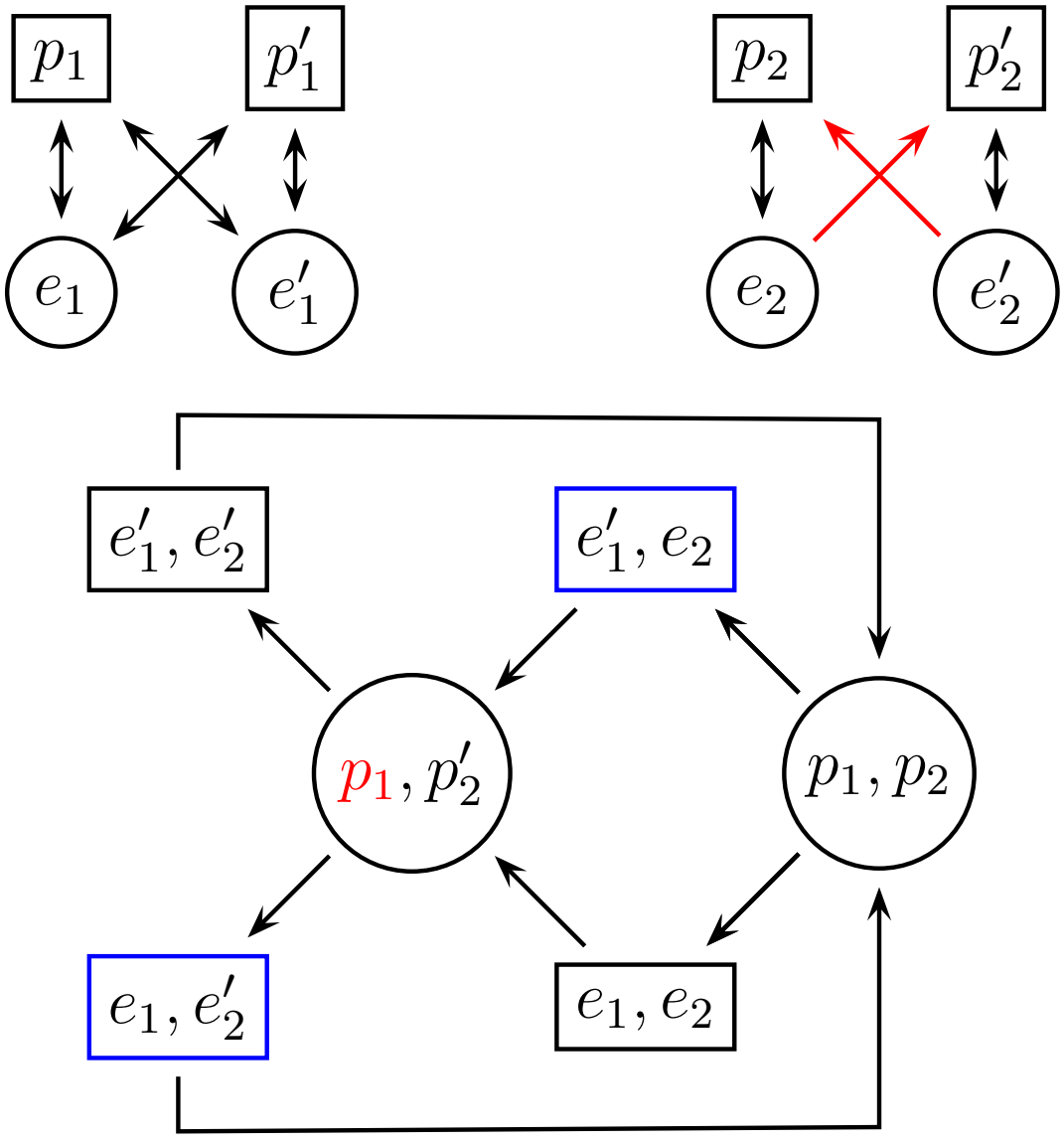
- Adams may have a global strategy in a game but not a distributed one. (Distributed games are not determined).

- Distributed games are like concurrent games, but the players who have partial information play with and not against each other.

- It is not decidable if there is a distributed winning strategy in a given distributed game.

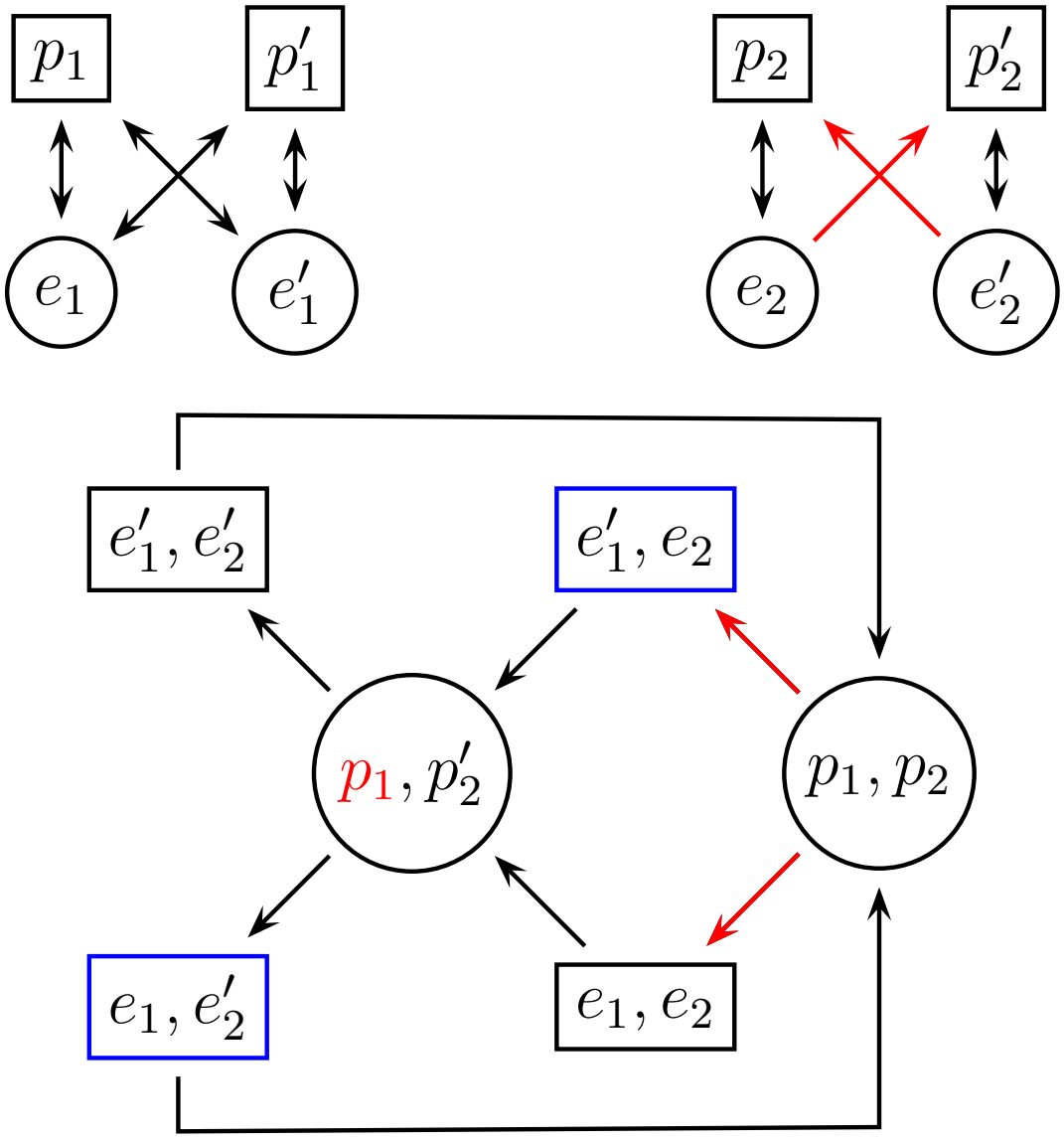
- There may be a memoryless global strategy but all distributed strategies may require memory.

Example 3



- Goal: Avoid blue positions.

Example 3



- Goal: Avoid blue positions.

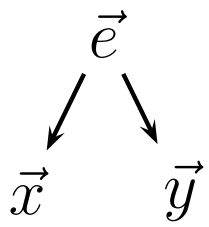
Deterministic games

- The game is deterministic for the environment iff every environment position has at most one successor.

- If Adams have a global strategy in such a game then they have a distributed one.

Cor: Environment deterministic distributed games are solvable. (Existence of distributed strategies is decidable).

Simplification theorems

- A game is k-deterministic
  if whenever  then $\vec{x}[k] \neq \vec{y}[k]$.

(Adam k, can deduce the move of Eve).

- If a game is 1 and n-deterministic then we can "glue together" players 1 and n (Thm 1).
  - We get a game with smaller number of Adams.
  - There is a distributed strategy in the new game iff there is one in the old game.

- If a game is not i-deterministic then, under some conditions, we can apply a kind of "powerset construction" to make it i-deterministic (Thm 2).

Summary

- Distributed games are in general neither determined nor algorithmically solvable.
- Many known settings of distributed synthesis are representable in distributed games.
  - Pipelines.
  - Local specifications and double flanked pipelines.
  - Madhusudan & Thiagarajan setting.
  - Rudie & Wonham distributed control.
- The solutions require some coding and two theorems.
- Distributed games can be hopefully as useful for distributed synthesis problem as two player games are for the centralized synthesis problem.

Directions

- Classes of graphs for which game solving is decidable.
- Unsafe higher-order program schemes.
- Good winning conditions for push-down systems.
- More decidable cases for distributed synthesis.
- Randomized strategies in distributed games.

Conclusions

- Games are behind model-checking and synthesis problems.
- Parity games are tied with the μ-calculus model-checking (other logics also can be easily put into the game setting).
- This connection is sometimes lost in more elaborate settings but sometimes stays (concurrent probabilistic games with parity conditions).
- Often in these new settings games are all what is left from the classical setting.
- New game models are needed to capture concurrency directly.

Conclusions

- Games are behind model-checking and synthesis problems.
- Parity games are tied with the μ-calculus model-checking (other logics also can be easily put into the game setting).
- This connection is sometimes lost in more elaborate settings but sometimes stays (concurrent probabilistic games with parity conditions).
- Often in these new settings games are all what is left from the classical setting.
- New game models are needed to capture concurrency directly.

The playful universe is expanding.