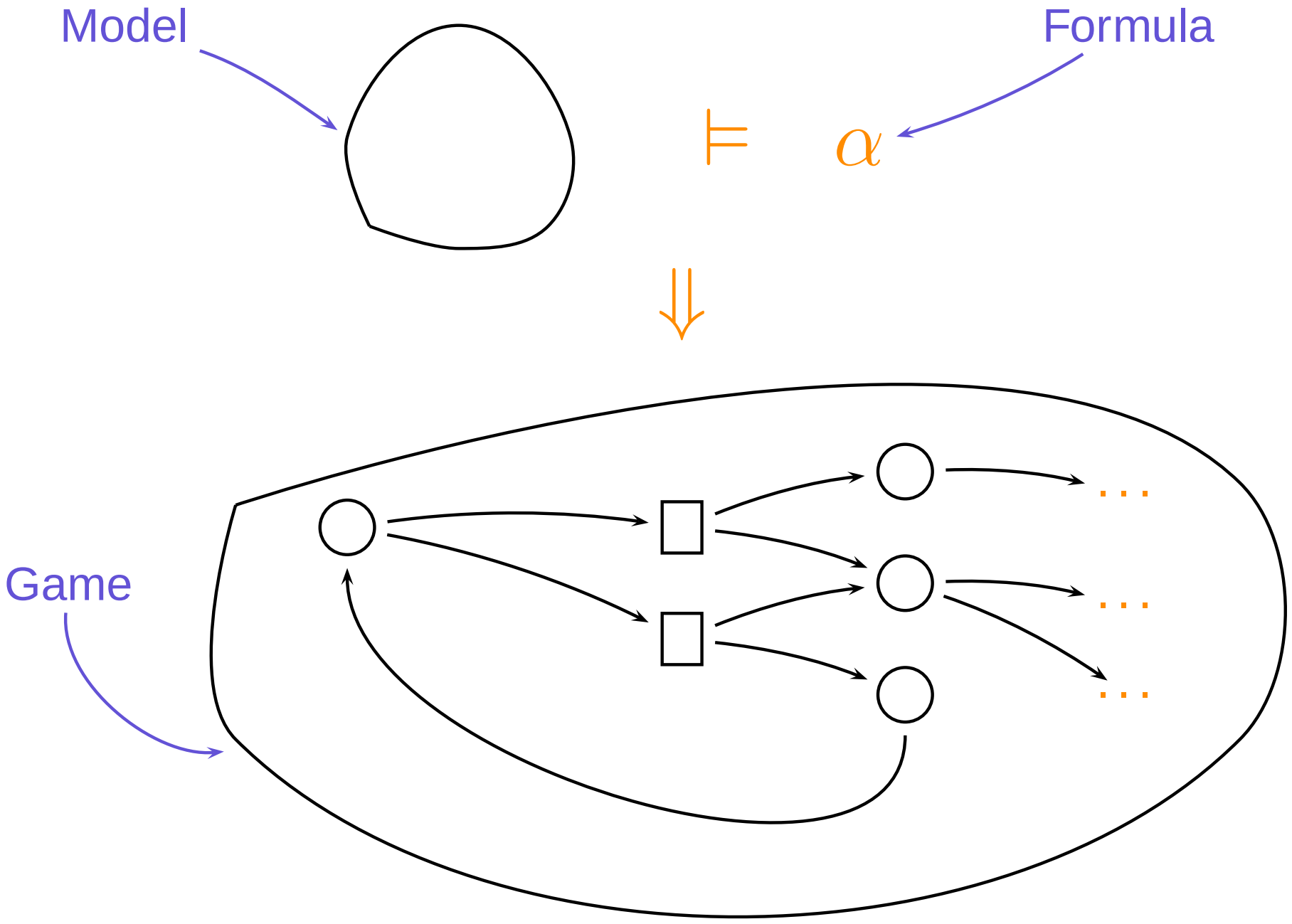
From logic to games

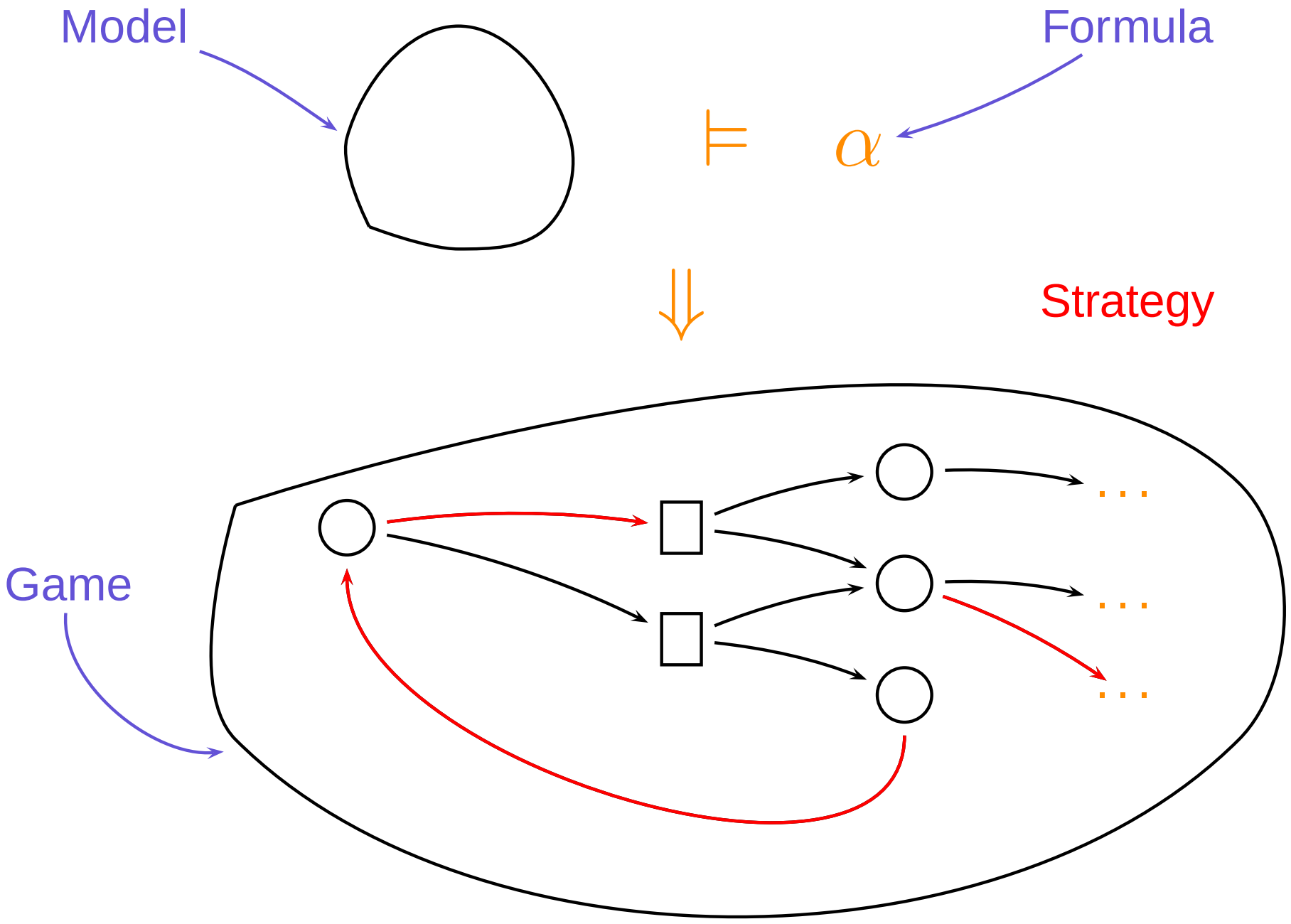
Igor Walukiewicz

CNRS, Bordeaux

The big picture



The big picture



Plan

- Why logical formalisms.
- Two formalisms for model-checking.
- Advantages of formal systems.
- Model-checking as a game.
- Two player infinite games.
- Solving games.
- Special strategies in games.

Logic at the birth of Computer Science

A. Turing, ‘‘On computable numbers, with an application to the Entscheidungsproblem’’, Proc of London Mathematical Society, 1936

- Shows that the famous “Hilbert’s Entscheidungsproblem” is algorithmically unsolvable.
- Proposes a precise definition of the fundamental notion of *algorithm*.
- Shows limitations of this notions.
- Introduces the concept of universal machine.

Beginnings of logic

- Frege proposes an universal formal language in which one can express all ordinary mathematics.
- Problem: Cantor's paradox
- Hilbert's program. Get rid of the worries about foundations of mathematics by:
  - simulating ordinary mathematics in a sufficiently strong formal system;
  - showing with elementary means ("finitist methods", which are not subject to doubt) that in this formal system a contradiction like $0 = 1$ could not be derived.
- The second point demands a procedure which can decide if a given formula is derivable in the calculus.
- This was shown impossible by Turing (and Church at the same time).
- But the first part of Hilbert's program was realized (ZFC).

The need of formal language

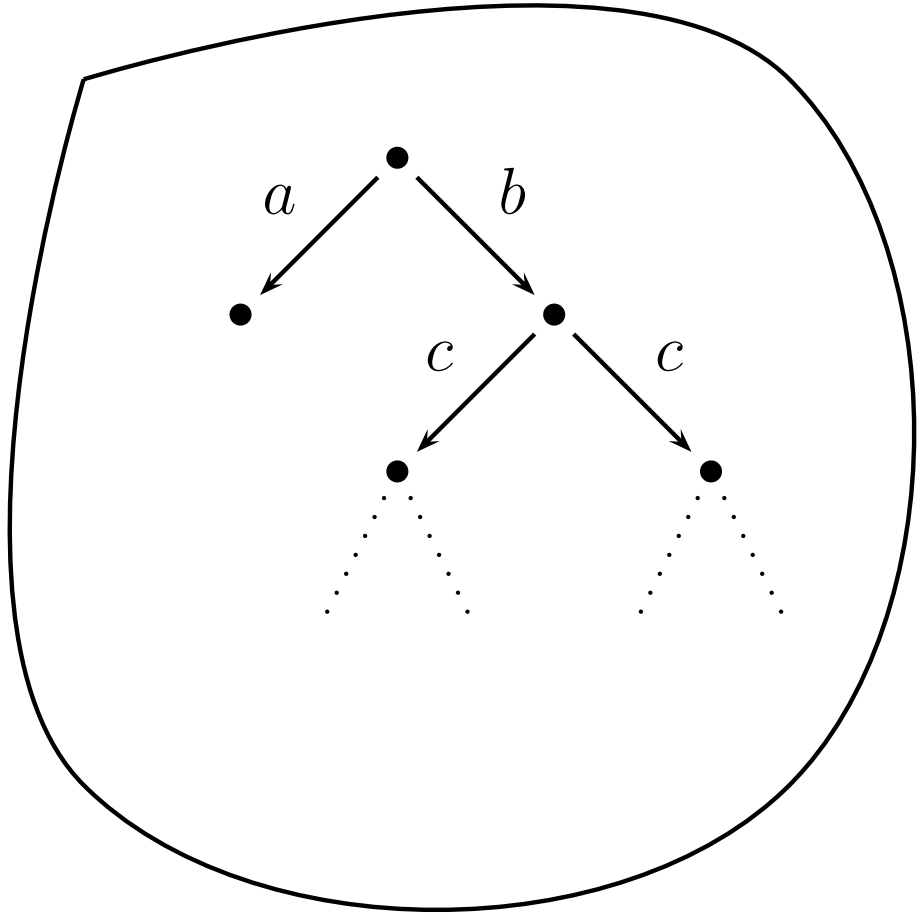
- Leibniz in XVII th century formulated a far-reaching vision of *characteristica universalis*:

> It should be possible to set up a kind of alphabet of human thoughts, and to invent and to decide everything by a combination of its letters and by analysis of the words composed from them.

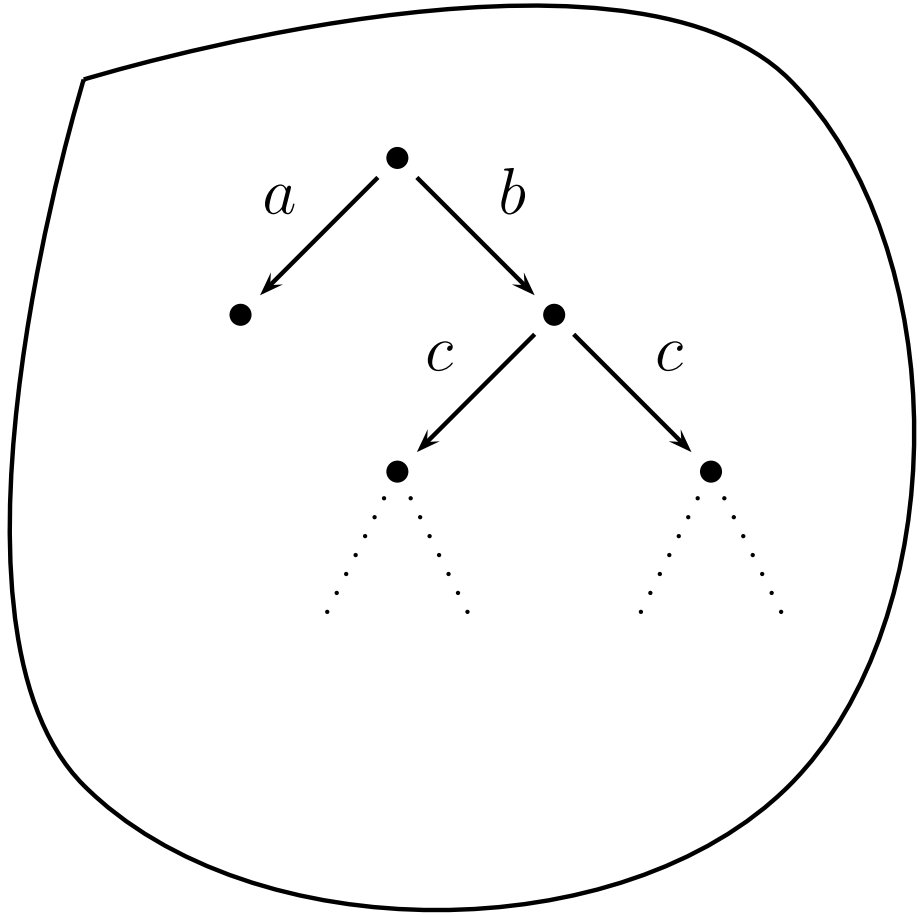
- In a sense ZFC satisfies this requirement. But it is arguably practical.
- In CS we want formal systems so computers can understand it.
- We prefer decidable systems: computer can decide if something is true.
- We are looking for something lighter (more specific) than ZFC.

Modeling computer systems

- A computer system can be anything: processor, communication protocol, airbag controller.
- A simple way of modelling such a system is by labelled graphs: $\langle S, \{R_a\}_{a \in Act} \rangle$, where $Act = \{a, b, \dots\}$ is some set of actions.



Modelling computer systems, cont.



- We need a decidable formal language.
- Where it is possible/easy to write interesting properties.
- Which has relatively low complexity of verification.

Plan

- Why logical formalisms.
- Two formalisms for model-checking.
- Advantages of formal systems.
- Model-checking as a game.
- Two player infinite games.
- Solving games.
- Special strategies in games.

MSOL on labelled graphs

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

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

- Models $\mathcal{M} = \langle S, \{R_a\}_{a \in Act} \rangle$.

- Semantics: $M, V \vDash \varphi$:

– $M, V \vDash y \in X$ if $V(y) \in V(X)$;

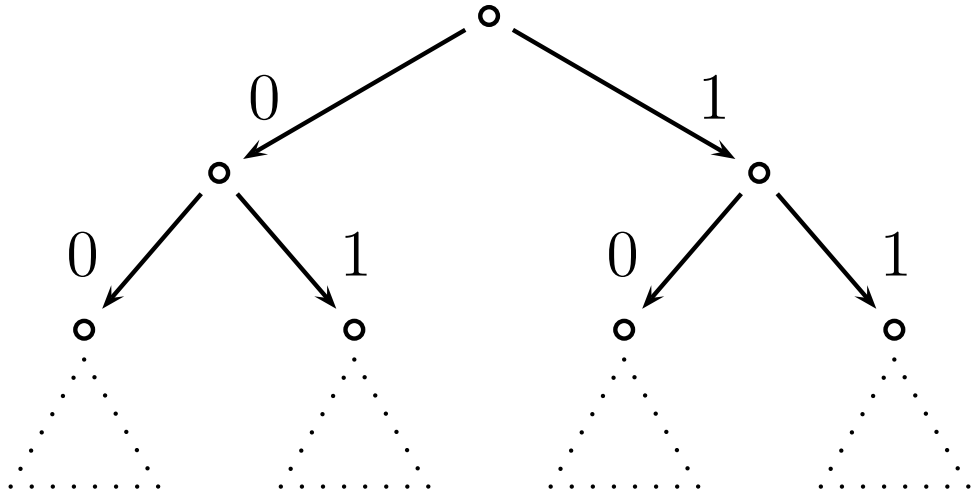
– $M, V \vDash \exists X.\varphi$ if there is $S \subseteq \{0,1\}^*$, s.t., $M, V[S/X] \vDash \varphi$.

- Set quantification allows to express many new properties:
  - connectivity,
  - reachability,
  - 3-colorability.

Problem: FOL theory of graphs is undecidable.

MSOL over trees

- A tree is graph where each node is reachable by a unique path from a distinguished node called the root.



Thm [Rabin'69]: The MSOL theory of trees is decidable.

Thm [Mayer'74]: The complexity of the MSOL theory of trees is not elementary.

- Formulas of the μ-calculus:

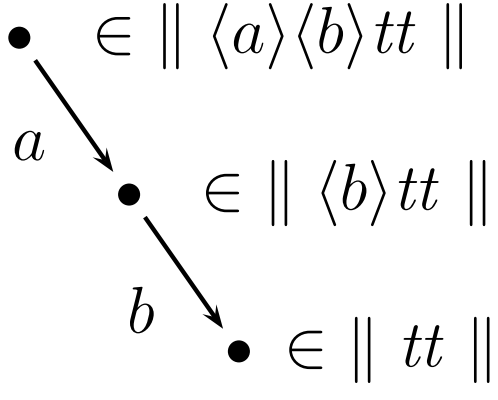
$$tt \mid ff \mid \alpha \vee \beta \mid \alpha \wedge \beta \mid \langle a \rangle \alpha \mid [a]\alpha \mid X \mid \mu X.\alpha(X) \mid \nu X.\alpha(X)$$

- Models $M = \langle S, \{R_a\}_{a \in Act} \rangle$.
- Semantics $\| \alpha \|_V^M$, $(V : Var \to \mathcal{P}(S))$

$$\| tt \|_V^M = S \qquad \| ff \|_V^M = \emptyset$$

$$\| \alpha \vee \beta \|_V^M = \| \alpha \|_V^M \cup \| \beta \|_V^M$$

$$\| \langle a \rangle \alpha \|_V^M = \{ s : \exists s'.\ R_a(s, s') \wedge s' \in \| \alpha \|_V^M \}$$



- Formulas of the μ-calculus:

$$tt \mid ff \mid \alpha \vee \beta \mid \alpha \wedge \beta \mid \langle a \rangle \alpha \mid [a]\alpha \mid X \mid \mu X.\alpha(X) \mid \nu X.\alpha(X)$$

- Models $M = \langle S, \{R_a\}_{a \in Act} \rangle$.
- Semantics $\| \alpha \|_V^M$, $(V : Var \to \mathcal{P}(S))$

$$\| tt \|_V^M = S \qquad \| ff \|_V^M = \emptyset$$

$$\| \alpha \vee \beta \|_V^M = \| \alpha \|_V^M \cup \| \beta \|_V^M$$

$$\| \langle a \rangle \alpha \|_V^M = \{ s : \exists s'.\ R_a(s, s') \wedge s' \in \| \alpha \|_V^M \}$$

$$\| X \|_V^M = V(X)$$

$$\| \mu X.\alpha(X) \|_V^M = \bigcap \{ S' \subseteq S : \| \alpha \|_{V[S'/X]}^M \subseteq S' \}$$

The meaning of the fixpoint

- In $\mu X.\alpha(X)$ variable X appears only positively in $\alpha(X)$.
- The formula $\alpha(X)$ defines a function:

$$\lambda X.\ \| \alpha(X) \|_V^M \ : \ \mathcal{P}(S) \to \mathcal{P}(S)$$

- The function is monotonic:

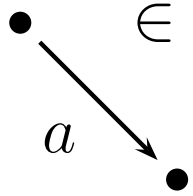
$$\| \alpha(B_1) \|_V^M \subseteq \| \alpha(B_2) \|_V^M \quad \text{if } B_1 \subseteq B_2.$$

- There are the least and the greatest fixpoints of this function:

$$\| \mu X.\alpha(X) \|_M^V = \bigcap \{B \subseteq S : \| \alpha(B) \|_M^V \subseteq B\}$$

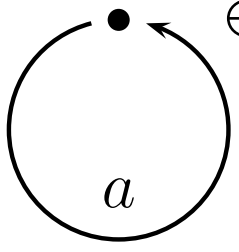
$$\| \nu X.\alpha(X) \|_M^V = \bigcup \{B \subseteq S : B \subseteq \| \alpha(B) \|_M^V\}$$

Examples



$\in [\![\langle a \rangle tt]\!]$

- $\| \mu X.\langle a \rangle X \| = \emptyset$



$\in \| \nu X.\langle a \rangle X \|$

Examples

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

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

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

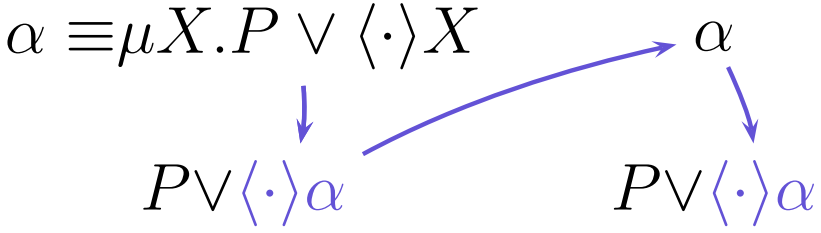
- Reachability: $\langle\cdot\rangle^* P \equiv \mu X.\ P \vee \langle\cdot\rangle X$.

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

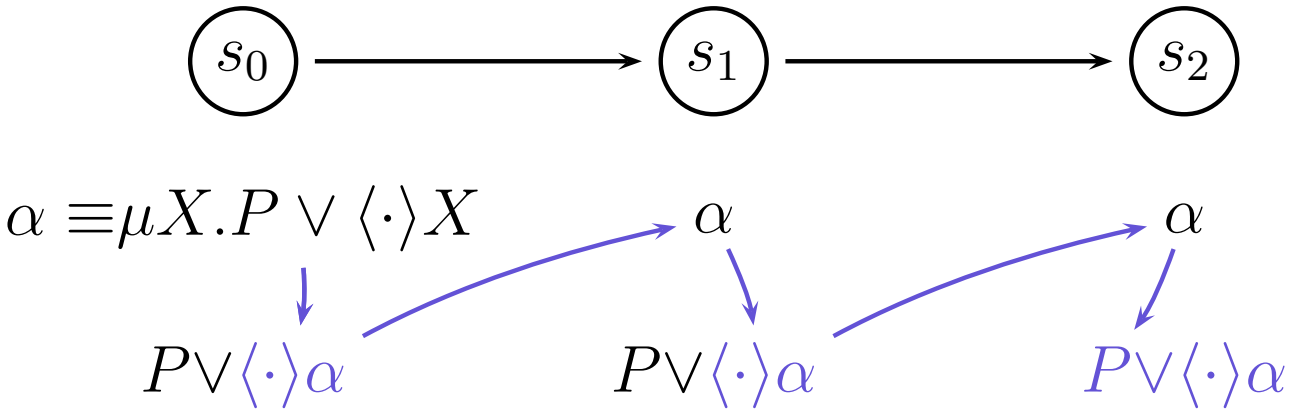
$\downarrow$

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

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



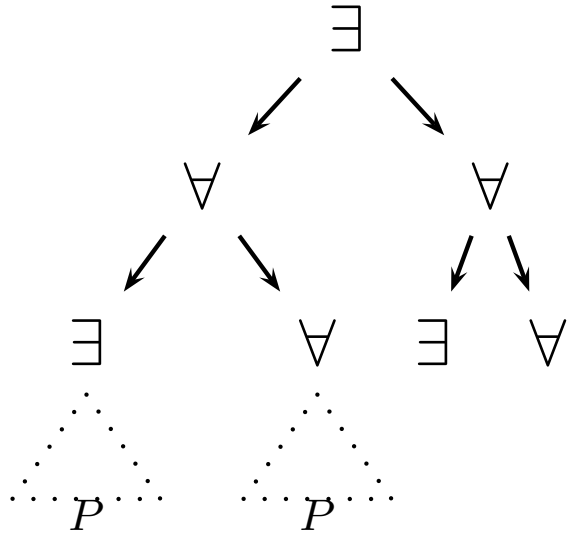
- Reachability: $\langle\cdot\rangle^* P \equiv \mu X.\ P \vee \langle\cdot\rangle X$.



More examples

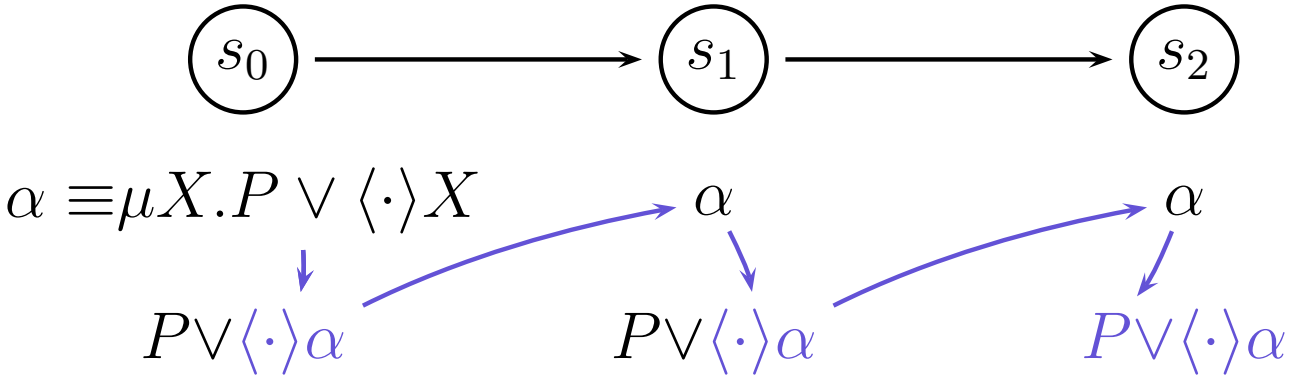
- Reachability: $\langle\cdot\rangle^* P \equiv \mu X.\ P \vee \langle\cdot\rangle X$.
- Existential until: $\exists(Q \mathbb{U} P) \equiv \mu X.\ P \vee (Q \wedge \langle\cdot\rangle X)$.
- Universal until: $\forall(Q \mathbb{U} P) \equiv \mu X.\ P \vee (Q \wedge [\,] X)$.
- Alternating reachability

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



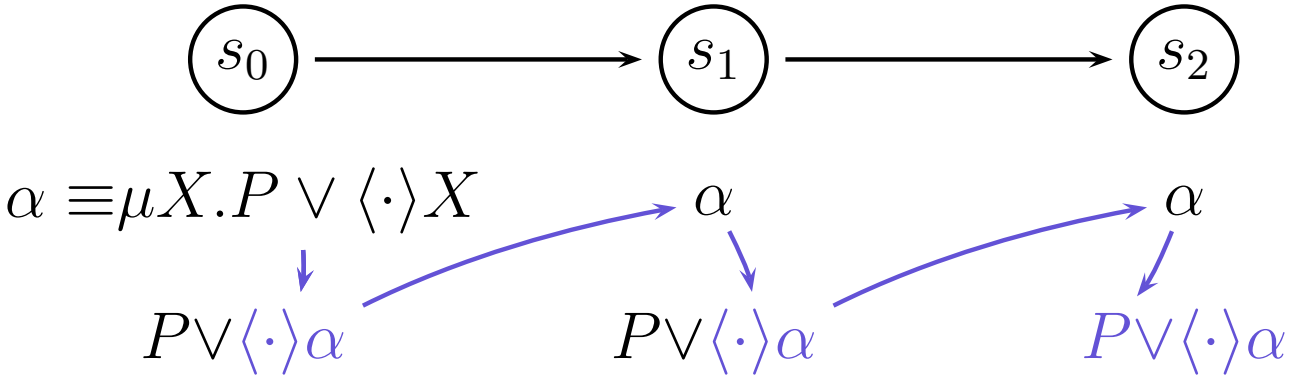
Examples (μ vs. ν)

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

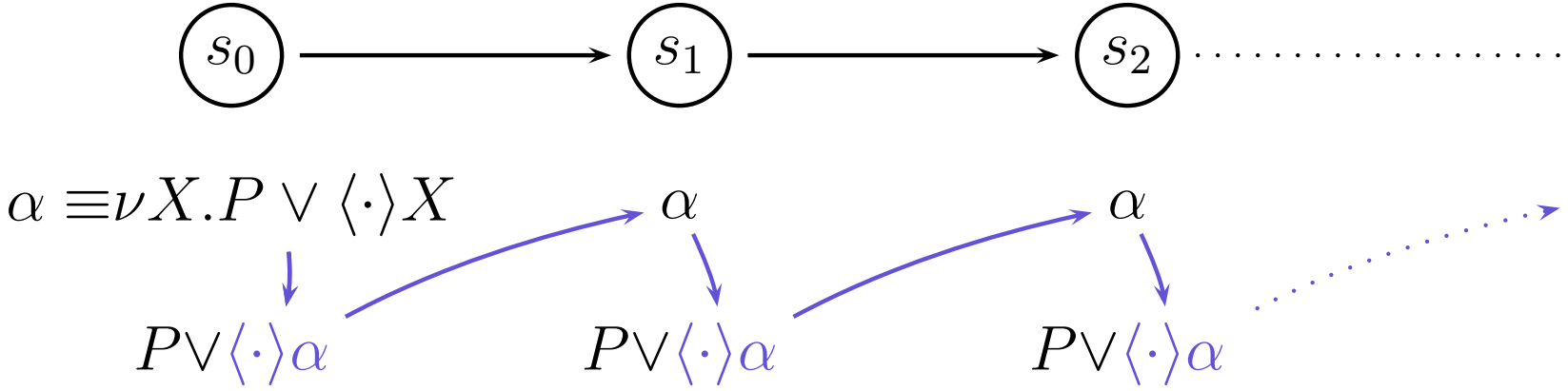


Examples (μ vs. ν)

- $\mu X.\ P \vee \langle\cdot\rangle X$ 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

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

Plan

- Why logical formalisms.
- Two formalisms for model-checking.
- Advantages of formal systems.
- Model-checking as a game.
- Two player infinite games.
- Solving games.
- Special strategies in games.

Comparing MSOL and the μ-calculus

- We can compare the two formalisms for defining graph properties.

- A μ-calculus sentence α defines a property: class of pairs (M, s) such that $s \in \| \alpha \|^M$.

- An MSOL formula with one free variable $\varphi(x)$ defines a property: class of pairs (M, s) such that $M \models \varphi(s)$.

- We can compare which properties are definable in MSOL and which in the μ-calculus.

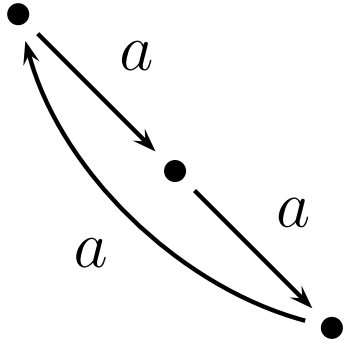
Why the μ-calculus?

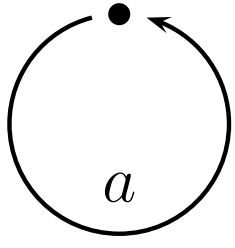
Thm: The μ-calculus theory of labeled graphs is EXPTIME-complete.(not decidable for MSOL)

Why the μ-calculus?

Thm: The μ-calculus theory of labeled graphs is EXPTIME-complete.(not decidable for MSOL)

Racall : Two nodes of a (deterministic) transition system are in bisimulation iff unwindings from these two nodes are isomorphic.





Why the μ-calculus?

Thm: The μ-calculus theory of labeled graphs is EXPTIME-complete.(not decidable for MSOL)

Racall : Two nodes of a (deterministic) transition system are in bisimulation iff unwindings from these two nodes are isomorphic.

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

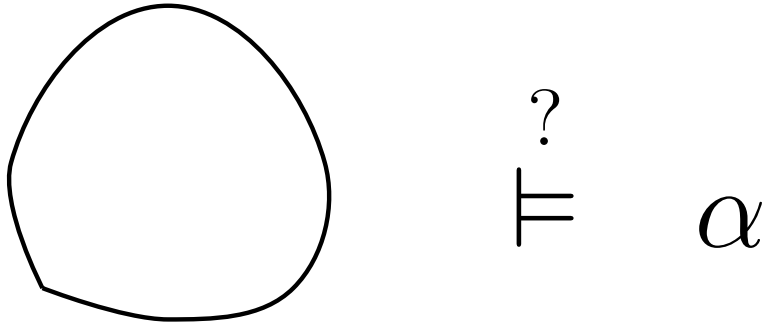
Thm [Janin & W.]: A property is expressible in the μ-calculus iff it is expressible in MSOL and bisimulation invariant.

- It is not decidable if an MSOL formula is bisimulation invariant.

Plan

- Why logical formalisms.
- Two formalisms for model-checking.
- Advantages of formal systems.
- Model-checking as a game.
- Two player infinite games.
- Solving games.
- Special strategies in games.

The model checking problem

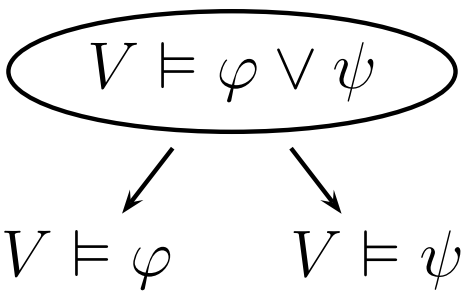


Given a finite labelled graph M, its vertex s and a sentence α, decide if $s \in \| \alpha \|^M$.

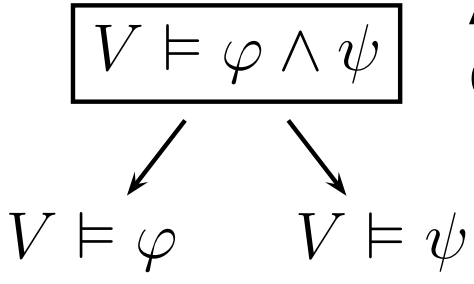
This problem reduces to a problem of solving games.

Propositional logic (model checking)

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



Eve chooses



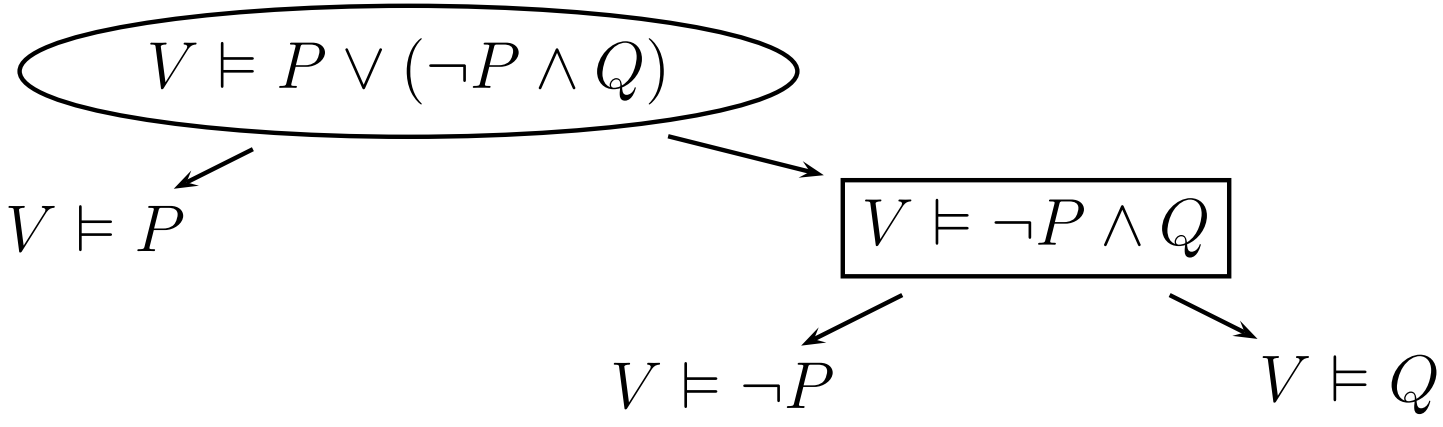
Adam chooses

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

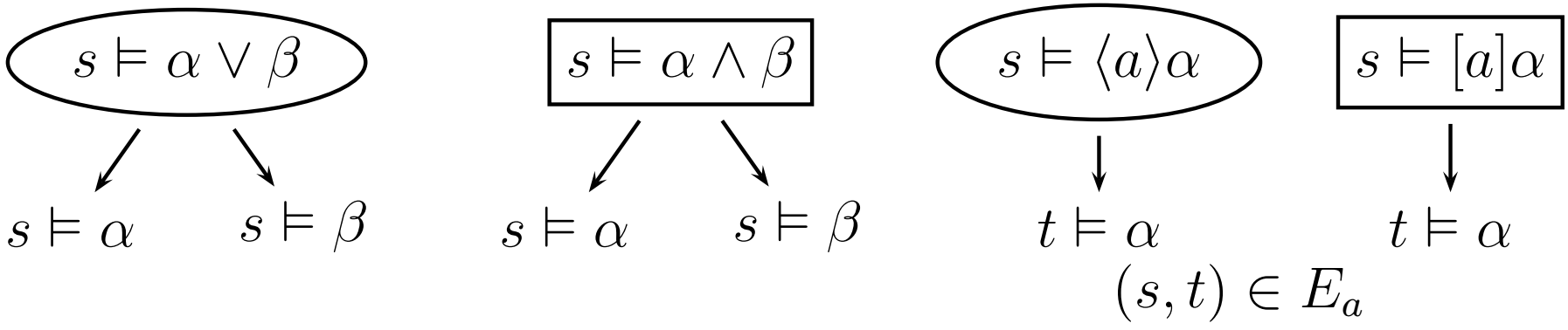
- Eve has a winning strategy from $V \vDash \varphi$ iff φ is true in V.

Example

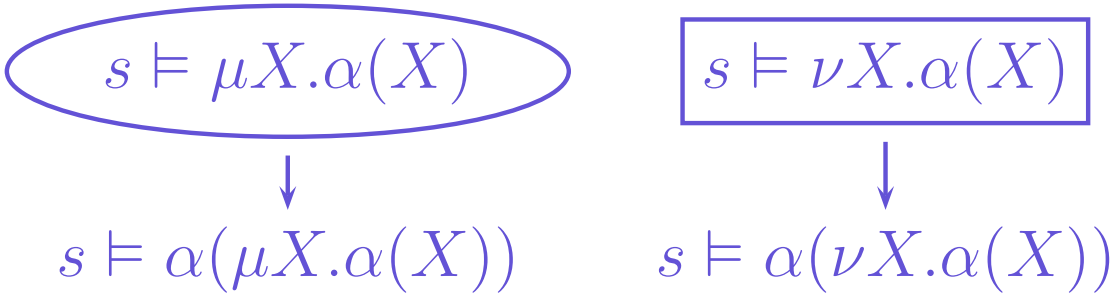
$$V \equiv (P = 0, Q = 1)$$



- We are given a transition system $\mathcal{M}$ and a formula α_0.
- Model checking rules



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



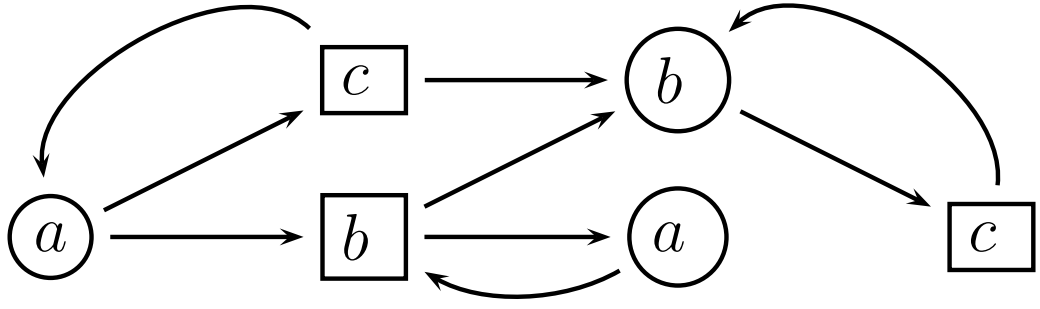
- The last two rules may be a source of infinite plays.

Plan

- Why logical formalisms.
- Two formalisms for model-checking.
- Advantages of formal systems.
- Model-checking as a game.
- Two player infinite games.
- Solving games.
- Special strategies in games.

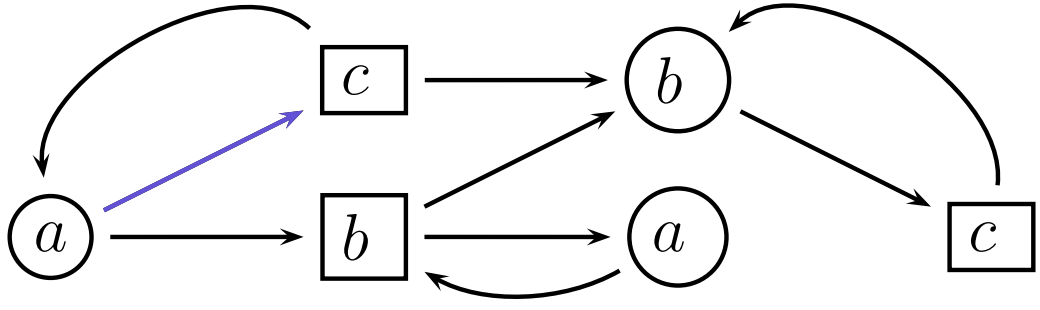
Path forming games

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



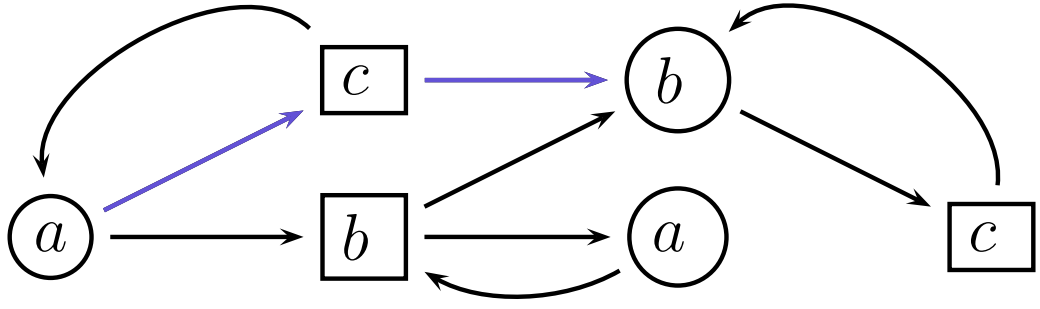
Path forming games

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



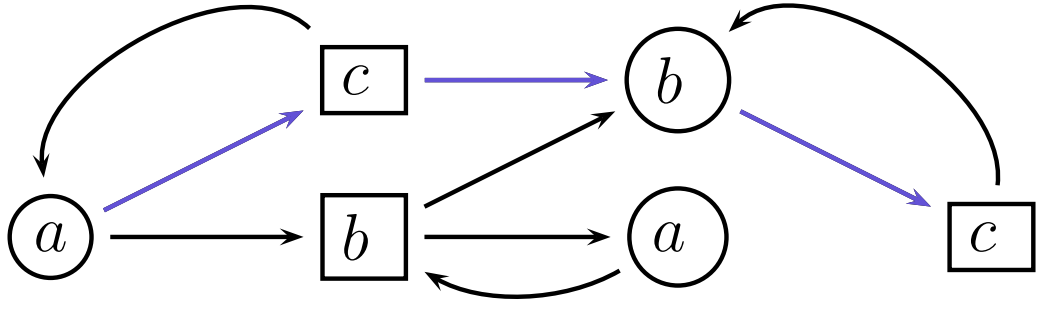
Path forming games

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



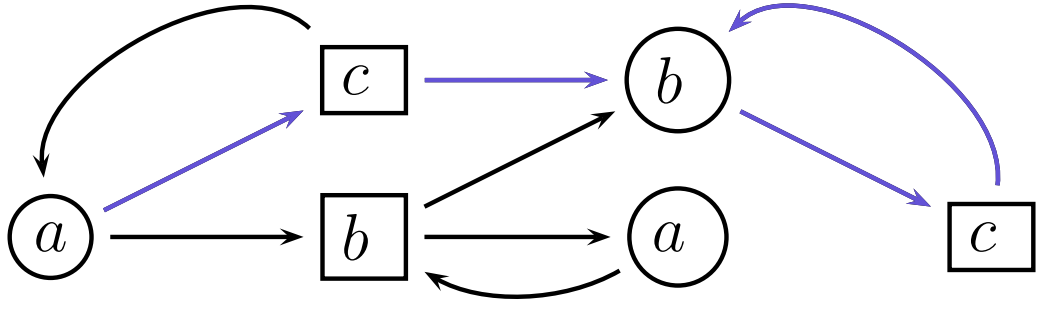
Path forming games

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

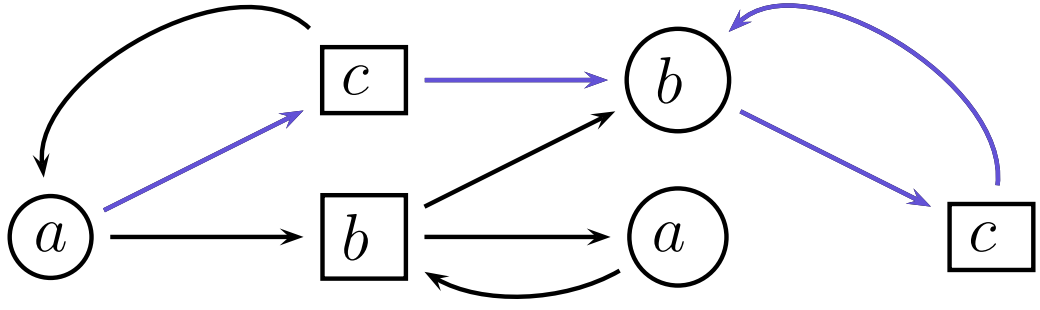


Path forming games

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

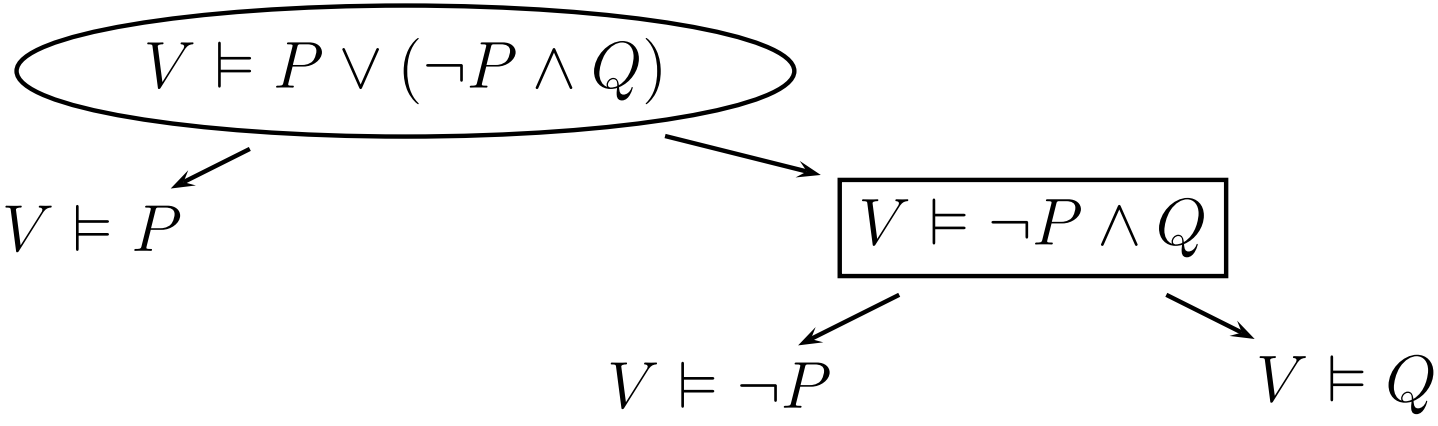


$$\mathcal{G} = \langle V_E,\ V_A,\ R,\ \lambda : V \to C,\ Acc \subseteq C^\omega \rangle$$

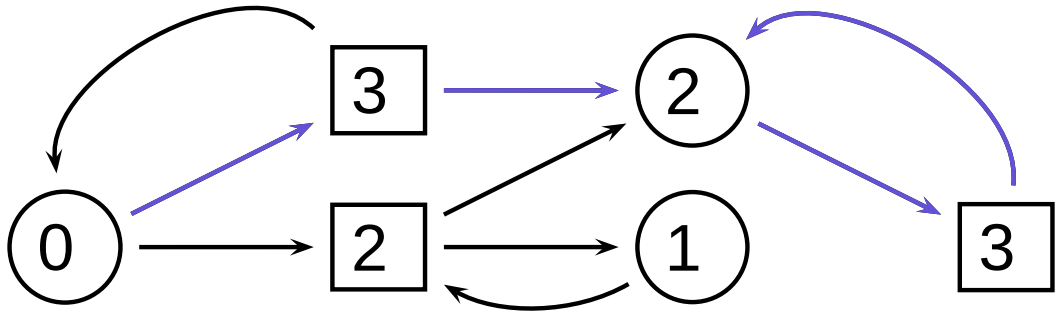


- Eve wins if the labeling of the path is in Acc.

  (There is an edge from every node.)



Winning conditions



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

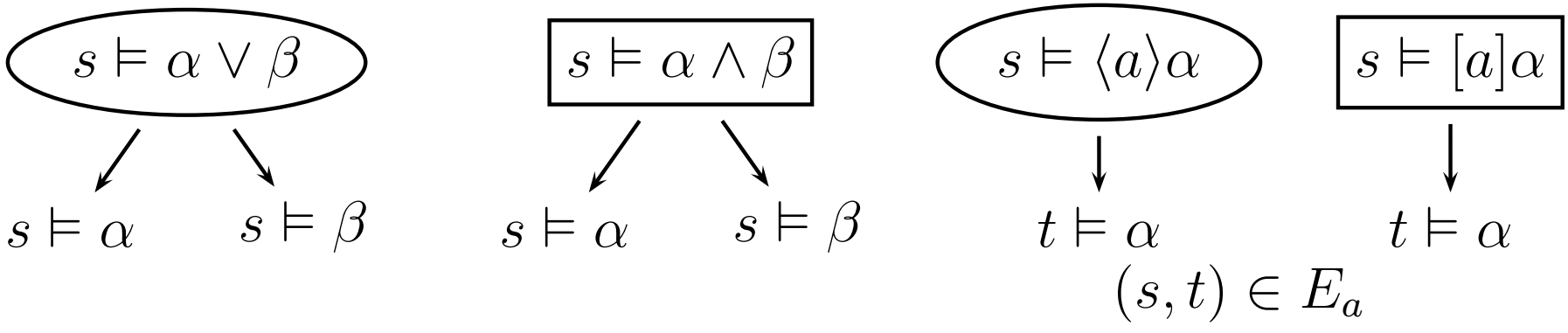
- $\mathrm{Inf}(\vec{v})$: the set of colours appearing infinitely often on a path $\vec{v}$.
- Muller condition: given by a partition of $\mathcal{P}(C)$ into $(\mathcal{F}_E, \mathcal{F}_A)$.

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

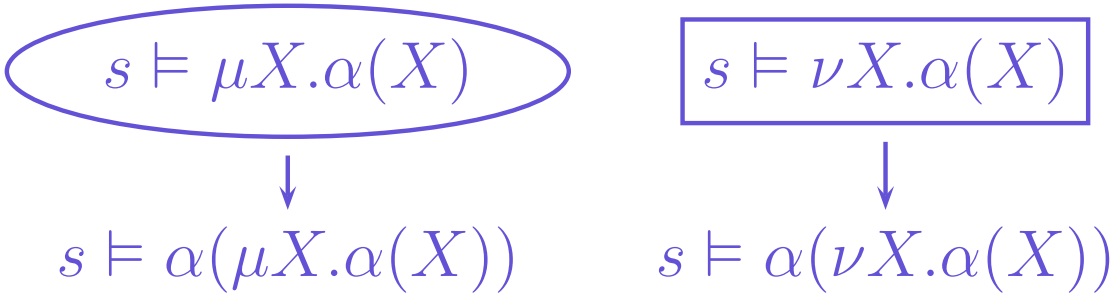
- Parity condition is given by a function $\Omega : V \to \{0, \ldots, d\}$.

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

- We are given a transition system $\mathcal{M}$ and a formula α_0.
- Model checking rules

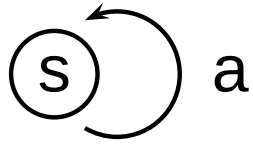


$s \models P$ Eve wins if $s \in P^{\mathcal{M}}$; $s \models \neg P$ Eve wins if $s \notin P^{\mathcal{M}}$.



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

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.

Approximations

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

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

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

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

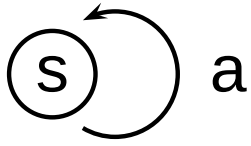
$$\nu X.\beta(X) = \bigcap_{\tau \in Ord} \nu^{\tau} X.\beta(X)$$

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

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

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

Infinite plays

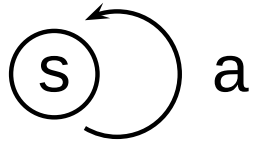


$$s \vDash \ \mu^{\tau} X.\langle a \rangle X$$
$$\downarrow$$
$$s \vDash \ \langle a \rangle (\mu^{\tau - 1} X.\langle a \rangle X)$$
$$\downarrow$$
$$s \vDash \ \mu^{\tau - 1} X.\langle a \rangle X$$
$$\downarrow$$
$$\vdots$$

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

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

Infinite plays

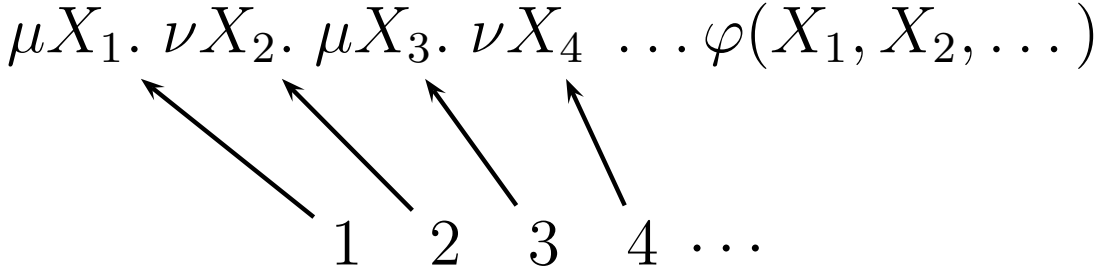


$$s \vDash {}_3\mu^{\tau} X.\langle a\rangle X \qquad s \vDash {}_3\nu^{\tau} X.\langle a\rangle X$$
$$\downarrow \qquad\qquad \downarrow$$
$$s \vDash {}_1\langle a\rangle(\mu^{\tau-1} X.\langle a\rangle X) \qquad s \vDash {}_2\langle a\rangle(\nu^{\tau} X.\langle a\rangle X)$$
$$\downarrow \qquad\qquad \downarrow$$
$$s \vDash {}_3\mu^{\tau-1} X.\langle a\rangle X \qquad s \vDash {}_3\nu^{\tau} X.\langle a\rangle X$$
$$\downarrow \qquad\qquad \downarrow$$
$$\vdots \qquad\qquad \vdots$$

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

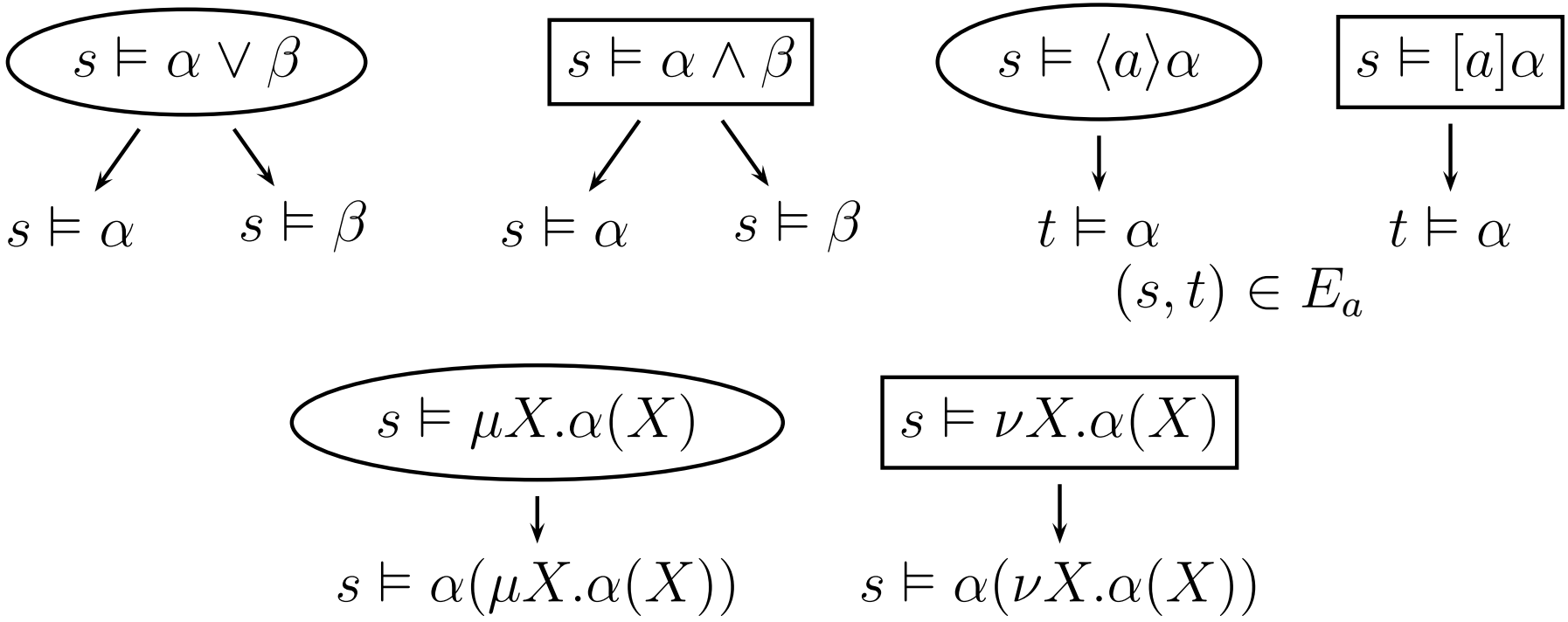
Assign rank 1 to μ-regeneration and rank 2 to ν-regeneration.

Defining winning conditions



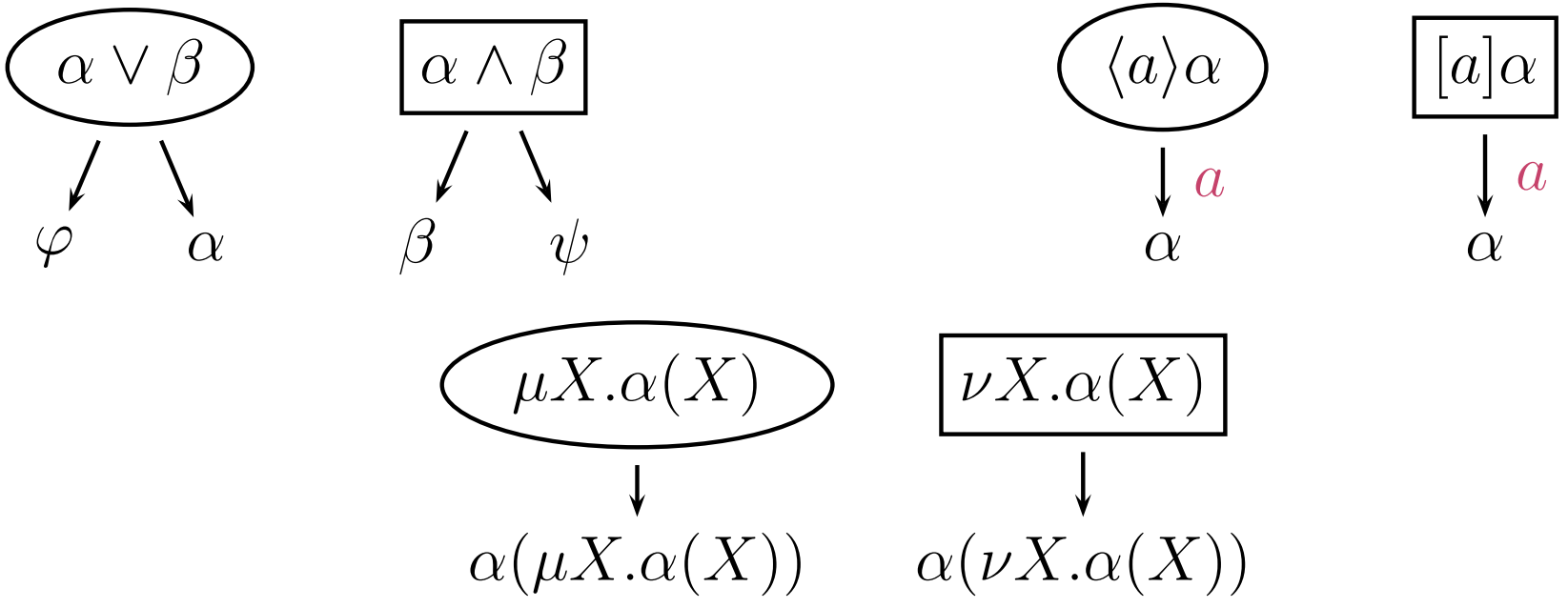
- μ's have odd ranks,
- ν's have even ranks,
- if β is a subformula of α then β has bigger rank than α.

- 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

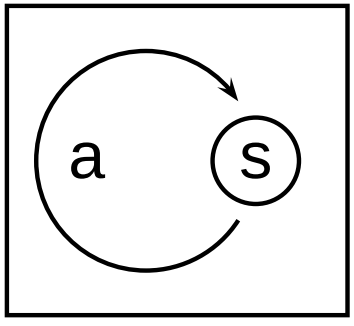


- These rules define a tableau $\mathcal{T}_\alpha$ for a formula α.
- Operation $\mathcal{M} \otimes \mathcal{T}_\alpha$ of "synchronized product" of a transition system and a tableau that gives the MC game.

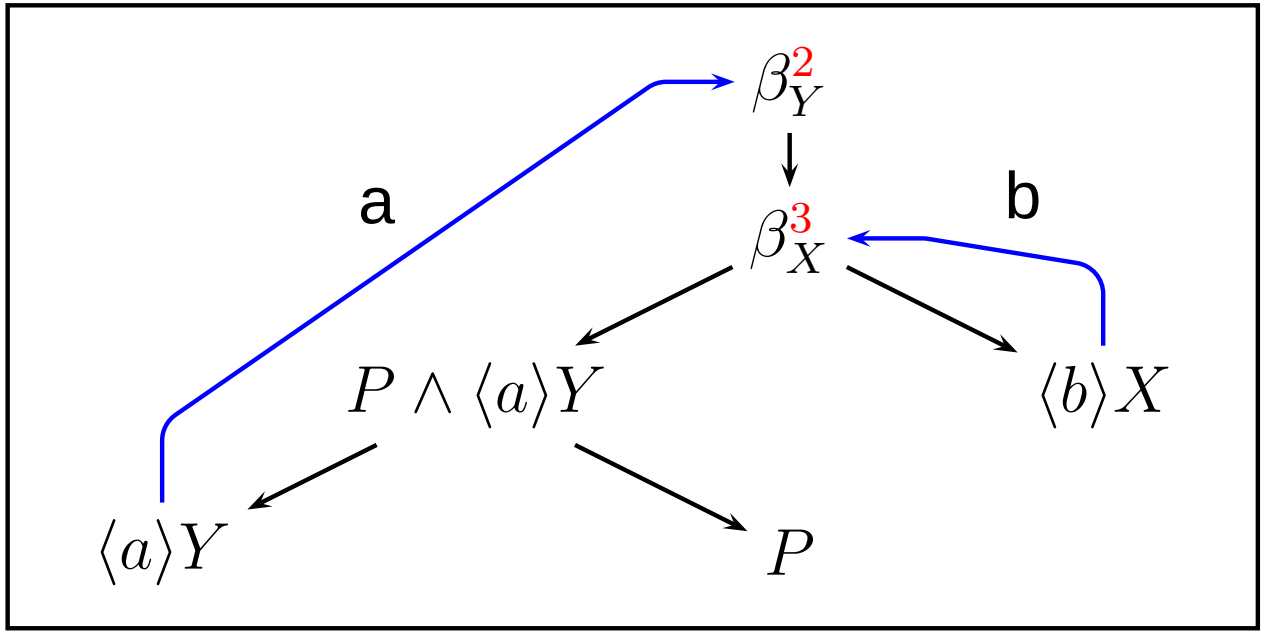
Obs: $\mathcal{M}, s_0 \vDash \alpha$ iff Eve wins from (s_0, α) in $\mathcal{M} \otimes \mathcal{T}_\alpha$.

Example

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



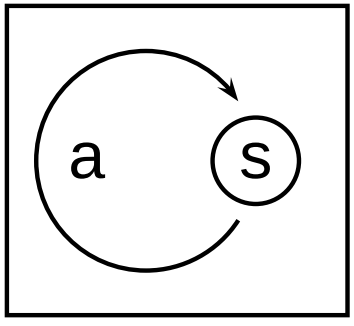
$\mathcal{M}$



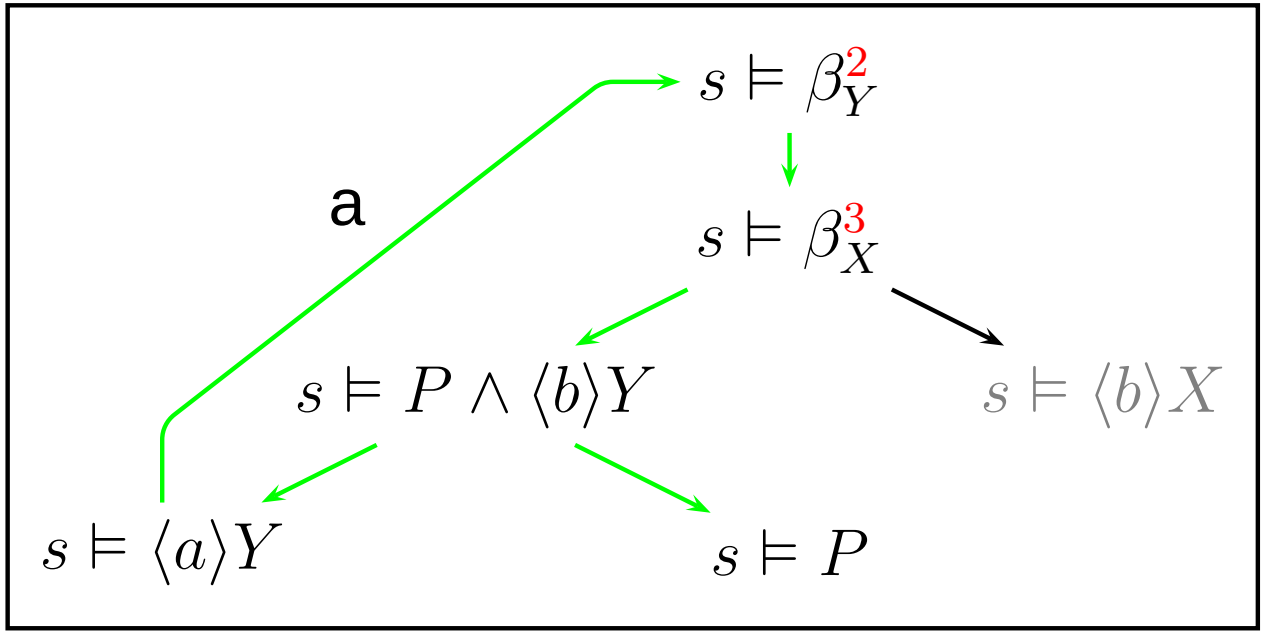
$\mathcal{T}_{\beta_Y}$

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 define 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
- propositions P_E designates Eve's positions,
- propostions $P_0, \ldots, P_d$ define $\Omega : V \to \{0, \ldots, d\}$.

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

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

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

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

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

In summary

- Parity games and model-checking for the mu-calculus are very close to each other (inter-reducible in linear time).
- 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 satisfiabiality 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.
- Because of this translation it is enough to consider the games solving problem instead of MC problem.

Plan

- Why logical formalisms.
- Two formalisms for model-checking.
- Advantages of formal systems.
- Model-checking as a game.
- Two player infinite games.
- Solving games.
- Special strategies in games.

Open problem: Solving parity games

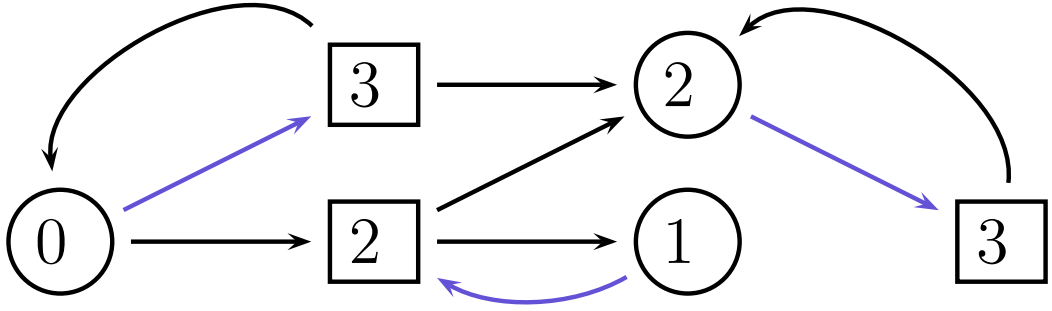
- Given a finite parity game $G = \langle V_E, V_a, R, \Omega : (V_E \cup V_a) \to \mathbb{N} \rangle$ decide if Eve has a winning strategy from a given position.

- This problem is in NP and its complement is also in NP. We do not know if the problem is in PTIME.

- This is one of the very few problems that have this status

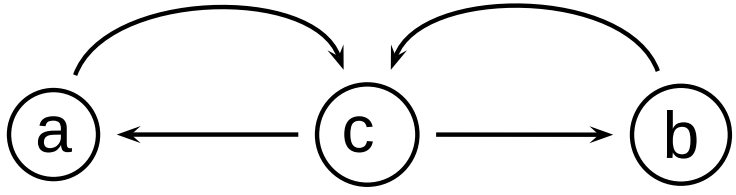
$$\mathcal{G} = \langle V_E,\ V_A,\ R,\ \lambda : V \to C,\ Acc \subseteq C^\omega \rangle$$

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

Strategy with memory: example



- Muller condition: $\mathcal{F}_E = \{\{a, b, c\}\}$.

  (Both a and b appear infinitely often.)

- Eve has a winning strategy in this game but no positional winning strategy.

Basic results

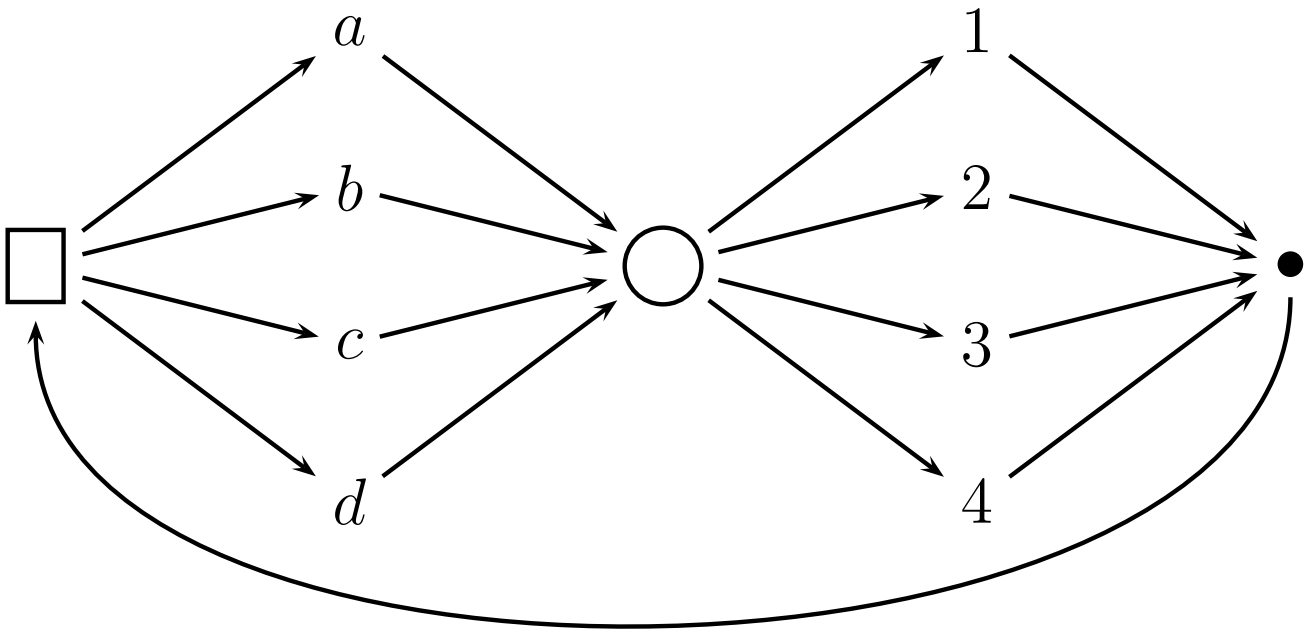
Thm [Martin]: Every game with a Muller winning condition is determined, i.e., from every vertex one of the players has a winning strategy.

Thm [Mostowski, Emerson & Jutla]: 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.

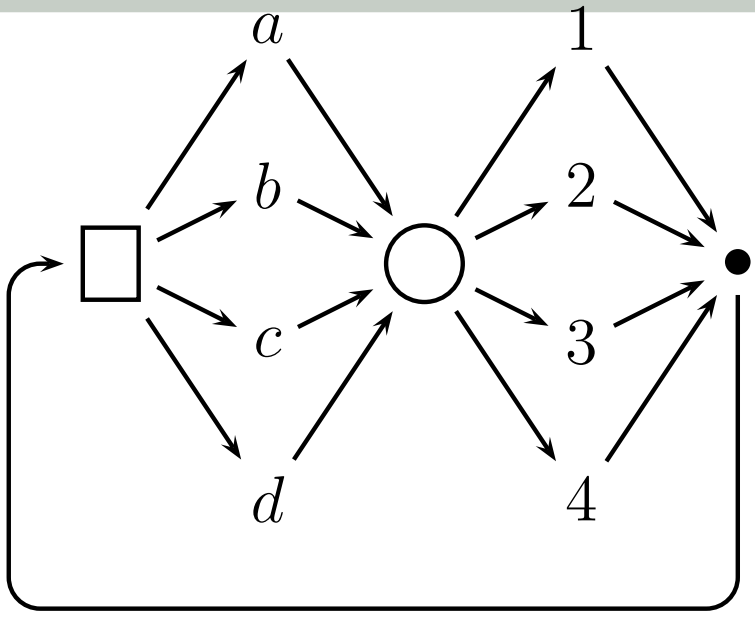
Fact : There is an algorithm for solving finite Muller games.

A funny game



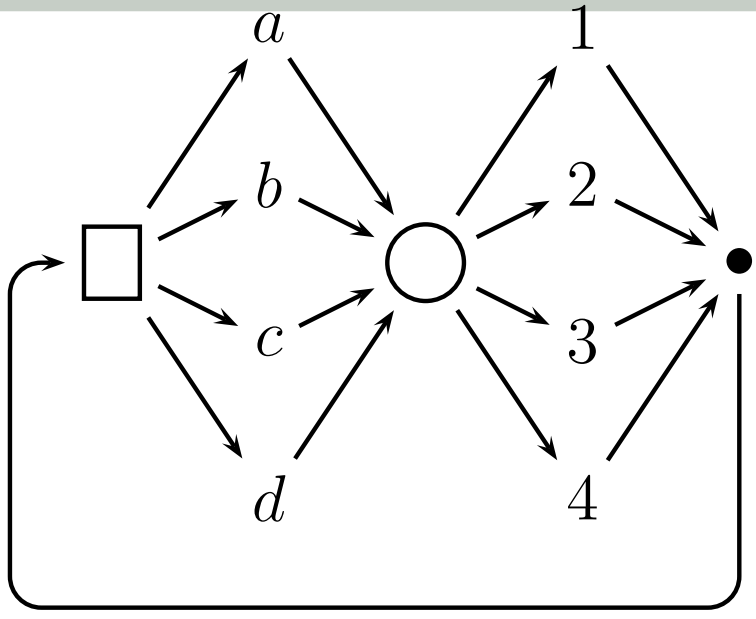
- The biggest number seen infinitely often $=$ the number of letters seen infinitely often.
- Examples: $a1a1a1\ldots$, $a1c1c1c1\ldots$, $a1c2a1c1a2c2\ldots$.
- Eve has a winning strategy with finite memory in this game.

LAR example



a b c <u>d</u> $\longrightarrow 1$

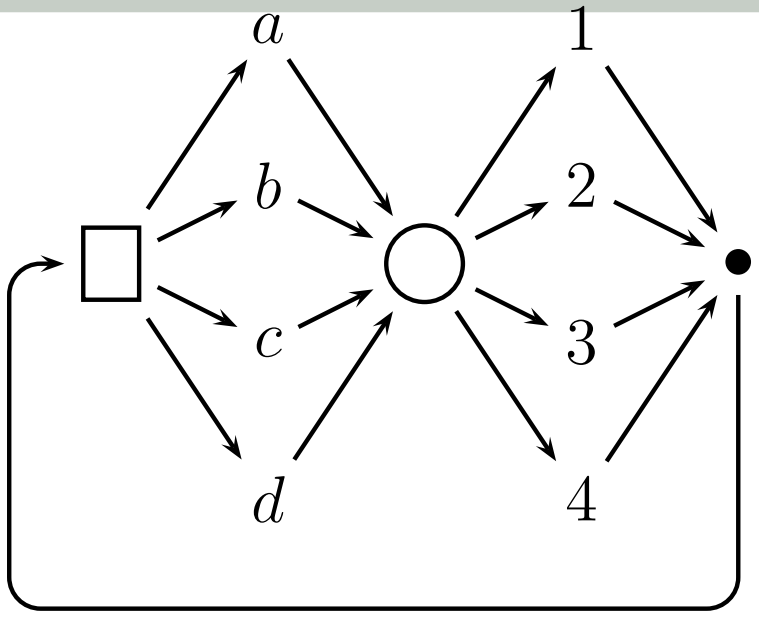
LAR example



a b c <u>d</u> $\longrightarrow 1$

b

LAR example

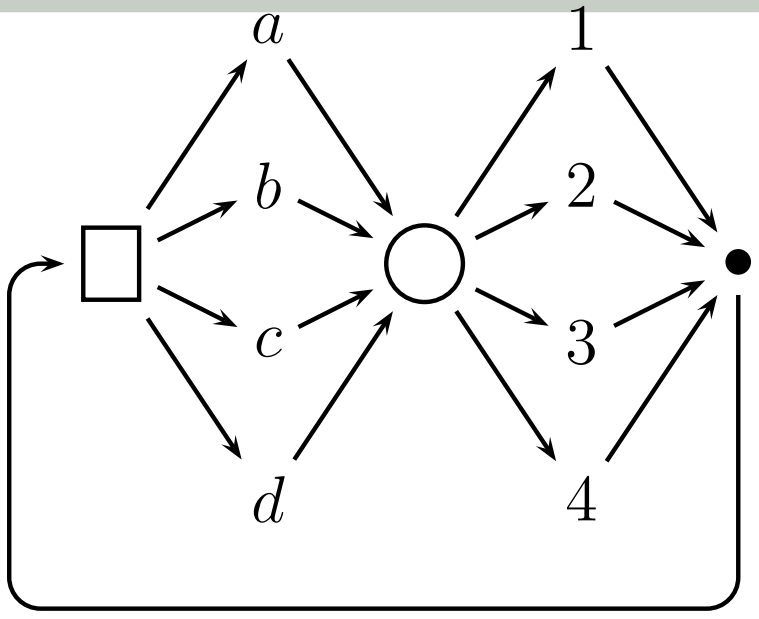


a b c <u>d</u> $\longrightarrow 1$

b

a <u>c</u> d b $\longrightarrow 3$

LAR example



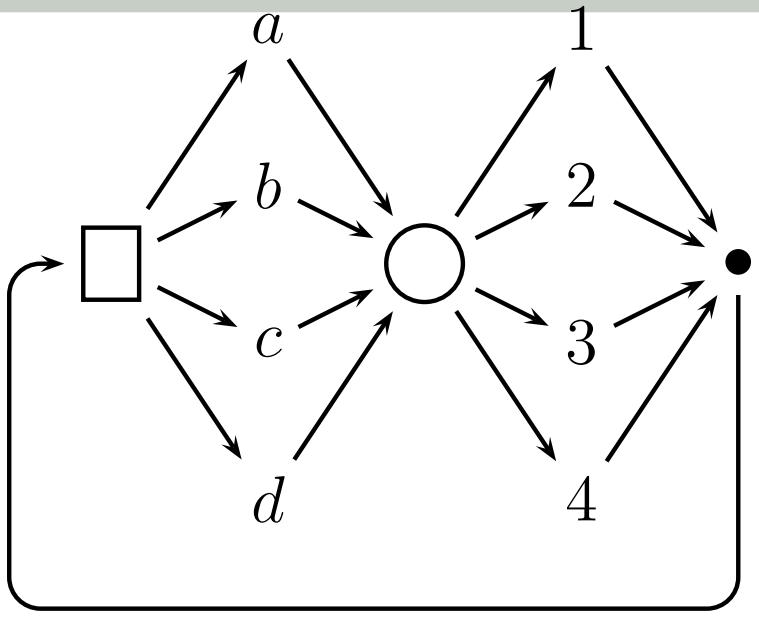
a b c <u>d</u> $\rightarrow 1$

b

a <u>c</u> d b $\rightarrow 3$

d

LAR example



a b c <u>d</u> $\longrightarrow 1$

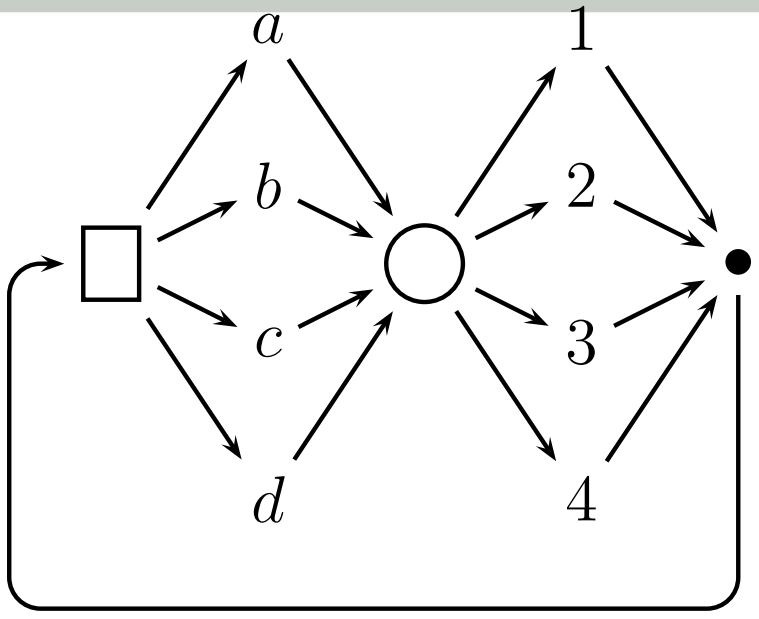
b

a <u>c</u> d b $\longrightarrow 3$

d

a c <u>b</u> d $\longrightarrow 2$

LAR example



a b c $\underline{\text{d}}$ $\longrightarrow 1$

b

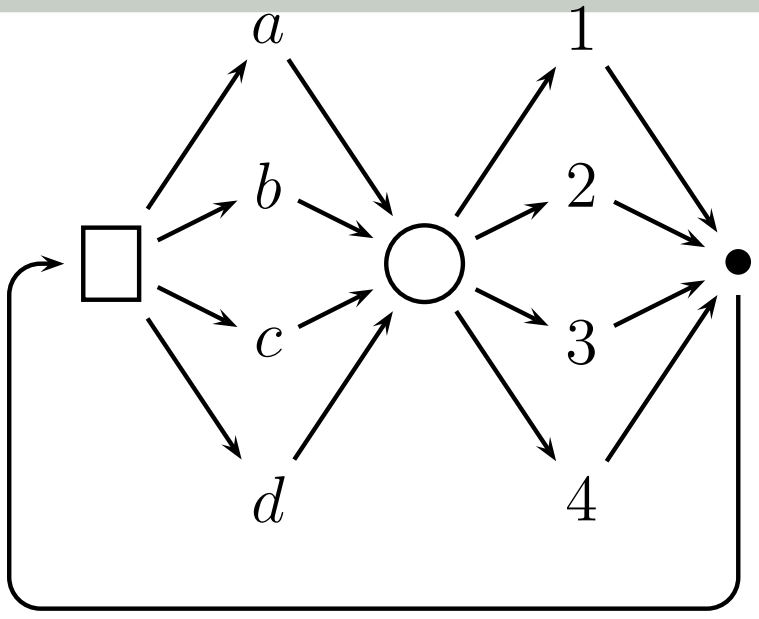
a $\underline{\text{c}}$ d b $\longrightarrow 3$

d

a c $\underline{\text{b}}$ d $\longrightarrow 2$

a

LAR example



a b c <u>d</u> $\rightarrow 1$

b

a <u>c</u> d b $\rightarrow 3$

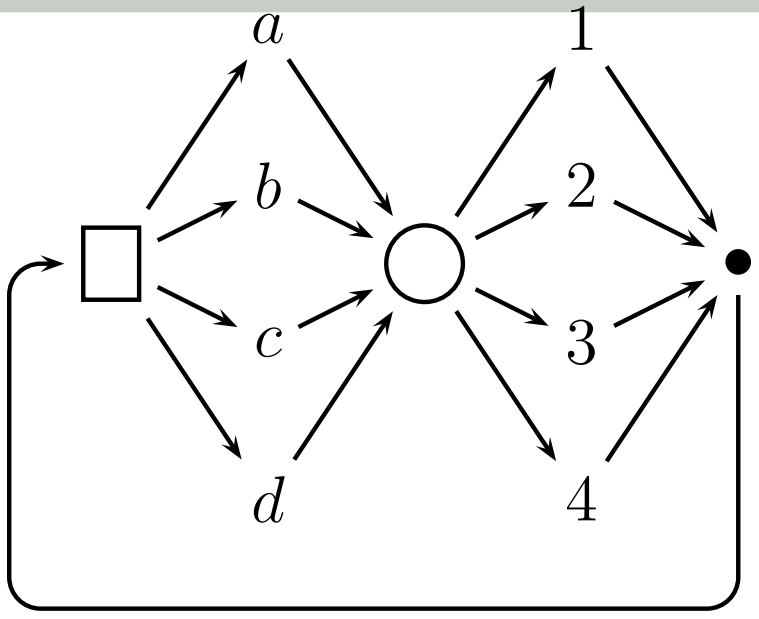
d

a c <u>b</u> d $\rightarrow 2$

a

<u>c</u> b d a $\rightarrow 4$

LAR example



a b c <u>d</u> $\rightarrow 1$

b

a <u>c</u> d b $\rightarrow 3$

d

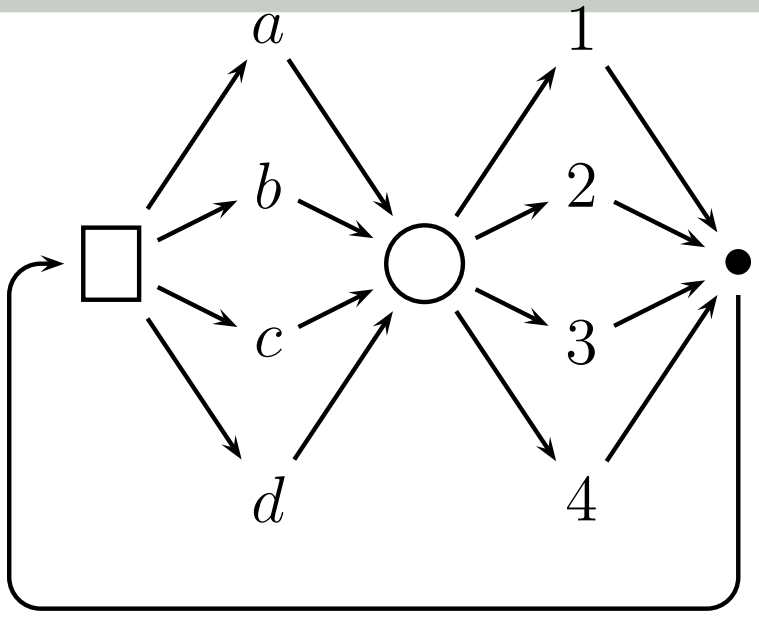
a c <u>b</u> d $\rightarrow 2$

a

<u>c</u> b d a $\rightarrow 4$

d

LAR example



a b c <u>d</u> $\rightarrow 1$

b

a <u>c</u> d b $\rightarrow 3$

d

a c <u>b</u> d $\rightarrow 2$

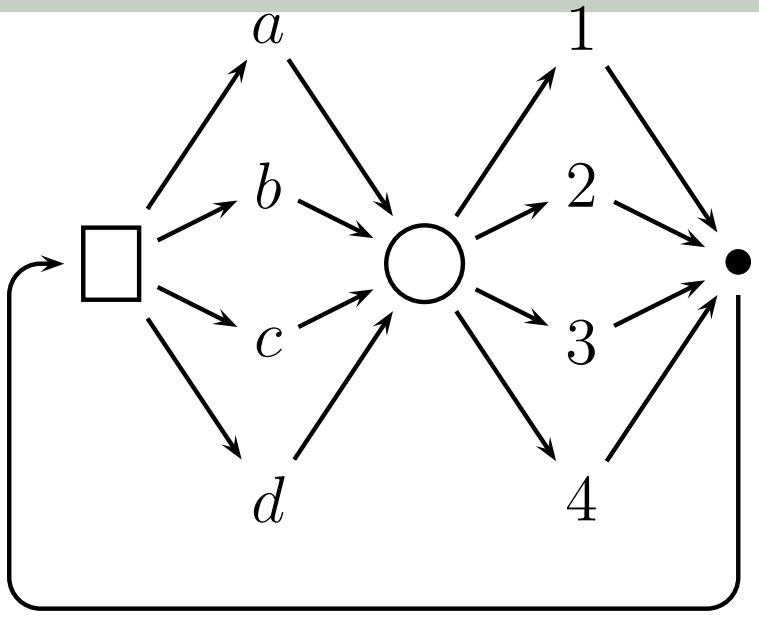
a

<u>c</u> b d a $\rightarrow 4$

d

c b <u>a</u> d $\rightarrow 2$

LAR example



a b c <u>d</u> $\rightarrow 1$

b

a <u>c</u> d b $\rightarrow 3$

d

a c <u>b</u> d $\rightarrow 2$

a

<u>c</u> b d a $\rightarrow 4$

d

c b <u>a</u> d $\rightarrow 2$

d

LAR example

a b c <u>d</u> $\rightarrow 1$

b

a <u>c</u> d b $\rightarrow 3$

d

a c <u>b</u> d $\rightarrow 2$

a

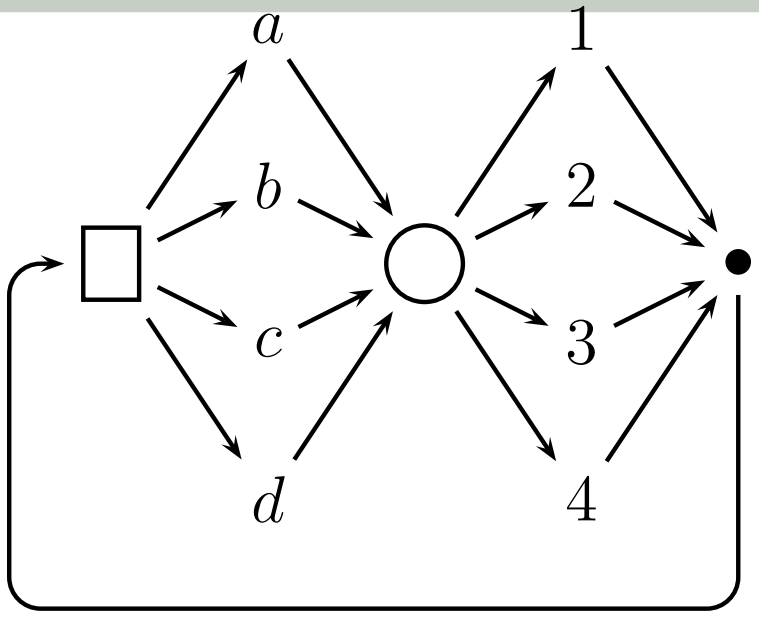
<u>c</u> b d a $\rightarrow 4$

d

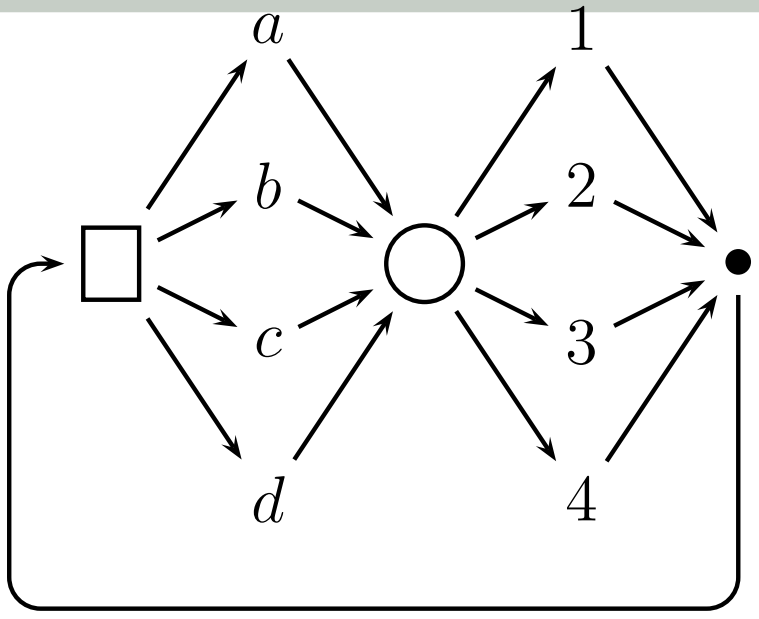
c b <u>a</u> d $\rightarrow 2$

d

c b a <u>d</u> $\rightarrow 1$



LAR example



a b c <u>d</u> $\rightarrow 1$

b

a <u>c</u> d b $\rightarrow 3$

d

a c <u>b</u> d $\rightarrow 2$

a

<u>c</u> b d a $\rightarrow 4$

d

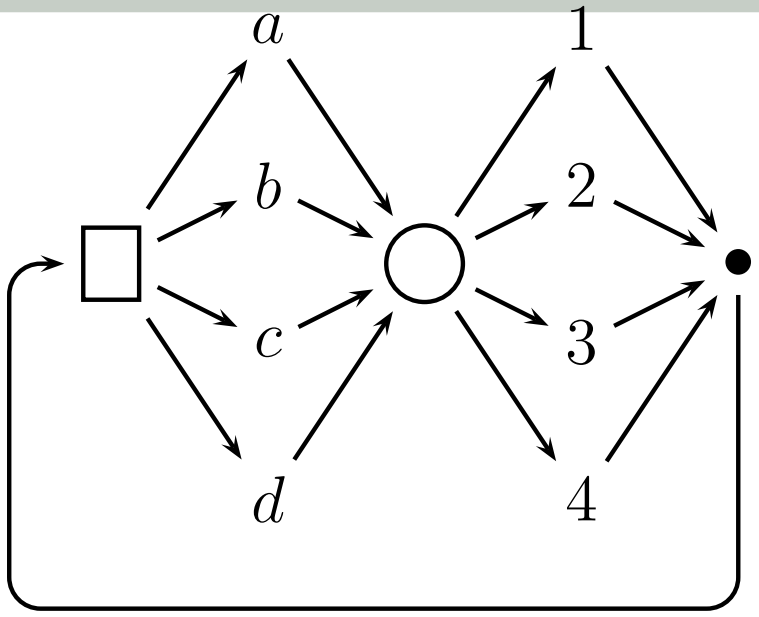
c b <u>a</u> d $\rightarrow 2$

d

c b a <u>d</u> $\rightarrow 1$

a

LAR example



a b c <u>d</u>	$\rightarrow 1$	
		b
a <u>c</u> d b	$\rightarrow 3$	
		d
a c <u>b</u> d	$\rightarrow 2$	
		a
<u>c</u> b d a	$\rightarrow 4$	
		d
c b <u>a</u> d	$\rightarrow 2$	
		d
c b a <u>d</u>	$\rightarrow 1$	
		a
c b <u>d</u> a	$\rightarrow 2$	

LAR example

a b c <u>d</u> $\rightarrow 1$

b

a <u>c</u> d b $\rightarrow 3$

d

a c <u>b</u> d $\rightarrow 2$

a

<u>c</u> b d a $\rightarrow 4$

d

c b <u>a</u> d $\rightarrow 2$

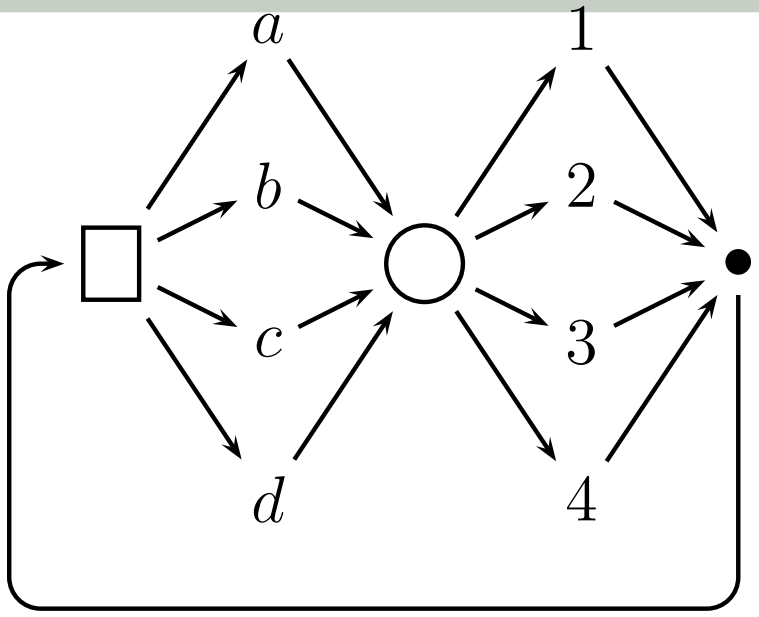
d

c b a <u>d</u> $\rightarrow 1$

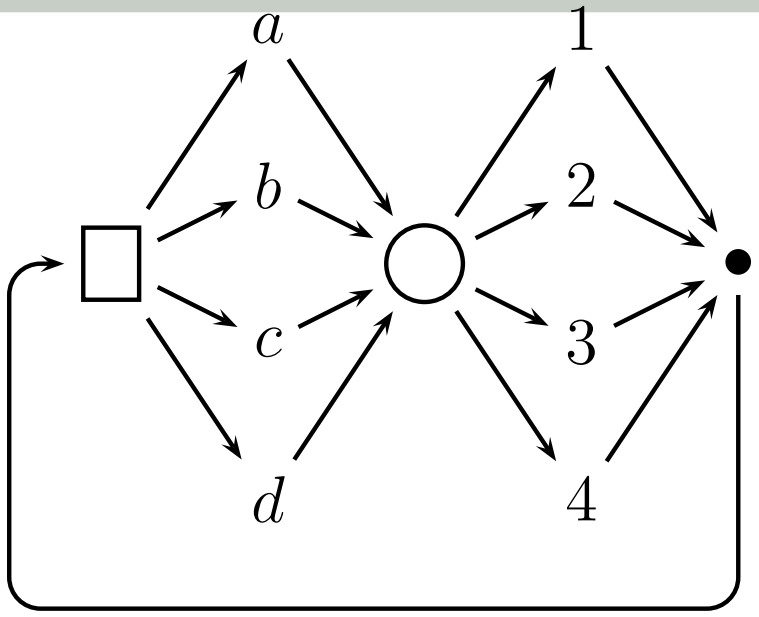
a

c b <u>d</u> a $\rightarrow 2$

d



LAR example



a b c d $\rightarrow 1$

b

a c d b $\rightarrow 3$

d

a c b d $\rightarrow 2$

a

c b d a $\rightarrow 4$

d

c b a d $\rightarrow 2$

d

c b a d $\rightarrow 1$

a

c b d a $\rightarrow 2$

d

c b a d $\rightarrow 2$

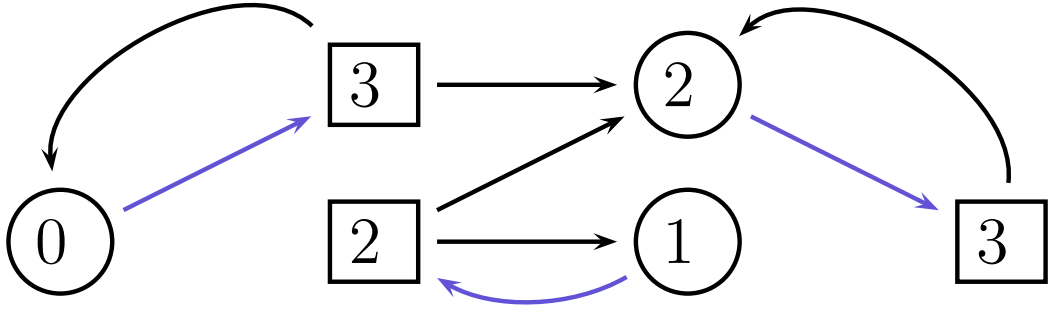
If from now on only a and d will appear then we will see 2 infinitely often and never 3 or 4.

Parity games admit memoryless strategies

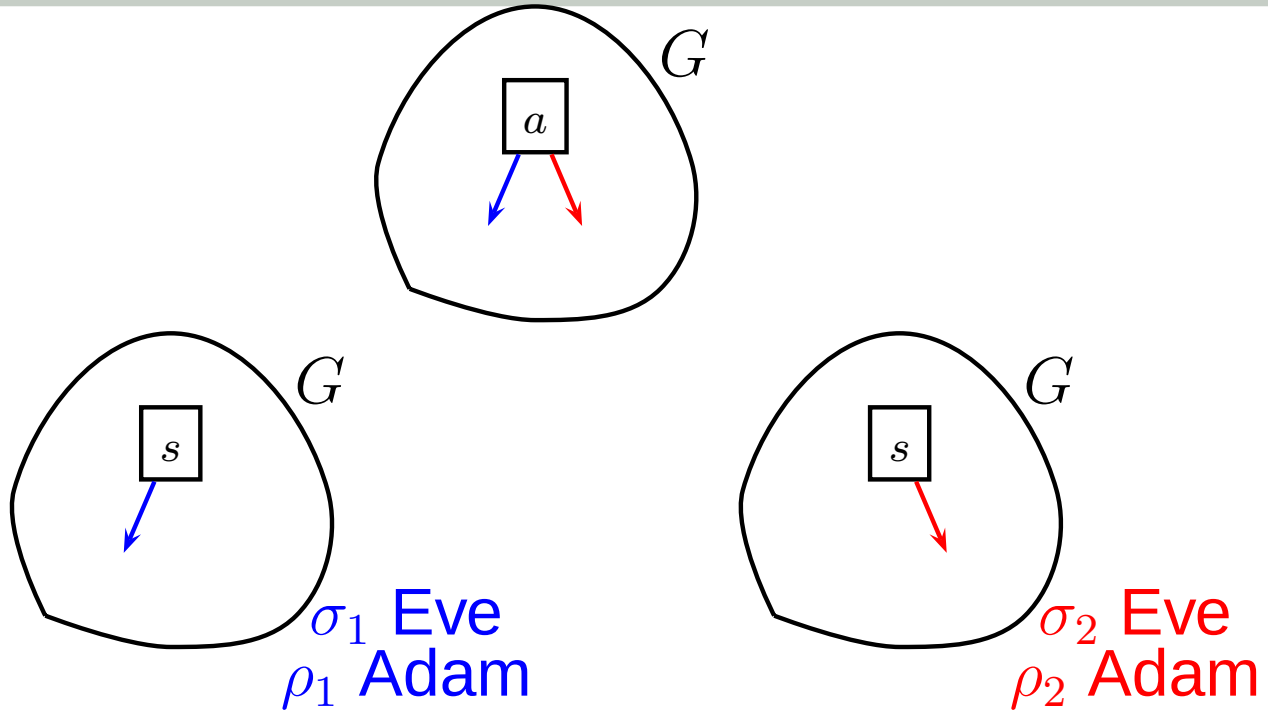
Thm [Mostowski, Emerson & Jutla]: In a finite parity game Eve has a memoryless strategy winning from each of her winning vertices.

- Proof by induction on the number of edges from Eve's positions

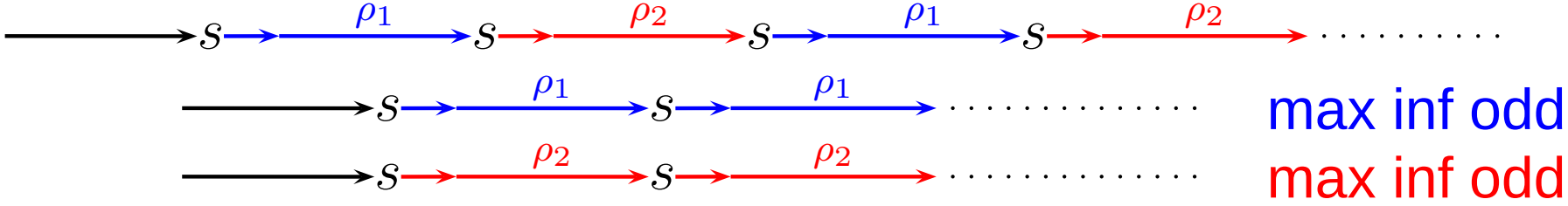
- If each position has one outgoing edge then this is the strategy for Eve.



Memoryless strategies: induction step



- If σ_1 or σ_2 is winning in G then we can use it.
- Suppose not. Then Adam can win from s in G.



Complexity classes

- A problem $L \subseteq \{0,1\}^*$ is in PTIME iff there is a machine M and a polynomial $p(n)$ such that on every input $w \in \{0,1\}^*$ machine M does at most $p(|w|)$ steps and answers yes/no correctly.

- A problem $L \subseteq \{0,1\}^*$ is in NP iff there is a polynomial $p'(n)$ such that for every $w \in L$ there is $w' \in \{0,1\}^*$ of size $< p'(|w|)$ such that $\{w\$w' : w \in L\}$ is in PTIME.

- Example: Satisfiability of a propositional formula:

$$(x_1 \vee x_2 \vee \neg x_3) \wedge (\neg x_1 \vee \neg x_2 \vee x_3)$$

Guess a valuation and check.

- Example: Parity games

Guess a memoryless strategy and check.

Digression: Primes

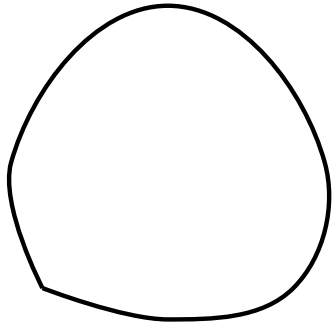
- COMPOSITE: Is a given number a composite number.

In NP: guess a decomposition.

- PRIMES: Is a given number a prime.

In NP: Lucas-Lehmer test. Number n is a prime iff $\exists a$ s.t.:
$a^n \equiv 1 \pmod{n}$ and $a^x \not\equiv 1 \pmod{n}$ for all $x = 1, \ldots, n-2$.

- In 2002 PRIMES were shown to be in PTIME
  [M. Agarwal, N. Saxena, N. Kayal]



$$\stackrel{?}{\models} \quad \alpha$$

$$\Downarrow$$

$$G = \langle V_E, V_A, R, \lambda : V \to \{0, \ldots, d\} \rangle$$

- Current algorithms work in time $|G|^{\mathcal{O}(d)}$ where d is the size of the range of λ.

- The size of d is related with the nesting depth of fixpoints in α.

Discounted pay-off games

- $G = \langle V_E, V_A, R, w : (V_E \cup V_A) \to \mathbb{R} \rangle$
- Outcome of $v_0, v_1, \ldots$ is $(1-\delta)\sum_{i=0}^{\infty} \delta^i w(v_i)$; here $0 < \delta < 1$ is a discount factor.
- Value of the game in a vertex v is a number $\mathcal{V}_v$ such that: Eve has a strategy from v to have an outcome $\geq \mathcal{V}_v$, and Adam has a strategy from v to have an outcome $\leq \mathcal{V}_v$.

Thm [Zwick and Paterson]: For every finite discounted pay-off game the value exists in every vertex and is given as a unique solution of the set of equations:

$$x_v = (1-\delta)w(v) + \begin{cases} \max_{(v,u)\in R} \delta x_u & \text{if } v \in V_E \\ \min_{(v,u)\in R} \delta x_u & \text{if } v \in V_A \end{cases}$$

- Define $\mathcal{F} : \mathbb{R}^n \to \mathbb{R}^n$ by:

$$\mathcal{F}_i(\vec{x}) = (1-\delta)w(i) + \begin{cases} \max_{(i,j)\in R} \delta x_j & \text{if } i \in V_E \\ \min_{(i,j)\in R} \delta x_j & \text{if } i \in V_A \end{cases}$$

- Consider the max norm $||\vec{x}|| = \max |x_i|$. We have:

$$\forall \vec{x}, \vec{y}.\ ||\mathcal{F}(\vec{x}) - \mathcal{F}(\vec{y})|| \leq \delta ||\vec{x} - \vec{y}||$$

- As $0 < \delta < 1$, mapping $\mathcal{F}$ is contracting with respect to the norm. So there is the unique fixed point $\vec{z} = \mathcal{F}(\vec{z})$.

- It is easy to see that Eve has a strategy to be not below $\vec{z}$ and Adam has a strategy to be not above $\vec{z}$.

Mean pay-off games

- $G = \langle V_E, V_A, R, w : (V_E \cup V_A) \to \mathbb{N} \rangle$

- Outcome for Eve of a play $v_0, v_1, \ldots$ is $\liminf_{n\to\inf} \frac{1}{n} \sum_{i=1}^{n} w(v_i)$.

For Adam it is $\limsup_{n\to\inf} \frac{1}{n} \sum_{i=1}^{n} w(v_i)$.

Thm [Ehrenfeucht & Mycielski]: Every vertex has a value v such that Eve has a strategy to be not below v and Adam a strategy to be not above v. Moreover the two players have memoryless strategies to achieve this.

Thm [Zwick & Paterson]: When $\delta \to 1$ then $\mathcal{V}_\delta^{ZP}(v) \to \mathcal{V}^{EM}(v)$.

Recall pay-off in ZP: $(1-\delta) \sum_{i=0}^{\infty} \delta^i w(v_i)$

Relation to parity games

- Loop games. $G = \langle V_E, V_A, R, w : (V_E \cup V_A) \to \mathbb{N} \rangle$. Players play until a cycle is closed. The outcome is the mean of the weights on the cycle.

Thm [Ehrenfeucht & Mycielski]: For all vertices

$$\mathcal{V}^{loop}(v) = \mathcal{V}^{EM}(v).$$

- Reduction of parity games to loop games:

$$G = \langle V_E, V_A, R, \Omega : (V_E \cup V_A) \to \{0, \ldots, d\} \rangle.$$

- Define $w(v) = (-n)^{\Omega(v)}$, where n is the number of vertices.

Obs: Eve has a winning strategy in a parity game with λ iff she has a strategy to obtain a positive value in the loop game with w.

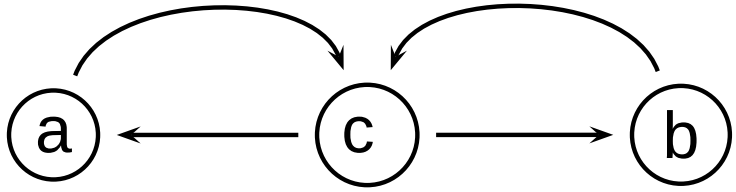
Plan

- Why logical formalisms.
- Two formalisms for model-checking.
- Advantages of formal systems.
- Model-checking as a game.
- Two player infinite games.
- Solving games.
- Special strategies in games.

Strategies: remainder

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

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

Conditions admitting positional strategies

- Memoryless strategies are interesting as:
- they are much easier to handle technically,
- the algorithms for finding them are simpler (of lower complexity),
- strategies are simple to describe (and use).

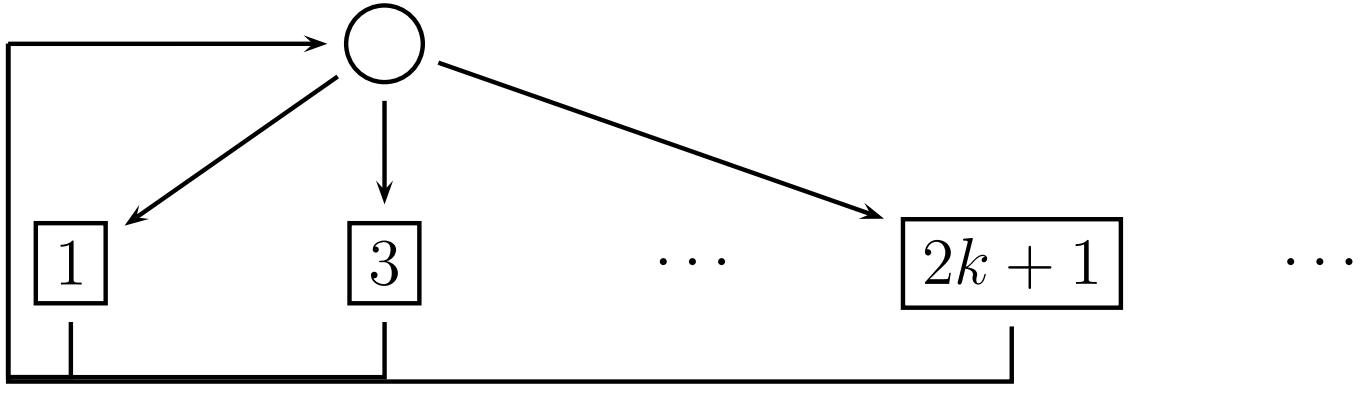
- A game is positionally determined iff both players have memoryless winning strategies from their winning positions.

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

Thm [McNaughton]: Parity conditions are the only Muller conditions admitting positional determinacy.

Infinite number of colours?

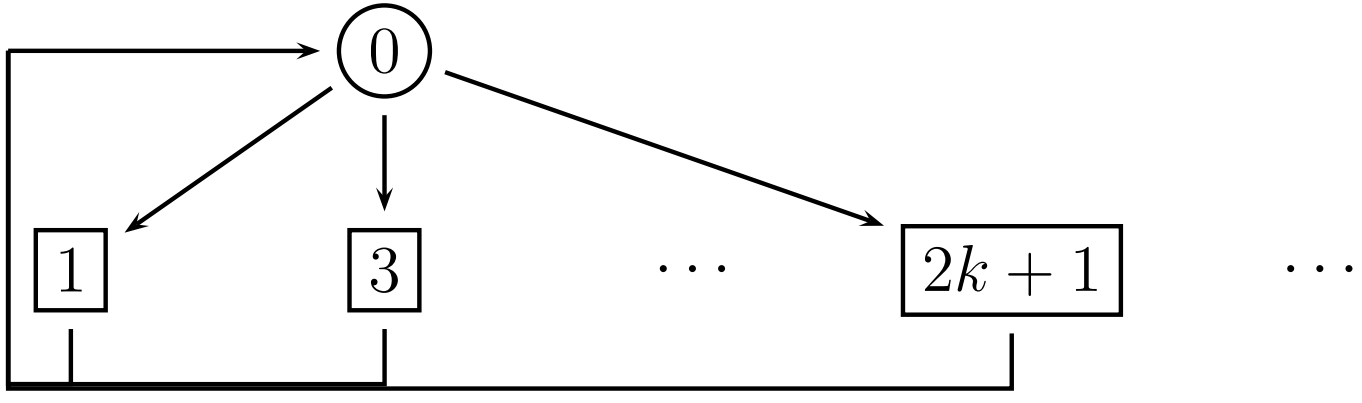
- Colouring function is now $\lambda : V \to \omega$
- Min-parity condition: **min**$(\mathrm{Inf}(p))$ is even or does not exist
- Max-parity condition: **max**$(\mathrm{Inf}(p))$ is even or does not exist



- What if all the vertices need to be colored?

Infinite number of colours?

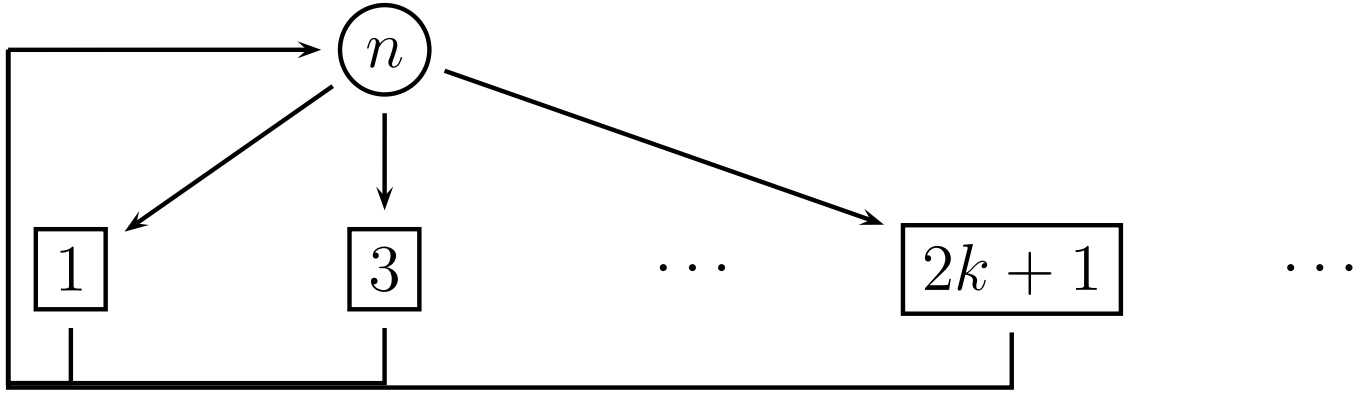
- Colouring function is now $\lambda : V \to \omega$
- Min-parity condition: **min**$(\mathrm{Inf}(p))$ is even or does not exist
- Max-parity condition: **max**$(\mathrm{Inf}(p))$ is even or does not exist



- What if all the vertices need to be colored?

Infinite number of colours?

- Colouring function is now $\lambda : V \to \omega$
- Min-parity condition: **min**$(\mathrm{Inf}(p))$ is even or does not exist
- Max-parity condition: **max**$(\mathrm{Inf}(p))$ is even or does not exist



- What if all the vertices need to be colored?

Characterization of winning conditions

- Muller conditions with infinite number of colours.

$$G = \{V_E, V_A, E, \lambda : V \to \omega\}$$

- 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.

Conclusions

- Formal languages are necessary for verification (if only due to the number of cases to check).

- The important issues are those of expessivity and complexity of a language.

  MSOL/bisimulation $\equiv$ μ-calculus.

- Verification process can usually be reduced to the problem of solving games.

  $M \stackrel{?}{\models} \alpha \quad \mapsto \quad$ finding a winner in $G(M, \alpha)$

- Games with memoryless strategies are easier to work with.

  Classes of games admitting positional determinacy

Modelchecking infnite graphs

From pushdown to regular graphs

Graphs of pushdown machines

- Pushdown machine (deterministic):
$\langle Q, \Sigma, \Gamma, q_0 \in Q, \delta : Q \times \Sigma \times \Gamma \to Q \times \{pop, push(z) : z \in \Gamma\}, F \subseteq Q \rangle$.

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

- Configuration graph
  - nodes: configurations
  - transitions:

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

  $(q, zw) \to (q', z'zw)$ 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: $\Delta \subseteq Q \times \Gamma \times Q \times (\{\varepsilon\} \cup \Gamma^2)$

$$qz \rightarrowtail q' \qquad qz \rightarrowtail q'z'z$$

Pushdown graph: $G(P)$

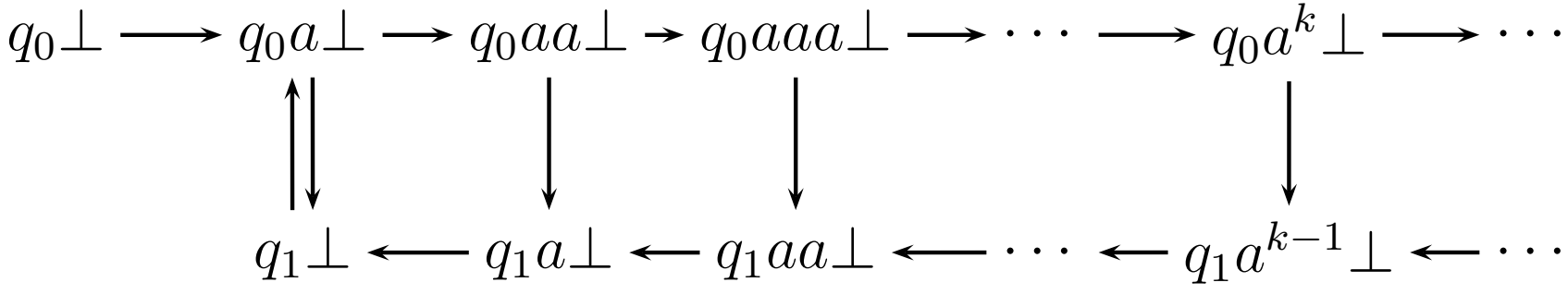
Vertices: $Q \times \Gamma^*$

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 $aqb \rightarrowtail q'a'b$ or $aqb \rightarrowtail ab'q'$ without restrictions on the place of application.

Pushdown graph: an example



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

$$q_0\bot \rightarrowtail q_0a\bot \qquad q_1\bot \rightarrowtail q_0a\bot$$
$$q_0a \rightarrowtail q_0aa \qquad q_0a \rightarrowtail q_1$$
$$q_1a \rightarrowtail q_1$$

Prefix-recognizable graphs

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

Rewrite rules: $\Delta \subseteq \mathcal{P}(Q^*) \times \mathcal{P}(Q^*)$

$L \rightarrowtail L'$ for L and L' regular languages.

Prefix-recognizable graph: $G(P)$

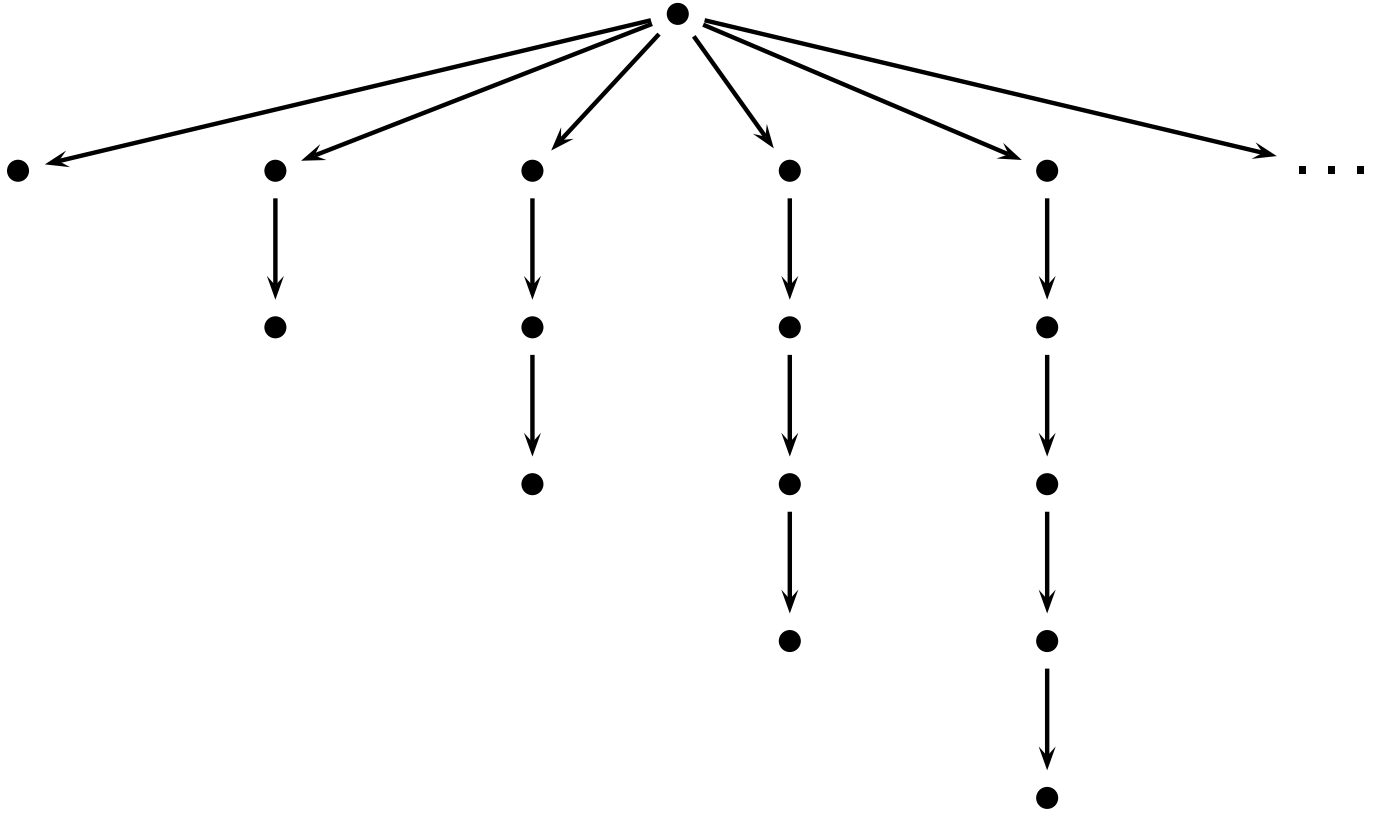
Vertices: Γ^*

Edges: $wu \rightarrow w'u$ if $w \in L$ and $w' \in L'$ for some $L \rightarrowtail L'$.

Rem: Prefix-recognizable graph of finite degree is a pushdown graph.

Thm [Carayol & Wöhrle]: Prefix-recognizable graphs are ε-closures of pushdown graphs.

Example of a prefix recognizable graph



Synchronized rational and rational graphs

- A relation $R \subseteq \Gamma^* \times \Gamma^*$ is rational if it is recognizable by a finite automaton with two heads moving asynchronously from left to right.

- A relation $R \subseteq \Gamma^* \times \Gamma^*$ is synchronous rational if the heads of the automaton always move together.

- A graph is rational if it is (Γ^*, R) where R is a rational relation.

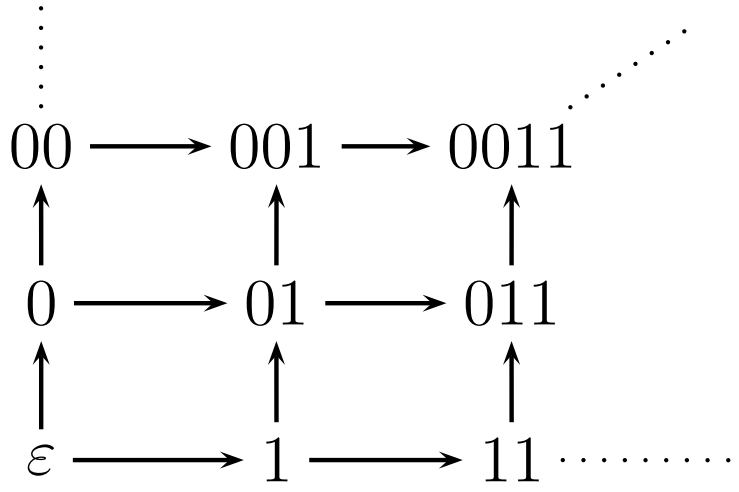
Rem: TM graphs are synchronous rational.

Rem: Synchronous rational are also called automatic.

Examples of automatic and rational graphs

- Synchrnous rational graph: grid

$R_u = \{(0^n 1^n, 0^{n+1} 1^n) : n \in \mathbb{N}\}$ $R_r = \{(0^n 1^n, 0^n 1^{n+1}) : n \in \mathbb{N}\}$



- Rational graph: Given $(u_1, \ldots, u_n)$, $(v_1, \ldots, v_n)$ of a Post correspondence problem define:

$$R = \{(u_{i_1} \ldots u_{i_k}, v_{i_1} \ldots v_{i_k}) : i_1, \ldots, i_k \in \{1, \ldots, n\}\}$$

- In general this is not a synchronous rational graph.

MSOL over trees

- The Σ-tree is Σ^* with the root ε and wb the b son of w.
- MSOL over Σ-trees:
  $$succ_b(x,y) \mid Z(x) \mid \neg\varphi \mid \varphi \wedge \psi \mid \exists x.\varphi \mid \exists Z.\varphi$$
- Semantics in the Σ-tree.

$+$ $G, V \models succ_b(x,y)$ iff $V(x)b = V(y)$

$+$ $G, V \models \exists Z.\alpha$ iff $G, V[A/Z] \models \alpha$ for some $A \subseteq E$.

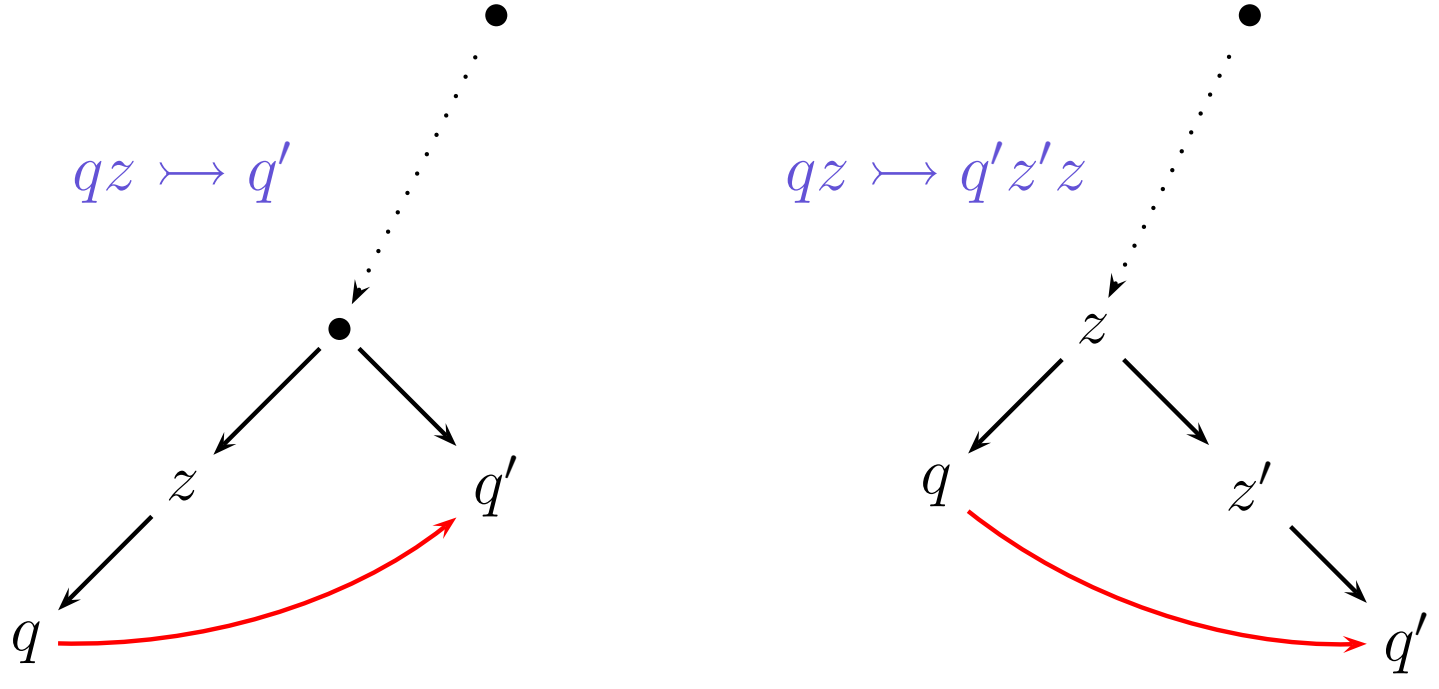
Thm[Rabin]: The MSO theory of the Σ-tree is decidable.

Thm[Caucal]: A pushdown tree can be defined inside the Σ-tree using MSOL formulas.

Cor[Muller, Schupp]: The MSO theory of a pushdown tree is decidable

Pushdown system inside a tree

- Pushdown rules: $qz \rightarrowtail q'$ $\qquad$ $qz \rightarrowtail q'z'z$
- Configuration is represented by a word $qz_k z_{k-1} \ldots z_1$.
- Hence we can identify configurations with some nodes of the tree $(Q \cup \Gamma)^*$.



Model checking pushdown systems

- Let $\rho : Q \to \mathcal{P}(\mathrm{Prop})$ be a valuation. This extends to

$$\rho : Q \times \Gamma^* \to \mathcal{P}(\mathrm{Prop}) \quad \text{by} \quad \rho(qw) = \rho(q).$$

So we have a model $M(P, \rho)$.

Model checking problem: Given a pushdown system $\mathcal{P}$ with a valuation ρ and a formula α check if $M(P, \rho), q_0 \perp \models \alpha$.

- Example: Alternating reachability

  Is there a choice of successors in E nodes such that every path from v passes through F.

EF, CTL, μ-calculus

- EF logic

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

!No $\exists G\alpha$!

- CTL

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

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

- μ-calculus

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

Results

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]

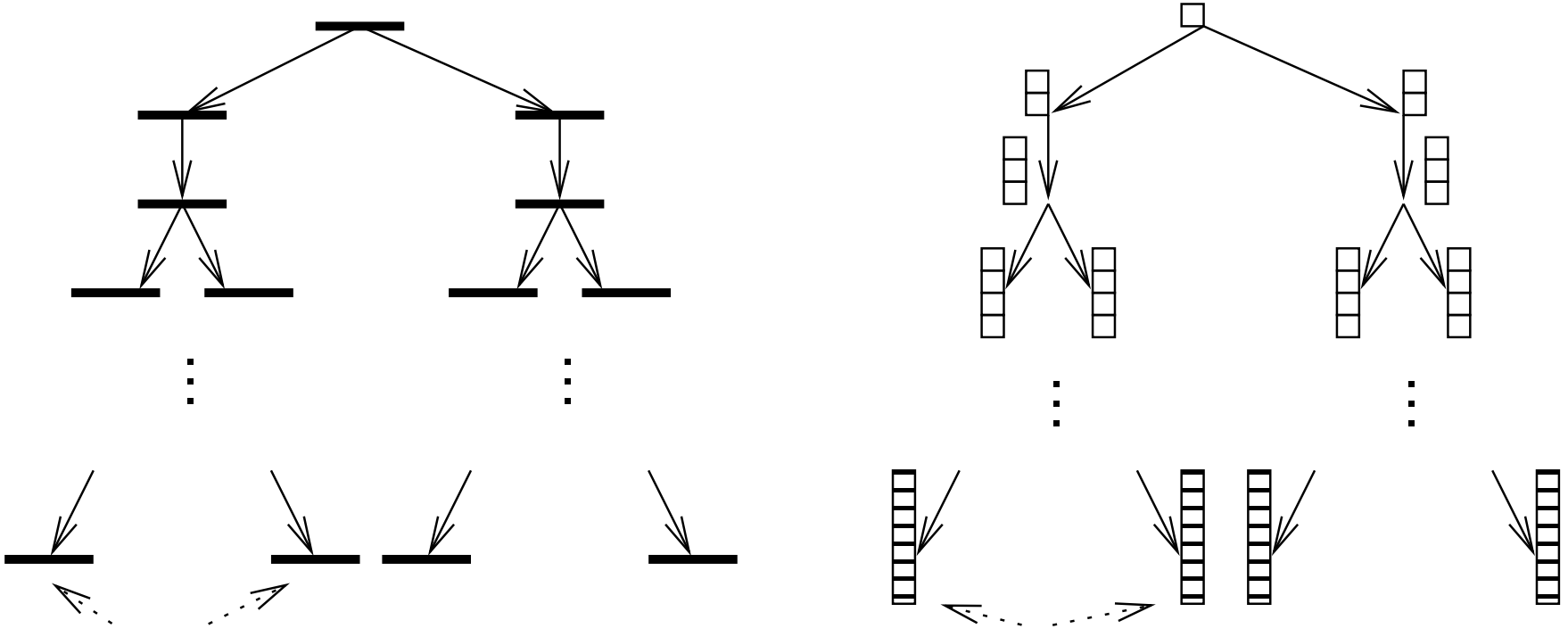
Thm: The same problem for CTL formulas is EXPTIME-complete.

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

Alt reachability: EXPTIME-hard

- Take ASPACE(n) machine M and input w. Construct P_w:

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



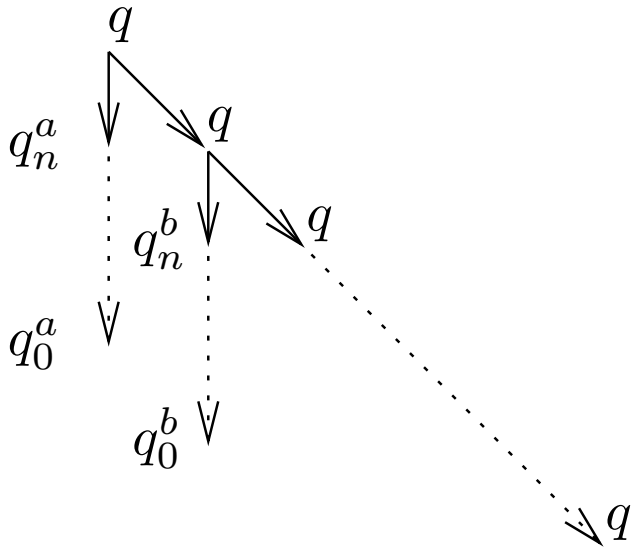
- How to check that a sequence of accumulated configurations is correct?

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.

$$qa \rightarrow q,\ q_n^a \qquad q\perp \rightarrow q_F$$

$$q_n^a b \rightarrow q_{n-1}^a \qquad q_0^a a \rightarrow q_F$$



Summary of pushdown model checking

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

Decidability: synchronized regular graphs

- TM graphs are synchronized regular.

- As reachability is expressible in MSOL, synchronized regular graphs may have undecidable MSO theory.

Thm: The FOL theory of a synchronized regular graph is decidable

- Every FOL definable relation in a synchronized regular graphs is synchronized regular. (Induction on the definition of the relation.)

Decidability: Regular graphs

Thm [Thomas]: There is a regular graph that has undecidable FO-theory.

- Consider a Post correspondence problem $(u_1, \ldots, u_n)$, $(v_1, \ldots, v_n)$ and the associated graph:
$$R = \{(u_{i_1} \ldots u_{i_k}, v_{i_1} \ldots v_{i_k}) : i_1, \ldots, i_k \in \{1, \ldots, n\}\}$$
- [Morvan] The problem has a solution iff there is a vertex with a self-loop: $u_{i_1} \ldots u_{i_k} = v_{i_1} \ldots v_{i_k}$

- We want a fixed graph with undecidable FOL theory.
- $G(U)$ Post graph associated with an universal Turing machine.
- For every M and w: $w \in L(M)$ iff the Post correspondence problem has a solution starting with $w\#c(M)\# \ldots$
- Equivalently: $w \in L(M)$ iff in $G(U)$ there is a vertex with a self loop and with prefix $w\#c(M)\#$.

Decidability:summary

- Push-down and prefix-recognizable graphs have decidable MSO theory.
- Push-down graphs have bounded out degree.
- Synchronous rational graphs have decidable FO theory. But may have undecidable MSO theory.
- Rational graphs may have undecidable FO theory.

Cor: The strict inclusion of the classes of graphs.

Conclusions

- Model checking pushdown graphs is almost as easy as finite graphs.
- For synchronous rational graphs only FOL theory is decidable (reachability is not).
- Rational graphs may have undecidable even FOL theory.