Logic and Games

A Tutorial

Erich Grädel

Outline

Part I: Model Checking Games

- Model checking games for modal logic and first-order logic
- The strategy problem for finite games
- Fragments of first-order logics with efficient model checking
- Fixed point logics: LFP and modal μ-calculus
- Parity games
- Model checking games for fixed point logics

Model checking via games

The model checking problem for a logic L

Given: structure $\mathfrak{A}$

formula $\psi \in L$

Question: $\mathfrak{A} \models \psi$?

Model checking via games

The model checking problem for a logic L

Given: structure $\mathfrak{A}$

formula $\psi \in L$

Question: $\mathfrak{A} \models \psi$?

Reduce model checking problem $\mathfrak{A} \models \psi$ to strategy problem for model checking game $G(\mathfrak{A}, \psi)$, played by

- Falsifier (also called Player 1, or Alter), and
- Verifier (also called Player 0, or Ego), such that

$$\mathfrak{A} \models \psi \iff \text{Verifier has winning strategy for } G(\mathfrak{A}, \psi)$$

Model checking via games

The model checking problem for a logic L

Given: structure $\mathfrak{A}$

formula $\psi \in L$

Question: $\mathfrak{A} \models \psi$?

Reduce model checking problem $\mathfrak{A} \models \psi$ to strategy problem for model checking game $G(\mathfrak{A}, \psi)$, played by

- Falsifier (also called Player 1, or Alter), and
- Verifier (also called Player 0, or Ego), such that

$$\mathfrak{A} \models \psi \iff \text{Verifier has winning strategy for } G(\mathfrak{A}, \psi)$$

$\Longrightarrow$ Model checking via construction of winning strategies

ML: propositional modal logic

Syntax: $\psi ::= P_i \mid \neg P_i \mid \psi \wedge \psi \mid \psi \vee \psi \mid \langle a \rangle \psi \mid [a]\psi$

Example: $P_1 \vee \langle a \rangle (P_2 \wedge [b]P_1)$

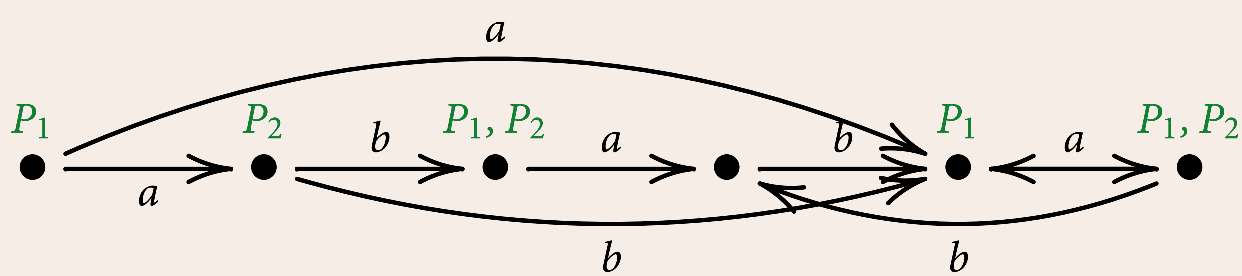
Syntax: $\psi ::= P_i \mid \neg P_i \mid \psi \wedge \psi \mid \psi \vee \psi \mid \langle a \rangle \psi \mid [a]\psi$

Example: $P_1 \vee \langle a \rangle (P_2 \wedge [b] P_1)$

Semantics: transition systems = Kripke structures = labeled graphs

$$\mathcal{K} = (V , (E_a)_{a \in A} , (P_i)_{i \in I})$$

V	$(E_a)_{a \in A}$	$(P_i)_{i \in I}$
states elements	actions binary relations	atomic propositions unary relations



$$[\![\psi]\!]^{\mathcal{K}} = \{v : \mathcal{K}, v \models \psi\} = \{v : \psi \text{ holds at state } v \text{ in } \mathcal{K}\}$$

$$\mathcal{K}, v \models \begin{matrix} \langle a \rangle \psi \\ [a]\psi \end{matrix} \quad :\Longleftrightarrow \quad \mathcal{K}, w \models \psi \text{ for } \begin{matrix} \text{some} \\ \text{all} \end{matrix} \; w \text{ with } (v, w) \in E_a$$

Model checking game for ML

Game $\mathcal{G}(\mathcal{K}, \psi)$ (for transition system $\mathcal{K}$ and $\psi \in \mathrm{ML}$)

Model checking game for ML

Game $\mathcal{G}(\mathcal{K}, \psi)$ (for transition system $\mathcal{K}$ and $\psi \in \mathrm{ML}$)

Positions: (φ, v) φ subformula of ψ, $v \in V$

From position (φ, v), Verifier wants to show that $\mathcal{K}, v \models \varphi$, while Falsifier wants to prove that $\mathcal{K}, v \not\models \varphi$.

Model checking game for ML

Game $\mathcal{G}(\mathcal{K}, \psi)$ (for transition system $\mathcal{K}$ and $\psi \in \mathrm{ML}$)

Positions: (φ, v) φ subformula of ψ, $v \in V$

From position (φ, v), Verifier wants to show that $\mathcal{K}, v \models \varphi$, while Falsifier wants to prove that $\mathcal{K}, v \not\models \varphi$.

Verifier moves:

$$(\varphi \vee \vartheta, v) \longrightarrow (\varphi, v), \quad (\varphi \vee \vartheta, v) \longrightarrow (\vartheta, v)$$

$$(\langle a \rangle \varphi, v) \longrightarrow (\varphi, w), \quad w \in vE_a$$

Model checking game for ML

Game $\mathcal{G}(\mathcal{K}, \psi)$ (for transition system $\mathcal{K}$ and $\psi \in \mathrm{ML}$)

Positions: (φ, v) φ subformula of ψ, $v \in V$

From position (φ, v), Verifier wants to show that $\mathcal{K}, v \models \varphi$, while Falsifier wants to prove that $\mathcal{K}, v \not\models \varphi$.

Verifier moves:

$$(\varphi \vee \vartheta, v) \longrightarrow (\varphi, v), \qquad (\varphi \vee \vartheta, v) \longrightarrow (\vartheta, v)$$

$$(\langle a \rangle \varphi, v) \longrightarrow (\varphi, w), \quad w \in vE_a$$

Falsifier moves:

$$(\varphi \wedge \vartheta, v) \longrightarrow (\varphi, v), \qquad (\varphi \wedge \vartheta, v) \longrightarrow (\vartheta, v)$$

$$([a]\varphi, v) \longrightarrow (\varphi, w), \quad w \in vE_a$$

Model checking game for ML

Game $\mathcal{G}(\mathcal{K}, \psi)$ (for transition system $\mathcal{K}$ and $\psi \in \mathrm{ML}$)

Positions: (φ, v) φ subformula of ψ, $v \in V$

From position (φ, v), Verifier wants to show that $\mathcal{K}, v \models \varphi$, while Falsifier wants to prove that $\mathcal{K}, v \not\models \varphi$.

Verifier moves:

$$(\varphi \lor \vartheta, v) \longrightarrow (\varphi, v), \quad (\varphi \lor \vartheta, v) \longrightarrow (\vartheta, v)$$

$$(\langle a \rangle \varphi, v) \longrightarrow (\varphi, w), \quad w \in vE_a$$

Falsifier moves:

$$(\varphi \land \vartheta, v) \longrightarrow (\varphi, v), \quad (\varphi \land \vartheta, v) \longrightarrow (\vartheta, v)$$

$$([a]\varphi, v) \longrightarrow (\varphi, w), \quad w \in vE_a$$

Terminal positions: (P_i, v), $(\neg P_i, v)$

If $\mathcal{K}, v \models P_i$ then Verifier has won at (P_i, v), otherwise Falsifier has won.

Model checking game for ML

Game $\mathcal{G}(\mathcal{K}, \psi)$ (for transition system $\mathcal{K}$ and $\psi \in \mathrm{ML}$)

Positions: (φ, v) φ subformula of ψ, $v \in V$

From position (φ, v), Verifier wants to show that $\mathcal{K}, v \models \varphi$, while Falsifier wants to prove that $\mathcal{K}, v \not\models \varphi$.

Verifier moves:

$$(\varphi \vee \vartheta, v) \begin{array}{l} \nearrow (\varphi, v) \\ \searrow (\vartheta, v) \end{array} \qquad (\langle a \rangle \varphi, v) \longrightarrow (\varphi, w), \quad w \in vE_a$$

Falsifier moves:

$$(\varphi \wedge \vartheta, v) \begin{array}{l} \nearrow (\varphi, v) \\ \searrow (\vartheta, v) \end{array} \qquad ([a] \varphi, v) \longrightarrow (\varphi, w), \quad w \in vE_a$$

Terminal positions: (P_i, v), $(\neg P_i, v)$

If $\mathcal{K}, v \models P_i$ then Verifier has won at (P_i, v), otherwise Falsifier has won.

Lemma. $\mathcal{K}, v \models \varphi \iff$ Verifier has winning strategy from (φ, v).

Games and logics

Do games provide efficient solutions for model checking problems?

Games and logics

Do games provide efficient solutions for model checking problems?

This depends on the logic, and on what we mean by efficient!

Games and logics

Do games provide efficient solutions for model checking problems?

This depends on the logic, and on what we mean by efficient!

- How complicated are the resulting model checking games?
  - are all plays necessarily finite?
  - if not, what are the winning conditions for infinite plays?
  - structural complexity of the game graphs?
  - do the players always have perfect information?
- How big are the resulting game graphs?

  how does the size of the game depend on different parameters of the input structure and the formula?

Logics and games

First-order logic (FO) or modal logic (ML): Model checking games have

- only finite plays
- positional winning condition

winning regions computable in linear time wrt. size of game graph

Logics and games

First-order logic (FO) or modal logic (ML): Model checking games have

- only finite plays
- positional winning condition

winning regions computable in linear time wrt. size of game graph

Fixed-point logics (LFP or L_μ): Model checking games are parity games

- admit infinite plays
- parity winning condition

Open problem: Are winning regions and winning strategies of parity games computable in polynomial time?

Finite games: basic definitions

Two-player games with perfect information and positional winning condition, given by game graph (also called arena)

$$\mathcal{G} = (V, E), \qquad V = V_0 \cup V_1$$

- Player 0 (Ego) moves from positions $v \in V_0$,
  Player 1 (Alter) moves from $v \in V_1$,
- moves are along edges
  a play is a finite or infinite sequence $\pi = v_0 v_1 v_2 \cdots$ with $(v_i, v_{i+1}) \in E$
- winning condition: move or lose!
  Player σ wins at position v if $v \in V_{1-\sigma}$ and $vE = \varnothing$
  Note: this is a purely positional winning condition applying to finite plays only (infinite plays are draws)

Winning strategies and winning regions

Strategy for Player σ: $f : \{v \in V_\sigma : vE \neq \varnothing\} \to V$ with $(v, f(v)) \in E$.

f is **winning from position v** if Player σ wins all plays that start at v and are consistent with f.

Winning strategies and winning regions

Strategy for Player σ: $\quad f : \{v \in V_\sigma : vE \neq \emptyset\} \to V \quad$ with $(v, f(v)) \in E$.

f is **winning from position v** if Player σ wins all plays that start at v and are consistent with f.

Winning regions W_0, W_1:

$$W_\sigma = \{v \in V : \text{Player } \sigma \text{ has winning strategy from position } v\}$$

Winning strategies and winning regions

Strategy for Player σ: $f : \{v \in V_\sigma : vE \neq \varnothing\} \to V$ with $(v, f(v)) \in E$.

f is **winning from position v** if Player σ wins all plays that start at v and are consistent with f.

Winning regions W_0, W_1:

$$W_\sigma = \{v \in V : \text{Player } \sigma \text{ has winning strategy from position } v\}$$

Algorithmic problems: Given a game $\mathcal{G}$

- compute winning regions W_0, W_1
- compute winning strategies

Associated decision problem:

$$\textsc{Game} := \{(\mathcal{G}, v) : \text{Player 0 has winning strategy for } \mathcal{G} \text{ from position } v\}$$

Algorithms for finite games

Theorem

GAME is PTIME-complete and solvable in time $O(|V| + |E|)$.

Algorithms for finite games

Theorem

GAME is PTIME-complete and solvable in time $O(|V| + |E|)$.

remains true for strictly alternating games on graphs $\mathcal{G} = (V, E)$.

Algorithms for finite games

Theorem

GAME is PTIME-complete and solvable in time $O(|V| + |E|)$.

remains true for strictly alternating games on graphs $\mathcal{G} = (V, E)$.

A simple polynomial-time algorithm

Compute winning regions inductively: $W_\sigma = \bigcup_{n\in\mathbb{N}} W_\sigma^n$ where

Algorithms for finite games

Theorem

GAME is PTIME-complete and solvable in time $O(|V| + |E|)$.

remains true for strictly alternating games on graphs $\mathcal{G} = (V, E)$.

A simple polynomial-time algorithm

Compute winning regions inductively: $W_\sigma = \bigcup_{n \in \mathbb{N}} W_\sigma^n$ where

- $W_\sigma^0 = \{v \in V_{1-\sigma} : vE = \varnothing\}$
  (winning terminal positions for Player σ)

Algorithms for finite games

Theorem

GAME is PTIME-complete and solvable in time $O(|V| + |E|)$.

remains true for strictly alternating games on graphs $\mathcal{G} = (V, E)$.

A simple polynomial-time algorithm

Compute winning regions inductively: $W_\sigma = \bigcup_{n \in \mathbb{N}} W_\sigma^n$ where

- $W_\sigma^0 = \{v \in V_{1-\sigma} : vE = \varnothing\}$
  (winning terminal positions for Player σ)
- $W_\sigma^{n+1} = \{v \in V_\sigma : vE \cap W_\sigma^n \neq \varnothing\} \cup \{v \in V_{1-\sigma} : vE \subseteq W_\sigma^n\}$
  (positions with winning strategy in $\leq n + 1$ moves for Player σ)

Algorithms for finite games

Theorem

Game is Ptime-complete and solvable in time $O(|V| + |E|)$.

remains true for strictly alternating games on graphs $\mathcal{G} = (V, E)$.

A simple polynomial-time algorithm

Compute winning regions inductively: $W_\sigma = \bigcup_{n \in \mathbb{N}} W_\sigma^n$ where

- $W_\sigma^0 = \{v \in V_{1-\sigma} : vE = \emptyset\}$
  (winning terminal positions for Player σ)
- $W_\sigma^{n+1} = \{v \in V_\sigma : vE \cap W_\sigma^n \neq \emptyset\} \cup \{v \in V_{1-\sigma} : vE \subseteq W_\sigma^n\}$
  (positions with winning strategy in $\leq n + 1$ moves for Player σ)

until $W_\sigma^{n+1} = W_\sigma^n$ (this happens for $n \leq |V|$).

A linear time algorithm for Game

Input: A game $\mathcal{G} = (V, V_0, V_1, E)$

forall $v \in V$ **let** (∗ 1: initialisation ∗)

  $\text{win}[v] := \perp, \quad P[v] := \{u : (u, v) \in E\}, \quad n[v] := |vE|$

forall $\sigma \in \{0, 1\}, v \in V_\sigma$ (∗ 2: calculate win ∗)

  if $n[v] = 0$ **then** **Propagate**$(v, 1 - \sigma)$

return win **end**

procedure **Propagate**(v, σ)

  if $\text{win}[v] \neq \perp$ **then** **return**

  $\text{win}[v] := \sigma$ (∗ 3: mark v as winning for Player σ ∗)

  forall $u \in P[v]$ **do** (∗ 4: propagate change to predecessors ∗)

    $n[u] := n[u] - 1$

    if $u \in V_\sigma$ or $n[u] = 0$ **then** **Propagate**(u, σ)

  enddo

GAME and the satisfiability of propositional Horn formulae

Propositional Horn formulae: conjunctions of clauses of form

$$X \leftarrow X_1 \wedge \cdots \wedge X_n \qquad \text{and} \qquad 0 \leftarrow X_1 \wedge \cdots \wedge X_n$$

Theorem. SAT-HORN is PTIME-complete and solvable in linear time.

(actually, GAME and SAT-HORN are essentially the same problem)

Game and the satisfiability of propositional Horn formulae

Propositional Horn formulae: conjunctions of clauses of form

$$X \leftarrow X_1 \wedge \cdots \wedge X_n \qquad \text{and} \qquad 0 \leftarrow X_1 \wedge \cdots \wedge X_n$$

Theorem. Sat-Horn is Ptime-complete and solvable in linear time.

(actually, Game and Sat-Horn are essentially the same problem)

1) Game $\leq_{\text{log-lin}}$ Sat-Horn:

For $\mathcal{G} = (V_0 \cup V_1, E)$ construct Horn formula ψ with clauses

$$u \leftarrow v \qquad \text{for all } u \in V_0 \text{ and } (u, v) \in E$$

$$u \leftarrow v_1 \wedge \cdots \wedge v_m \qquad \text{for all } u \in V_1,\ uE = \{v_1, \ldots, v_m\}$$

The minimal model of ψ is precisely the winning region of Player 0.

$$(\mathcal{G}, v) \in \text{Game} \iff \psi_{\mathcal{G}} \wedge (0 \leftarrow v) \text{ is unsatisfiable}$$

2) Sat-Horn $\leq_{\text{log-lin}}$ Game:

Define game $\mathcal{G}_\psi$ for Horn formula $\psi(X_1, \dots, X_n) = \bigwedge_{i \in I} C_i$

Positions: $\{0\} \cup \{X_1, \dots, X_n\} \cup \{C_i : i \in I\}$

Moves of Player 0: $X \longrightarrow C$ for $X = \text{head}(C)$

Moves of Player 1: $C \longrightarrow X$ for $X \in \text{body}(C)$

Note: Player 0 wins iff play reaches clause C with $\text{body}(C) = \varnothing$

Player 0 has winning strategy from position $X \iff \psi \models X$

Hence,

$$\text{Player 0 wins from position 0} \iff \psi \text{ unsatisfiable.}$$

Alternating algorithms

nondeterministic algorithms, with states divided into
accepting, rejecting, existential, and universal states

Alternating algorithms

nondeterministic algorithms, with states divided into
accepting, rejecting, existential, and universal states

Acceptance condition: game with Players $\exists$ and $\forall$, played on computation graph $C(M, x)$ of M on input x

Positions: configurations of M

Moves: $C \longrightarrow C'$ for C' successor configuration of C

- Player $\exists$ moves at existential configurations
  wins at accepting configurations
- Player $\forall$ moves at universal configurations
  wins at rejecting configurations

Alternating algorithms

nondeterministic algorithms, with states divided into accepting, rejecting, existential, and universal states

Acceptance condition: game with Players $\exists$ and $\forall$, played on computation graph $C(M, x)$ of M on input x

Positions: configurations of M

Moves: $C \to C'$ for C' successor configuration of C

- Player $\exists$ moves at existential configurations
  wins at accepting configurations
- Player $\forall$ moves at universal configurations
  wins at rejecting configurations

M accepts x $\quad :\Longleftrightarrow \quad$ Player $\exists$ has winning strategy for game on $C(M, x)$

Alternating versus deterministic complexity classes

Alternating time $\equiv$ deterministic space

Alternating space $\equiv$ exponential deterministic time

$$
\begin{array}{ccccccccc}
\textsc{Logspace} & \subseteq & \textsc{Ptime} & \subseteq & \textsc{Pspace} & \subseteq & \textsc{Exptime} & \subseteq & \textsc{Expspace} \\
 & & \| & & \| & & \| & & \| \\
 & & \textsc{Alogspace} & \subseteq & \textsc{Aptime} & \subseteq & \textsc{Apspace} & \subseteq & \textsc{Aexptime}
\end{array}
$$

Alternating versus deterministic complexity classes

Alternating time $\equiv$ deterministic space

Alternating space $\equiv$ exponential deterministic time

$$
\begin{array}{ccccccccc}
\textsc{Logspace} & \subseteq & \textsc{Ptime} & \subseteq & \textsc{Pspace} & \subseteq & \textsc{Exptime} & \subseteq & \textsc{Expspace} \\
 & & \| & & \| & & \| & & \| \\
 & & \textsc{Alogspace} & \subseteq & \textsc{Aptime} & \subseteq & \textsc{Apspace} & \subseteq & \textsc{Aexptime}
\end{array}
$$

Alternating logspace algorithm for GAME: Play the game !

Evaluation game for FO

FO: $\psi ::= R_i\bar{x} \mid \neg R_i\bar{x} \mid x = y \mid x \neq y \mid \psi \wedge \psi \mid \psi \vee \psi \mid \exists x\psi \mid \forall x\psi$

Evaluation game for FO

FO: $\psi ::= R_i\overline{x} \mid \neg R_i\overline{x} \mid x = y \mid x \neq y \mid \psi \wedge \psi \mid \psi \vee \psi \mid \exists x\psi \mid \forall x\psi$

The game $\mathcal{G}(\mathfrak{A}, \psi)$ (for $\mathfrak{A} = (A, R_1, \ldots, R_m)$, $R_i \subseteq A^{r_i}$)

Evaluation game for FO

FO: $\psi ::= R_i\bar{x} \mid \neg R_i\bar{x} \mid x = y \mid x \neq y \mid \psi \wedge \psi \mid \psi \vee \psi \mid \exists x\psi \mid \forall x\psi$

The game $\mathcal{G}(\mathfrak{A}, \psi)$ (for $\mathfrak{A} = (A, R_1, \ldots, R_m)$, $R_i \subseteq A^{r_i}$)

Positions: $\varphi(\bar{a})$ $\varphi(\bar{x})$ subformula of ψ, $\bar{a} \in A^k$

Evaluation game for FO

FO: $\psi ::= R_i\bar{x} \mid \neg R_i\bar{x} \mid x = y \mid x \neq y \mid \psi \wedge \psi \mid \psi \vee \psi \mid \exists x\psi \mid \forall x\psi$

The game $\mathcal{G}(\mathfrak{A}, \psi)$ (for $\mathfrak{A} = (A, R_1, \dots, R_m)$, $R_i \subseteq A^{r_i}$)

Positions: $\varphi(\bar{a})$ $\quad\varphi(\bar{x})$ subformula of ψ, $\bar{a} \in A^k$

Verifier moves:

$$\varphi \vee \vartheta \to \varphi, \qquad \varphi \vee \vartheta \to \vartheta$$

$$\exists x\varphi(x, \bar{b}) \longrightarrow \varphi(a, \bar{b}) \quad (a \in A)$$

Falsifier moves:

$$\varphi \wedge \vartheta \to \varphi, \qquad \varphi \wedge \vartheta \to \vartheta$$

$$\forall x\varphi(x, \bar{b}) \longrightarrow \varphi(a, \bar{b}) \quad (a \in A)$$

Evaluation game for FO

FO: $\psi ::= R_i\bar{x} \mid \neg R_i\bar{x} \mid x = y \mid x \neq y \mid \psi \wedge \psi \mid \psi \vee \psi \mid \exists x\psi \mid \forall x\psi$

The game $\mathcal{G}(\mathfrak{A}, \psi)$ (for $\mathfrak{A} = (A, R_1, \dots, R_m)$, $R_i \subseteq A^{r_i}$)

Positions: $\varphi(\bar{a})$ $\varphi(\bar{x})$ subformula of ψ, $\bar{a} \in A^k$

Verifier moves:

$$\varphi \vee \vartheta \to \varphi, \quad \varphi \vee \vartheta \to \vartheta \qquad \exists x\varphi(x, \bar{b}) \longrightarrow \varphi(a, \bar{b}) \quad (a \in A)$$

Falsifier moves:

$$\varphi \wedge \vartheta \to \varphi, \quad \varphi \wedge \vartheta \to \vartheta \qquad \forall x\varphi(x, \bar{b}) \longrightarrow \varphi(a, \bar{b}) \quad (a \in A)$$

Winning condition: φ atomic / negated atomic

Verifier / Falsifier wins at $\varphi(\bar{a}) \iff \mathfrak{A} \models / \not\models \varphi(\bar{a})$

Complexity of FO model checking

To decide whether $\mathfrak{A} \models \psi$, construct the game $\mathcal{G}(\mathfrak{A}, \psi)$ and check whether Verifier has winning strategy from initial position ψ.

Efficient implementation: on-the-fly construction of game while solving it

Complexity of FO model checking

To decide whether $\mathfrak{A} \models \psi$, construct the game $\mathcal{G}(\mathfrak{A}, \psi)$ and check whether Verifier has winning strategy from initial position ψ.

Efficient implementation: on-the-fly construction of game while solving it

Size of game graph can be exponential: $|\mathcal{G}(\mathfrak{A}, \psi)| \leq |\psi| \cdot |A|^{\mathrm{width}(\psi)}$

width(ψ): maximal number of free variables in subformulae

Complexity of FO model checking

To decide whether $\mathfrak{A} \models \psi$, construct the game $\mathcal{G}(\mathfrak{A}, \psi)$ and check whether Verifier has winning strategy from initial position ψ.

Efficient implementation: on-the-fly construction of game while solving it

Size of game graph can be exponential: $|\mathcal{G}(\mathfrak{A}, \psi)| \leq |\psi| \cdot |A|^{\text{width}(\psi)}$

width(ψ): maximal number of free variables in subformulae

Complexity of FO model checking:

alternating time: $O(|\psi| + \text{qd}(\psi) \log |A|)$ qd(ψ): quantifier-depth of ψ

alternating space: $O(\text{width}(\psi) \cdot \log |A| + \log |\psi|)$

Complexity of FO model checking

To decide whether $\mathfrak{A} \models \psi$, construct the game $\mathcal{G}(\mathfrak{A}, \psi)$ and check whether Verifier has winning strategy from initial position ψ.

Efficient implementation: on-the-fly construction of game while solving it

Size of game graph can be exponential: $|\mathcal{G}(\mathfrak{A}, \psi)| \leq |\psi| \cdot |A|^{\mathrm{width}(\psi)}$

width(ψ): maximal number of free variables in subformulae

Complexity of FO model checking:

alternating time: $O(|\psi| + \mathrm{qd}(\psi) \log |A|)$ qd(ψ): quantifier-depth of ψ

alternating space: $O(\mathrm{width}(\psi) \cdot \log |A| + \log |\psi|)$

deterministic time: $O(|\psi| \cdot |A|^{\mathrm{width}(\psi)})$

deterministic space: $O(|\psi| + \mathrm{qd}(\psi) \log |A|)$

Complexity of FO model checking

- **Structure complexity** (ψ fixed) : ALOGTIME $\subseteq$ LOGSPACE
- **Expression complexity** and **combined complexity:** PSPACE

Complexity of FO model checking

- **Structure complexity** (ψ fixed) : ALOGTIME $\subseteq$ LOGSPACE
- **Expression complexity** and **combined complexity:** PSPACE

Crucial parameter for complexity: width of formula

$$\mathrm{FO}^k := \{\psi \in \mathrm{FO} : \mathrm{width}(\psi) \leq k\} = k\text{-variable fragment of FO}$$

Complexity of FO model checking

- **Structure complexity** (ψ fixed) : $\quad$ ALOGTIME $\subseteq$ LOGSPACE
- **Expression complexity** and **combined complexity:** $\quad$ PSPACE

Crucial parameter for complexity: width of formula

$$\mathrm{FO}^k := \{\psi \in \mathrm{FO} : \mathrm{width}(\psi) \leq k\} = k\text{-variable fragment of FO}$$

ModCheck(FO^k) is PTIME-complete and solvable in time $O(|\psi| \cdot |A|^k)$

Complexity of FO model checking

- **Structure complexity** (ψ fixed) : ALOGTIME $\subseteq$ LOGSPACE
- **Expression complexity** and **combined complexity:** PSPACE

Crucial parameter for complexity: width of formula

$$\mathrm{FO}^k := \{\psi \in \mathrm{FO} : \mathrm{width}(\psi) \leq k\} = k\text{-variable fragment of FO}$$

ModCheck(FO^k) is PTIME-complete and solvable in time $O(|\psi| \cdot |A|^k)$

Fragments of FO with model checking complexity $O(|\psi| \cdot \|\mathfrak{A}\|)$:
— **ML** : propositional modal logic
— **FO^2** : formulae of width two
— **GF** : the guarded fragment of first-order logic

The guarded fragment of first-order logic (GF)

Fragment of first-order logic with only guarded quantification

$$\exists \bar{y}(\alpha(\bar{x}, \bar{y}) \land \varphi(\bar{x}, \bar{y})) \qquad \forall \bar{y}(\alpha(\bar{x}, \bar{y}) \rightarrow \varphi(\bar{x}, \bar{y}))$$

with guards α : atomic formulae containing all free variables of φ

The guarded fragment of first-order logic (GF)

Fragment of first-order logic with only guarded quantification

$$\exists \bar{y}(\alpha(\bar{x}, \bar{y}) \wedge \varphi(\bar{x}, \bar{y})) \qquad \forall \bar{y}(\alpha(\bar{x}, \bar{y}) \rightarrow \varphi(\bar{x}, \bar{y}))$$

with guards α : atomic formulae containing all free variables of φ

Generalizes modal quantification: $\mathrm{ML} \subseteq \mathrm{GF} \subseteq \mathrm{FO}$

$$\langle a \rangle \varphi \equiv \exists y(E_a xy \wedge \varphi(y)) \qquad [a]\varphi \equiv \forall y(E_a xy \rightarrow \varphi(y))$$

Guarded logics generalize and, to some extent, explain the good algorithmic and model-theoretic properties of modal logics.

Model-theoretic and algorithmic properties of GF

- Satisfiability for GF is decidable (Andréka, van Benthem, Németi)
- GF has finite model property (Grädel)
- GF has (generalized) tree model property:
  every satisfiable formula has model of small tree width (Grädel)
- Extension by fixed points remains decidable (Grädel, Walukiewicz)
- . . .
- Guarded logics have small model checking games:
  $\|\mathcal{G}(\mathfrak{A}, \psi)\| = O(|\psi| \cdot \|\mathfrak{A}\|)$
  $\Longrightarrow$ efficient game-based model checking algorithms

Advantages of game based approach to model checking

- intuitive top-down definition of semantics
  (very effective for teaching logic)
- versatile and general methodology,
  can be adapted to many logical formalisms
- isolates the real combinatorial difficulties of an evaluation problem,
  abstracts from syntactic details.
- if you understand games, you understand alternating algorithms
- closely related to automata based methods
- algorithms and complexity results for many logic problems follow
  from results on games

Logics and games

First-order logic (FO) or **modal logic (ML):** Model checking games have

- only finite plays
- positional winning condition

Winning regions computable in linear time wrt. size of game graph

Logics and games

First-order logic (FO) or **modal logic (ML):** Model checking games have

- only finite plays
- positional winning condition

Winning regions computable in linear time wrt. size of game graph

In many computer science applications, more expressive logics are needed:

temporal logics, dynamic logics, fixed-point logics,. . .

Model checking games for these logics admit infinite plays and need more complicated winning conditions.

Logics and games

First-order logic (FO) or **modal logic (ML):** Model checking games have

- only finite plays
- positional winning condition

Winning regions computable in linear time wrt. size of game graph

In many computer science applications, more expressive logics are needed:

temporal logics, dynamic logics, fixed-point logics,. . .

Model checking games for these logics admit infinite plays and need more complicated winning conditions.

$\Longrightarrow$ we have to consider the theory of infinite games

Parity games

$G = (V, E, \Omega), \qquad V = V_0 \cup V_1, \qquad \Omega : V \longrightarrow \mathbb{N}$

Player 0 moves at positions $v \in V_0$, Player 1 at positions $v \in V_1$

$\Omega(v)$ is the priority of position v

Parity games

$G = (V, E, \Omega)$, $\quad V = V_0 \cup V_1$, $\quad \Omega : V \longrightarrow \mathbb{N}$

Player 0 moves at positions $v \in V_0$, Player 1 at positions $v \in V_1$

$\Omega(v)$ is the priority of position v

Play: finite or infinite sequence $\pi = v_0 v_1 v_2 \cdots$ with $(v_i, v_{i+1}) \in E$

Parity games

$G = (V, E, \Omega), \qquad V = V_0 \cup V_1, \qquad \Omega : V \longrightarrow \mathbb{N}$

Player 0 moves at positions $v \in V_0$, Player 1 at positions $v \in V_1$

$\Omega(v)$ is the priority of position v

Play: finite or infinite sequence $\pi = v_0 v_1 v_2 \cdots$ with $(v_i, v_{i+1}) \in E$

Winning condition:

– finite plays: who cannot move, loses

– infinite plays: least priority seen infinitely often determines winner

$$\text{Player 0 wins } \pi \iff \min\{k : (\exists^\infty i)\Omega(v_i) = k\} \text{ is even}$$

Parity games

$G = (V, E, \Omega)$, $\qquad V = V_0 \cup V_1$, $\qquad \Omega : V \longrightarrow \mathbb{N}$

Player 0 moves at positions $v \in V_0$, Player 1 at positions $v \in V_1$

$\Omega(v)$ is the priority of position v

Play: finite or infinite sequence $\pi = v_0 v_1 v_2 \cdots$ with $(v_i, v_{i+1}) \in E$

Winning condition:

– finite plays: who cannot move, loses

– infinite plays: least priority seen infinitely often determines winner

$$\text{Player 0 wins } \pi \iff \min\{k : (\exists^\infty i)\Omega(v_i) = k\} \text{ is even}$$

Winning regions W_0, W_1:

$$W_i = \{v \in V : \text{Player } i \text{ has winning strategy from position } v\}$$

Least fixed point logics

Extend a basic logical formalism by least and greatest fixed points

FO	(first-order logic)	$\longrightarrow$	LFP	(least fixed point logic)
ML	(modal logic)	$\longrightarrow$	L_μ	(modal μ-calculus)
GF	(guarded fragment)	$\longrightarrow$	μGF	(guarded fixed point logic)
conjunctive queries		$\longrightarrow$	Datalog / Stratified Datalog	

Least fixed point logics

Extend a basic logical formalism by least and greatest fixed points

FO	(first-order logic)	$\longrightarrow$	**LFP**	(least fixed point logic)
ML	(modal logic)	$\longrightarrow$	L_μ	(modal μ-calculus)
GF	(guarded fragment)	$\longrightarrow$	μ**GF**	(guarded fixed point logic)
conjunctive queries		$\longrightarrow$	Datalog / Stratified Datalog	

Idea: Capture recursion.

For any definable monotone relational operator

$$F_\varphi : T \mapsto \{\bar{x} : \varphi(T, \bar{x})\}$$

make also the least and the greatest fixed point of F_φ definable:

$$[\mathbf{lfp}\, T\bar{x} \,.\, \varphi(T, \bar{x})](\bar{z}) \qquad [\mathbf{gfp}\, T\bar{x} \,.\, \varphi(T, \bar{x})](\bar{z})$$

$$\mu X \,.\, \varphi \qquad \nu X \,.\, \varphi$$

Greatest fixed points (in LFP)

$[\mathbf{gfp}\, T\overline{x}\,.\,\varphi(T,\overline{x})](\overline{a})$: $\overline{a}$ contained in greatest T with $T = \{\overline{x} : \varphi(T,\overline{x})\}$

Greatest fixed points (in LFP)

$[\mathbf{gfp}\ T\overline{x}\,.\,\varphi(T,\overline{x})](\overline{a})$: $\overline{a}$ contained in greatest T with $T = \{\overline{x} : \varphi(T,\overline{x})\}$

this T exists if $F_\varphi : T \mapsto \{\overline{x} : \varphi(T,\overline{x})\}$ is monotone (preserves $\subseteq$)

to guarantee monotonicity: require that T positive in φ

Greatest fixed points (in LFP)

$[\mathbf{gfp}\, T\overline{x} \,.\, \varphi(T,\overline{x})](\overline{a})$: $\overline{a}$ contained in greatest T with $T = \{\overline{x} : \varphi(T,\overline{x})\}$

this T exists if $F_\varphi : T \mapsto \{\overline{x} : \varphi(T,\overline{x})\}$ is monotone (preserves $\subseteq$)
to guarantee monotonicity: require that T positive in φ

Inductive construction of the greatest fixed point on a structure $\mathfrak{A}$:

$$T^0 := A^k \quad \text{(all tuples of appropriate arity)}$$

$$T^{\alpha+1} := F_\varphi(T^\alpha)$$

$$T^\lambda := \bigcap_{\alpha<\lambda} T^\alpha \quad (\lambda \text{ limit ordinal})$$

$\Longrightarrow$ decreasing sequence of stages $(T^\alpha \supseteq T^{\alpha+1})$,
converges to a fixed point T^∞ of F_φ

Greatest fixed points (in LFP)

$[\mathbf{gfp}\, T\overline{x}\,.\,\varphi(T,\overline{x})](\overline{a})$: $\overline{a}$ contained in greatest T with $T = \{\overline{x} : \varphi(T,\overline{x})\}$

this T exists if $F_\varphi : T \mapsto \{\overline{x} : \varphi(T,\overline{x})\}$ is monotone (preserves $\subseteq$)
to guarantee monotonicity: require that T positive in φ

Inductive construction of the greatest fixed point on a structure $\mathfrak{A}$:

$$T^0 := A^k \quad \text{(all tuples of appropriate arity)}$$

$$T^{\alpha+1} := F_\varphi(T^\alpha)$$

$$T^\lambda := \bigcap_{\alpha<\lambda} T^\alpha \quad (\lambda \text{ limit ordinal})$$

$\Longrightarrow$ decreasing sequence of stages $(T^\alpha \supseteq T^{\alpha+1})$,
converges to a fixed point T^∞ of F_φ

Fact: $T^\infty = \mathbf{gfp}(F_\varphi)$ (Knaster, Tarski)

Example: Bisimulation

$\mathcal{K} = (V, E, P_1, \dots, P_m)$ transition system

Bisimilarity on $\mathcal{K}$ is the greatest equivalence relation $Z \subseteq V \times V$ such that:

if $(u, v) \in Z$ then

– u and v have the same atomic properties

– from u and v there are edges into the same equivalence classes

Example: Bisimulation

$\mathcal{K} = (V, E, P_1, \ldots, P_m)$ transition system

Bisimilarity on $\mathcal{K}$ is the greatest equivalence relation $Z \subseteq V \times V$ such that:

if $(u, v) \in Z$ then

– u and v have the same atomic properties

– from u and v there are edges into the same equivalence classes

Thus, bisimilarity is the greatest fixed point of the refinement operator

$$Z \quad \mapsto \quad \{(u, v) : \mathcal{K} \models \varphi(Z, u, v)\} \quad \text{where}$$

$$\varphi := \bigwedge_{i \leq m} P_i u \leftrightarrow P_i v \wedge$$
$$\forall x\,(Eux \rightarrow \exists y(Evy \wedge Zxy)) \;\wedge\; \forall y(Evy \rightarrow \exists x(Eux \wedge Zxy))$$

u and v are bisimilar in $\mathcal{K} \quad \Longleftrightarrow \quad \mathcal{K} \models [\mathbf{gfp}\, Zuv \,.\, \varphi](u, v)$

Least fixed point logic LFP

Syntax. LFP extends FO by fixed point rule:

- For every formula $\psi(T, x_1 \dots x_k) \in \text{LFP}[\tau \cup \{T\}]$,
  T k-ary relation variable, occuring only positive in ψ,
  build formulae $[\mathbf{lfp}\ T\overline{x}\,.\,\psi](\overline{x})$ and $[\mathbf{gfp}\ T\overline{x}\,.\,\psi](\overline{x})$

Semantics. On τ-structure $\mathfrak{A}$, $\psi(T, \overline{x})$ defines monotone operator

$$\begin{aligned} \psi^{\mathfrak{A}} : \mathcal{P}(A^k) &\longrightarrow \mathcal{P}(A^k) \\ T &\longmapsto \{\overline{a} : (\mathfrak{A}, T) \models \psi(T, \overline{a})\} \end{aligned}$$

- $\mathfrak{A} \models [\mathbf{lfp}\ T\overline{x}\,.\,\psi(T, \overline{x})](\overline{a}) \quad :\Longleftrightarrow \overline{a} \in \mathbf{lfp}(\psi^{\mathfrak{A}})$
  $\mathfrak{A} \models [\mathbf{gfp}\ T\overline{x}\,.\,\psi(T, \overline{x})](\overline{a}) \quad :\Longleftrightarrow \overline{a} \in \mathbf{gfp}(\psi^{\mathfrak{A}})$

Modal μ-calculus L_μ

Syntax. L_μ extends ML by fixed point rule:

- With every formula $\psi(X)$, where X occurs only positive in ψ L_μ also contains the formulae $\mu X.\psi$ and $\nu X.\psi$

Semantics. On transition system $\mathcal{K}$, $\psi(X)$ defines operator

$$\psi^{\mathcal{K}} : X \longmapsto [\![\psi]\!]^{(\mathcal{K},X)} = \{v : (\mathcal{K}, X), v \models \psi\}$$

$\psi^{\mathcal{K}}$ is monotone, and therefore has a least and a greatest fixed point

$$\mathbf{lfp}(\psi^{\mathcal{K}}) = \bigcap\{X : \psi^{\mathcal{K}}(X) \subseteq X\}, \qquad \mathbf{gfp}(\psi^{\mathcal{K}}) = \bigcup\{X : X \subseteq \psi^{\mathcal{K}}(X)\}$$

- $[\![\mu X.\psi]\!]^{\mathcal{K}} := \mathbf{lfp}(\psi^{\mathcal{K}}), \qquad [\![\nu X.\psi]\!]^{\mathcal{K}} := \mathbf{gfp}(\psi^{\mathcal{K}})$

Inductive generation of fixed points

$\psi(X)$ defines operator $\psi^{\mathcal{K}} : X \mapsto \{v : (\mathcal{K}, X), v \models \psi\}$

$$X^0 := \varnothing \qquad\qquad Y^0 := V$$

$$X^{\alpha+1} := \psi^{\mathcal{K}}(X^\alpha) \qquad\qquad Y^{\alpha+1} := \psi^{\mathcal{K}}(Y^\alpha)$$

$$X^\lambda := \bigcup_{\alpha<\lambda} X^\alpha \qquad (\lambda \text{ limit ordinal}) \qquad Y^\lambda := \bigcap_{\alpha<\lambda} Y^\alpha$$

$X^0 \subseteq \cdots \subseteq X^\alpha \subseteq X^{\alpha+1} \subseteq \cdots$ $\qquad$ $Y^0 \supseteq \cdots \supseteq Y^\alpha \supseteq Y^{\alpha+1} \supseteq \cdots$

Inductive generation of fixed points

$\psi(X)$ defines operator $\psi^{\mathcal{K}} : X \mapsto \{v : (\mathcal{K}, X), v \models \psi\}$

$$X^0 := \varnothing \qquad\qquad Y^0 := V$$

$$X^{\alpha+1} := \psi^{\mathcal{K}}(X^\alpha) \qquad\qquad Y^{\alpha+1} := \psi^{\mathcal{K}}(Y^\alpha)$$

$$X^\lambda := \bigcup_{\alpha<\lambda} X^\alpha \qquad (\lambda \text{ limit ordinal}) \qquad Y^\lambda := \bigcap_{\alpha<\lambda} Y^\alpha$$

$$X^0 \subseteq \cdots \subseteq X^\alpha \subseteq X^{\alpha+1} \subseteq \cdots \qquad\qquad Y^0 \supseteq \cdots \supseteq Y^\alpha \supseteq Y^{\alpha+1} \supseteq \cdots$$

These inductive sequences reach fixed points

$$X^\alpha = X^{\alpha+1} =: X^\infty, \qquad\qquad Y^\beta = Y^{\beta+1} =: Y^\infty$$

for some α, β, with $|\alpha|, |\beta| \leq |V|$

Inductive generation of fixed points

$\psi(X)$ defines operator $\psi^{\mathcal{K}} : X \mapsto \{v : (\mathcal{K}, X), v \models \psi\}$

$$X^0 := \varnothing \qquad\qquad Y^0 := V$$

$$X^{\alpha+1} := \psi^{\mathcal{K}}(X^\alpha) \qquad\qquad Y^{\alpha+1} := \psi^{\mathcal{K}}(Y^\alpha)$$

$$X^\lambda := \bigcup_{\alpha<\lambda} X^\alpha \qquad (\lambda \text{ limit ordinal}) \qquad Y^\lambda := \bigcap_{\alpha<\lambda} Y^\alpha$$

$$X^0 \subseteq \cdots \subseteq X^\alpha \subseteq X^{\alpha+1} \subseteq \cdots \qquad\qquad Y^0 \supseteq \cdots \supseteq Y^\alpha \supseteq Y^{\alpha+1} \supseteq \cdots$$

These inductive sequences reach fixed points

$$X^\alpha = X^{\alpha+1} =: X^\infty, \qquad\qquad Y^\beta = Y^{\beta+1} =: Y^\infty$$

for some α, β, with $|\alpha|, |\beta| \leq |V|$

$$X^\infty = [\![\mu X.\psi]\!]^{\mathcal{K}} \qquad\qquad Y^\infty = [\![\nu X.\psi]\!]^{\mathcal{K}}$$

L_μ: Examples

- $\mathcal{K}, w \models \nu X . \langle a \rangle X \iff$ there is an infinite a-path from w in $\mathcal{K}$

  $\mathcal{K}, w \models \mu X . P \vee [a]X \iff$ every infinite a-path from w eventually hits P

L_μ: Examples

- $\mathcal{K}, w \models \nu X . \langle a \rangle X \iff$ there is an infinite a-path from w in $\mathcal{K}$

  $\mathcal{K}, w \models \mu X . P \vee [a]X \iff$ every infinite a-path from w eventually hits P

- $\mathcal{K}, w \models \nu X \, \mu Y . \Diamond((P \wedge X) \vee Y) \iff$ on some path from w, P occurs infinitely often

L_μ: Examples

- $\mathcal{K}, w \models \nu X \,.\, \langle a \rangle X \quad \Longleftrightarrow$ there is an infinite a-path from w in $\mathcal{K}$

  $\mathcal{K}, w \models \mu X \,.\, P \vee [a]X \quad \Longleftrightarrow$ every infinite a-path from w eventually hits P

- $\mathcal{K}, w \models \nu X \, \mu Y \,.\, \Diamond((P \wedge X) \vee Y) \quad \Longleftrightarrow$ on some path from w, P occurs infinitely often

- Logics of knowledge: multi-modal propositional logics where $[a]\varphi$ stands for "agent a knows φ"

  add common knowledge:

  everybody knows φ, and everybody knows that everybody knows φ, and everybody knows that everybody knows that everybody knows ...

  expressible as a greatest fixed point: $C\varphi \;\equiv\; \nu X \,.\, (\varphi \wedge \bigwedge_a [a]X)$

Finite games and LFP

- GAME is definable in LFP / L_μ

Player 0 has winning strategy for game $\mathcal{G}$ from position v

$$\Longleftrightarrow$$

$$\mathcal{G} = (V, V_0, V_1, E) \models [\mathbf{lfp}\, Wx\,.\,(V_0x \wedge \exists y(Exy \wedge Wy)) \\ \vee (V_1x \wedge \forall y(Exy \rightarrow Wy)](v)$$

$$\Longleftrightarrow$$

$$\mathcal{G}, v \models \mu W\,.\,(V_0 \wedge \Diamond W) \vee (V_1 \wedge \Box W)$$

Finite games and LFP

- $\textsc{Game}$ is definable in LFP / L_μ

Player 0 has winning strategy for game $\mathcal{G}$ from position v

$$\iff$$

$$\mathcal{G} = (V, V_0, V_1, E) \models [\mathbf{lfp}\, Wx\,.\,(V_0x \wedge \exists y(Exy \wedge Wy)) \\ \vee (V_1x \wedge \forall y(Exy \rightarrow Wy))](v)$$

$$\iff$$

$$\mathcal{G}, v \models \mu W\,.\,(V_0 \wedge \Diamond W) \vee (V_1 \wedge \Box W)$$

- $\textsc{Game}$ is complete for LFP
(via quantifier-free reductions on finite structures)

Importance of the modal μ-calculus

- encompasses most of the popular logics used in hardware verification: LTL, CTL, CTL*, PDL,. . . , and also many logics from other fields: game logic, description logics, etc.

- reasonably good algorithmic properties:
  - satisfiability problem decidable (Exptime-complete)
  - efficient model checking for practically important fragments of L_μ
  - automata-based algorithms

- nice model-theoretic properties:
  - finite model property
  - tree model property

- L_μ is the bisimulation-invariant fragment of MSO

Importance of the modal μ-calculus

- encompasses most of the popular logics used in hardware verification: LTL, CTL, CTL*, PDL,. . . , and also many logics from other fields: game logic, description logics, etc.

- reasonably good algorithmic properties:
  - satisfiability problem decidable (Exptime-complete)
  - efficient model checking for practically important fragments of L_μ
  - automata-based algorithms

- nice model-theoretic properties:
  - finite model property
  - tree model property

- L_μ is the bisimulation-invariant fragment of MSO

Disadvantage: Fixed-point formulae are hard to read

Model checking games for LFP and L_μ

LFP-game: extend FO-game by moves

$$\begin{aligned} [\mathbf{fp}\, T\overline{x}\,.\,\varphi](\overline{a}) &\longrightarrow \varphi(T, \overline{a}) \qquad (\mathbf{fp} \in \{\mathbf{lfp}, \mathbf{gfp}\}) \\ T\overline{b} &\longrightarrow \varphi(T, \overline{b}) \end{aligned}$$

Similarly for L_μ: extend ML-game by moves

$$\begin{aligned} (\lambda X\,.\,\varphi, u) &\longrightarrow (\varphi, u) \qquad (\lambda \in \{\mu, \nu\}) \\ (X, w) &\longrightarrow (\varphi, w) \end{aligned}$$

Model checking games for LFP and L_μ

LFP-game: extend FO-game by moves

$$[\mathbf{fp}\, T\overline{x}\,.\,\varphi](\overline{a}) \longrightarrow \varphi(T,\overline{a}) \qquad (\mathbf{fp} \in \{\mathbf{lfp}, \mathbf{gfp}\})$$

$$T\overline{b} \longrightarrow \varphi(T,\overline{b})$$

Similarly for L_μ: extend ML-game by moves

$$(\lambda X\,.\,\varphi, u) \longrightarrow (\varphi, u) \qquad (\lambda \in \{\mu, \nu\})$$

$$(X, w) \longrightarrow (\varphi, w)$$

Infinite plays possible

Model checking games for LFP and L_μ

LFP-game: extend FO-game by moves

$$[\mathbf{fp}\, T\overline{x}\,.\,\varphi](\overline{a}) \longrightarrow \varphi(T,\overline{a}) \qquad (\mathbf{fp} \in \{\mathbf{lfp}, \mathbf{gfp}\})$$

$$T\overline{b} \longrightarrow \varphi(T,\overline{b})$$

Similarly for L_μ: extend ML-game by moves

$$(\lambda X\,.\,\varphi, u) \longrightarrow (\varphi, u) \qquad (\lambda \in \{\mu, \nu\})$$

$$(X, w) \longrightarrow (\varphi, w)$$

Infinite plays possible

need winning condition for infinite plays

Model checking game for L_μ: Example

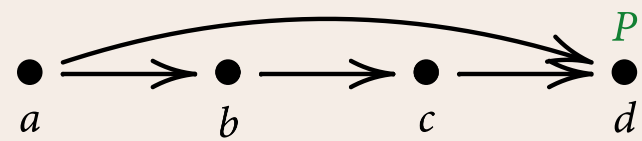
$$\psi = \mu X.P \lor \Box X \qquad \equiv \qquad [\mathbf{lfp}\, Tx\,.\, Px \lor \forall y(Exy \rightarrow Ty)](x)$$

$$\mathcal{K}: \qquad \begin{tikzcd} \underset{a}{\bullet} \arrow[r] \arrow[rrr, bend left] & \underset{b}{\bullet} \arrow[r] & \underset{c}{\bullet} \arrow[r] & \overset{P}{\underset{d}{\bullet}} \end{tikzcd} \qquad \text{from node } a$$

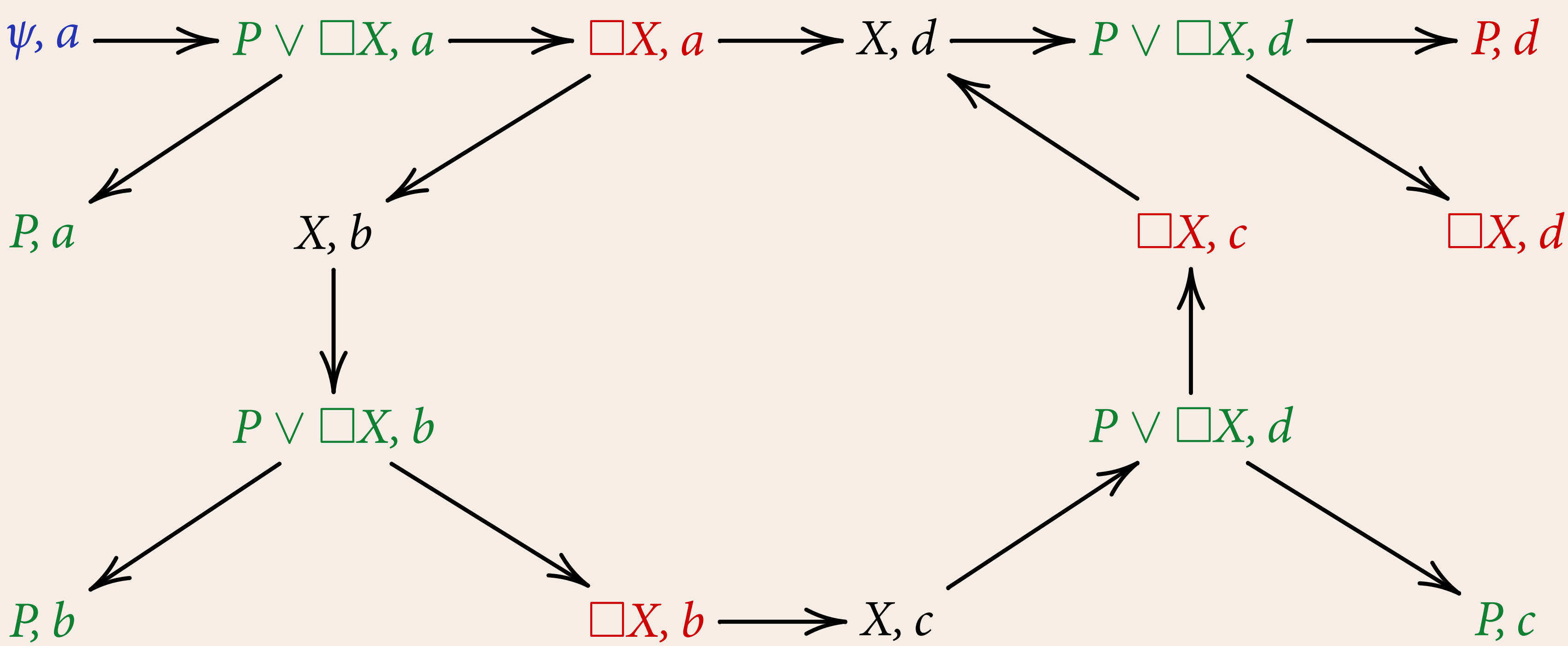
Model checking game for L_μ: Example

$$\psi = \mu X.P \lor \Box X \qquad \equiv \qquad [\mathbf{lfp}\, Tx\,.\, Px \lor \forall y(Exy \rightarrow Ty)](x)$$

$\mathcal{K}$:



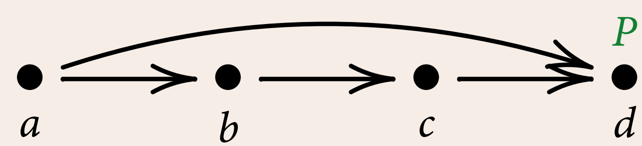
from node a



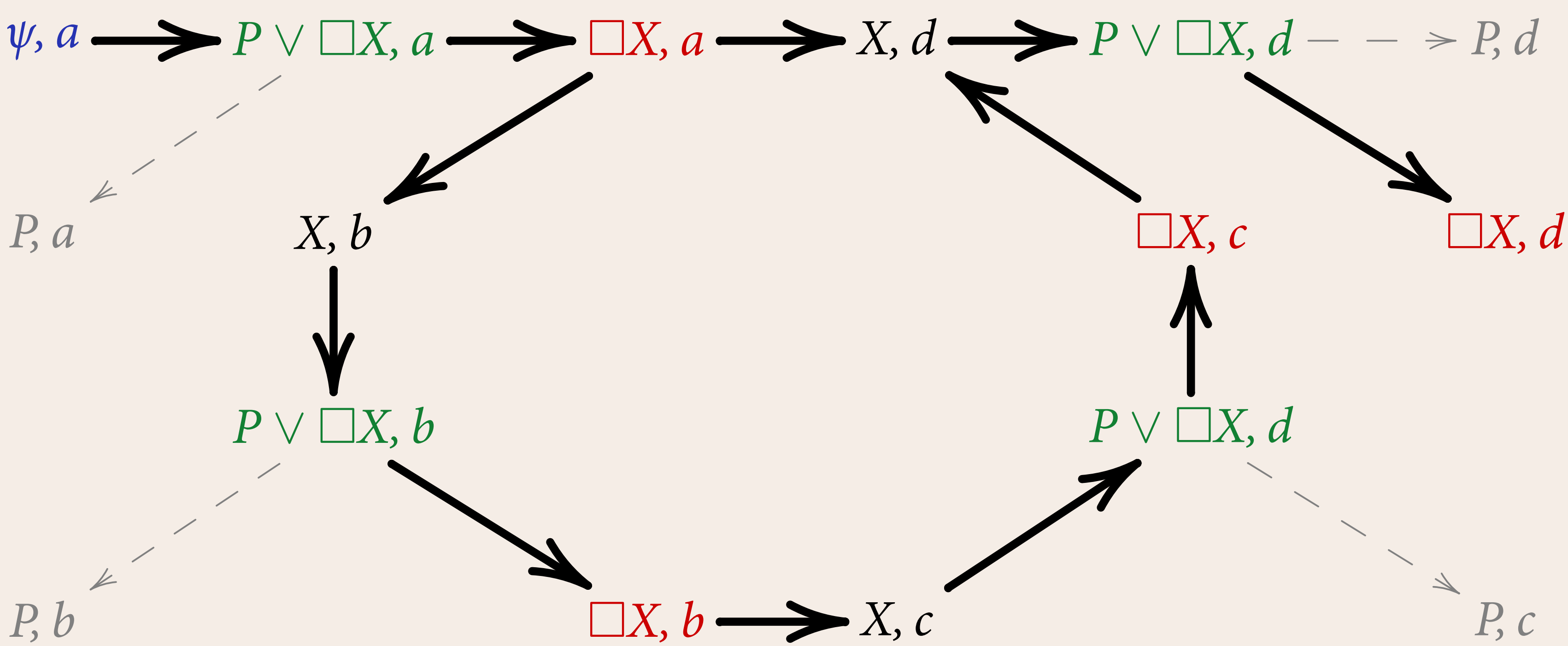
Model checking game for L_μ: Example

$$\psi = \mu X.P \lor \Box X \qquad \equiv \qquad [\mathbf{lfp}\, Tx\,.\, Px \lor \forall y(Exy \rightarrow Ty)](x)$$

$\mathcal{K}$:



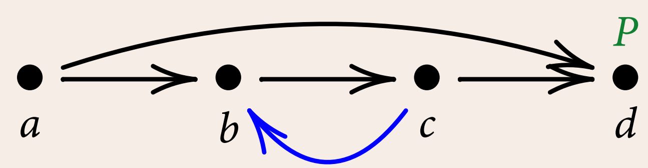
from node a



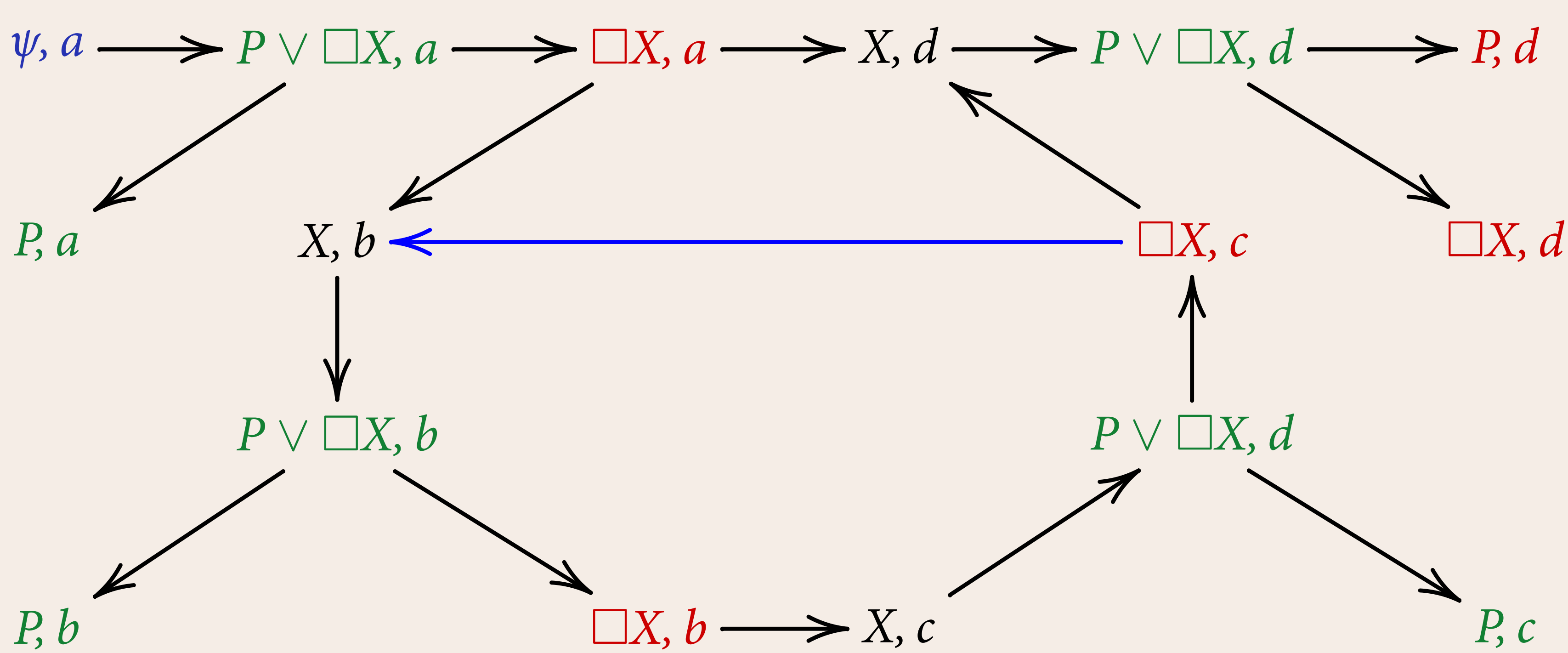
Model checking game for L_μ: Example

$\psi = \mu X.P \vee \Box X \qquad \equiv \qquad [\mathbf{lfp}\, Tx\,.\, Px \vee \forall y(Exy \rightarrow Ty)](x)$

$\mathcal{K}$:



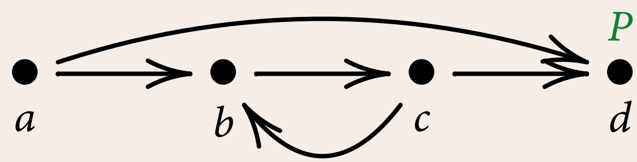
from node a



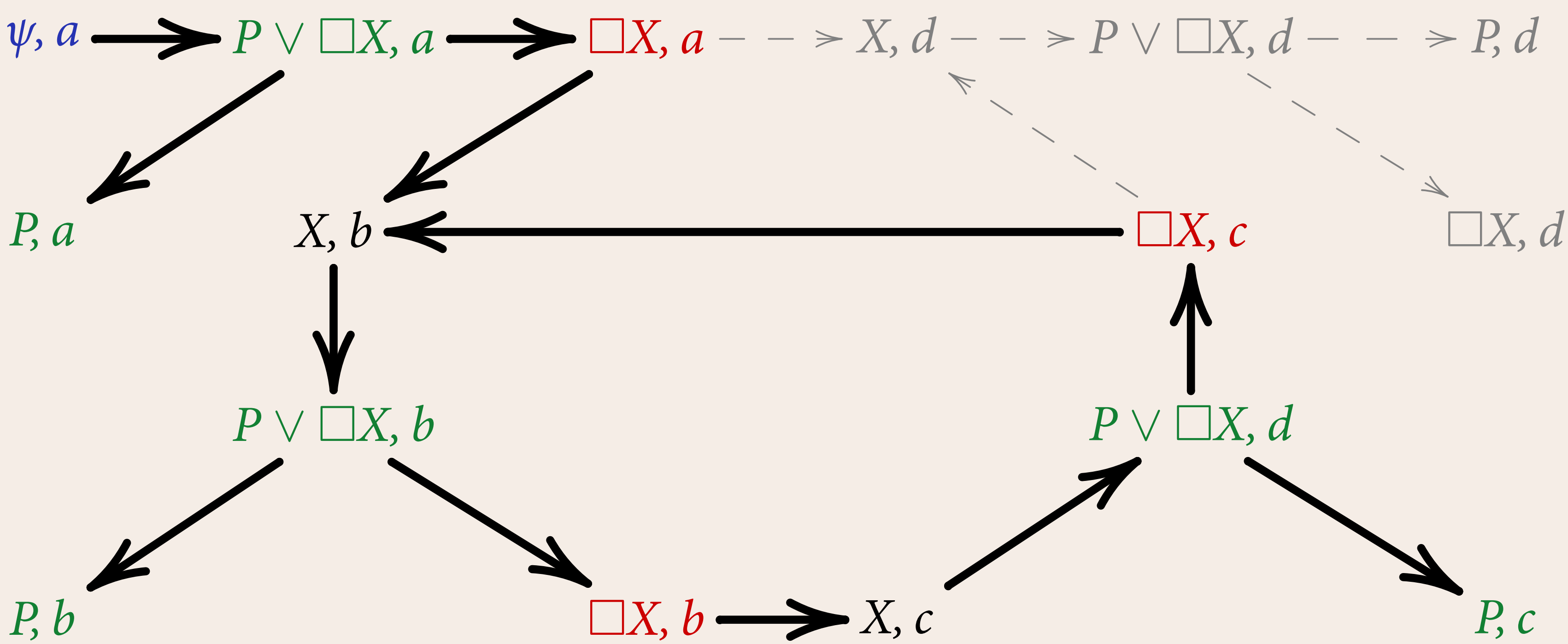
Model checking game for L_μ: Example

$\psi = \mu X.P \lor \Box X \quad \equiv \quad [\mathbf{lfp}\, Tx\,.\, Px \lor \forall y(Exy \to Ty)](x)$

$\mathcal{K}$:



from node a



Winning conditions

On formulae $[\mathbf{lfp}\ T\overline{x}\,.\,\psi(T,\overline{x})](\overline{a})$ or $\mu X.\psi$ (where ψ has no fixed points), Verifier must win in a finite number of steps.

By forcing a cycle, Falsifier wins.

Winning conditions

On formulae $[\mathbf{lfp}\ T\overline{x}\,.\ \psi(T,\overline{x})](\overline{a})$ or $\mu X.\psi$ (where ψ has no fixed points), Verifier must win in a finite number of steps.

By forcing a cycle, Falsifier wins.

Are cycles always bad for Verifier?

Winning conditions

On formulae $[\mathbf{lfp}\ T\overline{x}\,.\,\psi(T,\overline{x})](\overline{a})$ or $\mu X.\psi$ (where ψ has no fixed points), Verifier must win in a finite number of steps.

By forcing a cycle, Falsifier wins.

Are cycles always bad for Verifier?

No, not if they correspond to greatest fixed points

Winning conditions

On formulae $[\mathbf{lfp}\, T\overline{x}\,.\,\psi(T,\overline{x})](\overline{a})$ or $\mu X.\psi$ (where ψ has no fixed points), Verifier must win in a finite number of steps.

By forcing a cycle, Falsifier wins.

Are cycles always bad for Verifier?

No, not if they correspond to greatest fixed points

- **lfp**-cycles: Falsifier wins
- **gfp**-cycles: Verifier wins

Winning conditions

On formulae $[\mathbf{lfp}\ T\overline{x}\,.\ \psi(T,\overline{x})](\overline{a})$ or $\mu X.\psi$ (where ψ has no fixed points), Verifier must win in a finite number of steps.

By forcing a cycle, Falsifier wins.

Are cycles always bad for Verifier?

No, not if they correspond to greatest fixed points

- **lfp**-cycles: Falsifier wins
- **gfp**-cycles: Verifier wins

What about cycles with both least and greatest fixed points?

Winning conditions

On formulae $[\mathbf{lfp}\ T\overline{x}\,.\,\psi(T,\overline{x})](\overline{a})$ or $\mu X.\psi$ (where ψ has no fixed points), Verifier must win in a finite number of steps.

By forcing a cycle, Falsifier wins.

Are cycles always bad for Verifier?

No, not if they correspond to greatest fixed points

- **lfp**-cycles: Falsifier wins
- **gfp**-cycles: Verifier wins

What about cycles with both least and greatest fixed points?

The outermost fixed point on cycle determines the winner

Model checking games for LFP and L_μ

Extend FO-game by moves

$$\begin{aligned} [\mathbf{fp}\ T\bar{x}\,.\,\varphi](\bar{a}) &\longrightarrow \varphi(T,\bar{a}) \\ T\bar{a} &\longrightarrow \varphi(T,\bar{a}) \end{aligned}$$

Parity game, with following priority assignment:

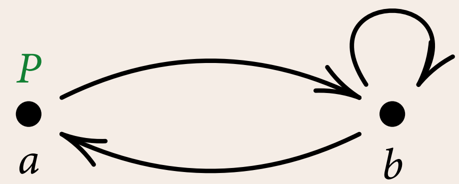
- $\Omega(T\bar{a})$ is $\begin{cases} \text{even} & \text{if } T \text{ } \mathbf{gfp}\text{-variable} \\ \text{odd} & \text{if } T \text{ } \mathbf{lfp}\text{-variable} \end{cases}$
- $\Omega(T\bar{a}) \leq \Omega(T'\bar{b})$ if T' depends on T
  (i.e. if T free in $[\mathbf{fp}\ T'\bar{x}\,.\,\varphi(T', T, \bar{x})](\bar{a})$)
- $\Omega(\varphi)$ maximal, for other formulae φ

Analogous for L_μ

Model checking game with nested cycles: Example

$\psi = \nu X \underbrace{\mu Y \,.\, \Diamond((P \wedge X) \vee Y)}_{\varphi} \equiv$ on some path, P occurs infinitely often

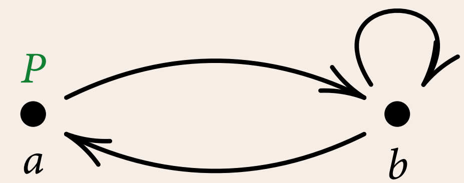
$\mathcal{K}:$

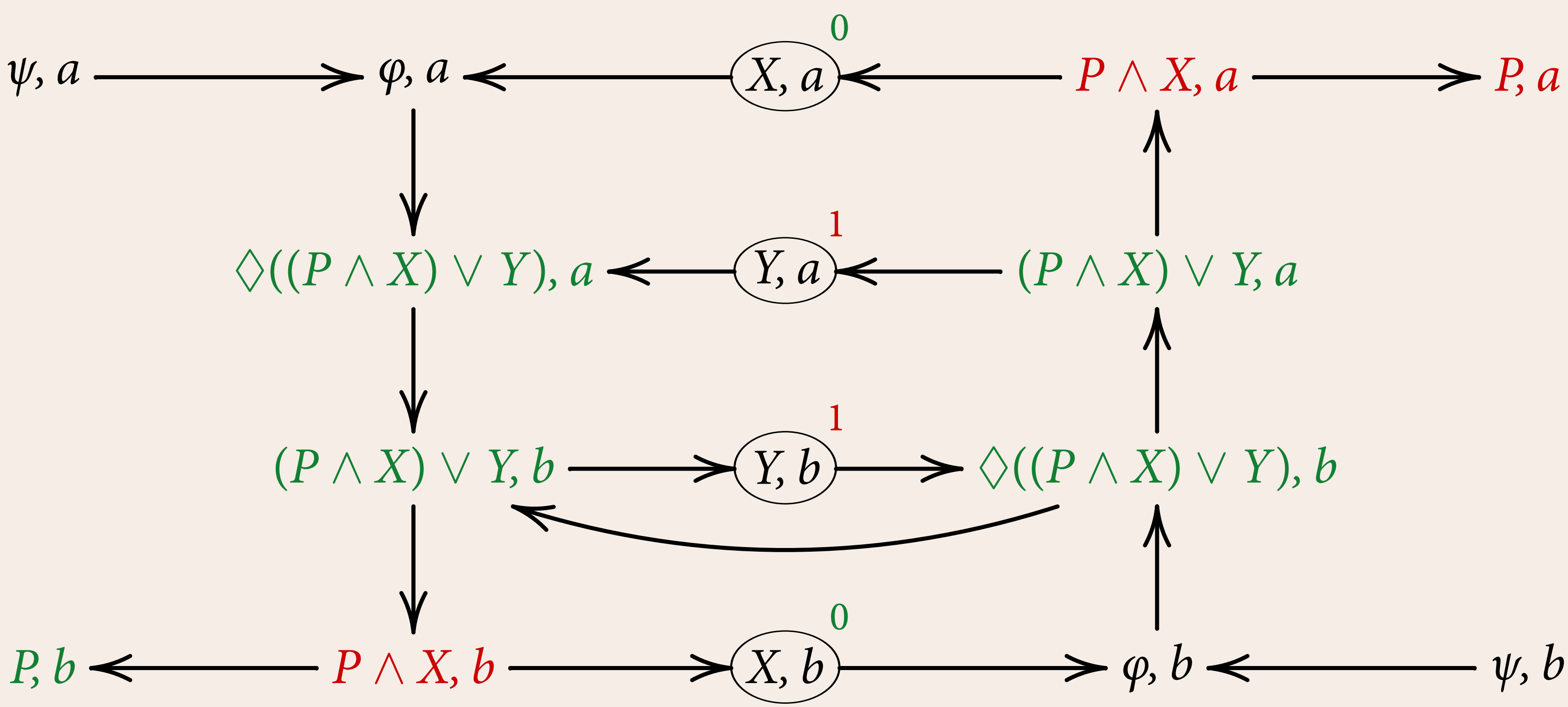


Model checking game with nested cycles: Example

$\psi = \nu X\, \underbrace{\mu Y . \Diamond((P \wedge X) \vee Y)}_{\varphi}$ $\equiv$ on some path, P occurs infinitely often

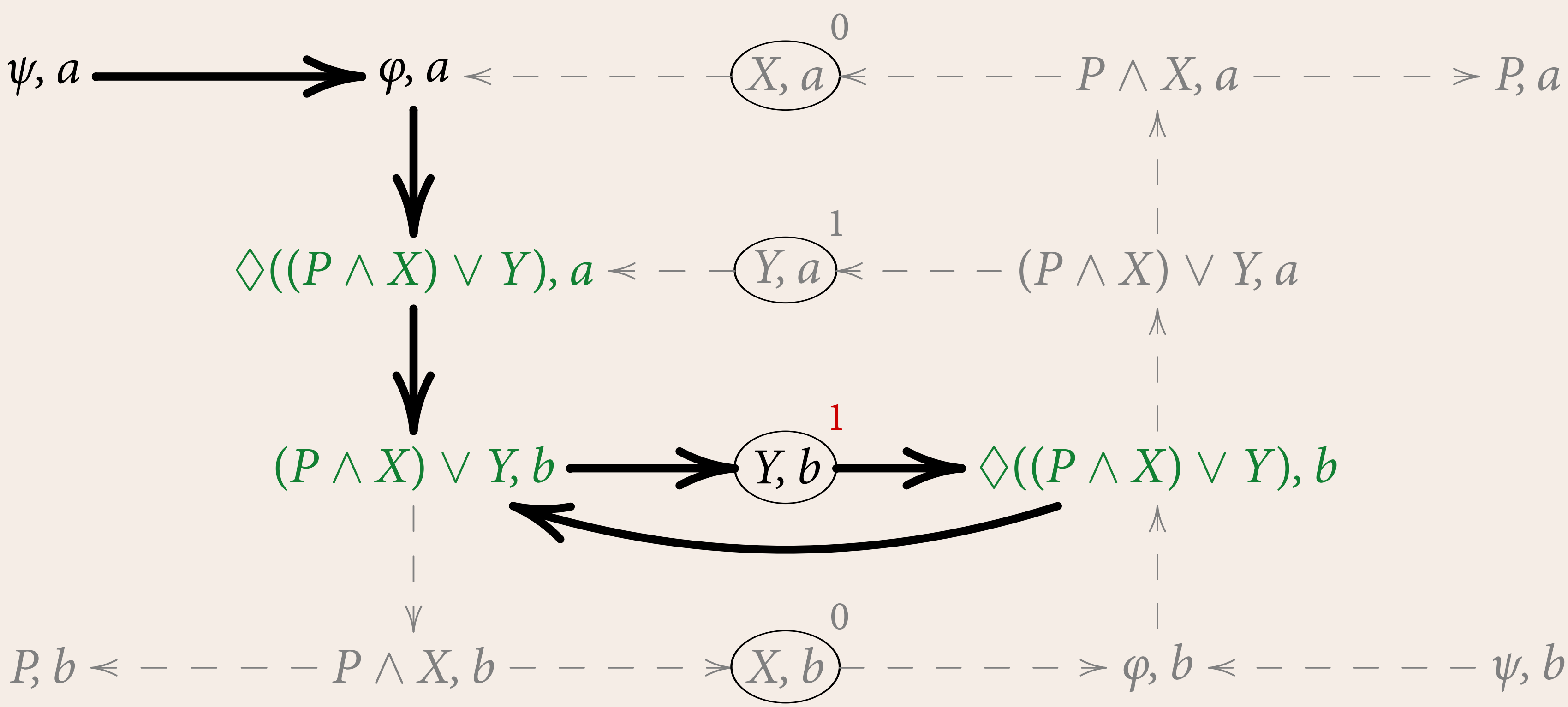
$\mathcal{K}$:





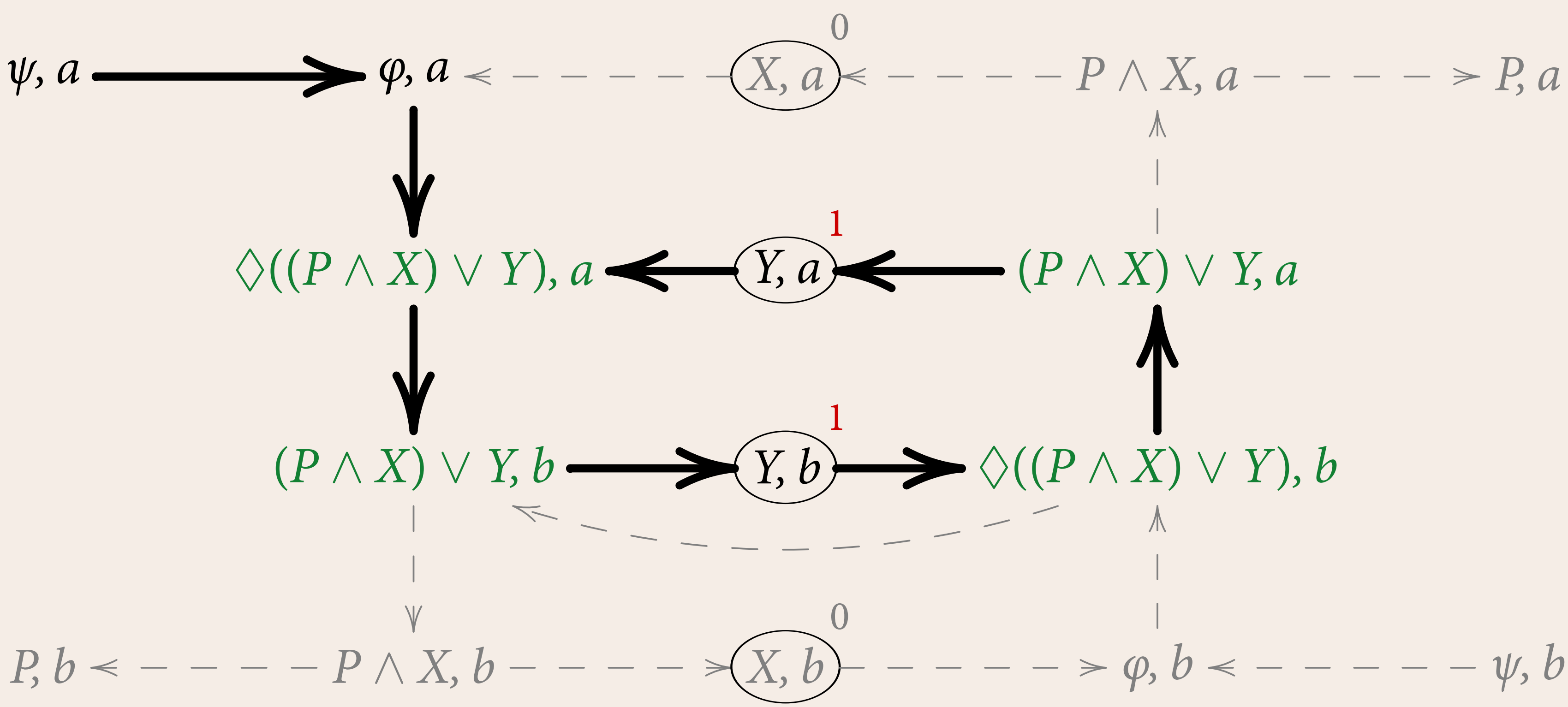
Model checking game with nested cycles: Example

Bad cycles for Verifier: Least priority is odd



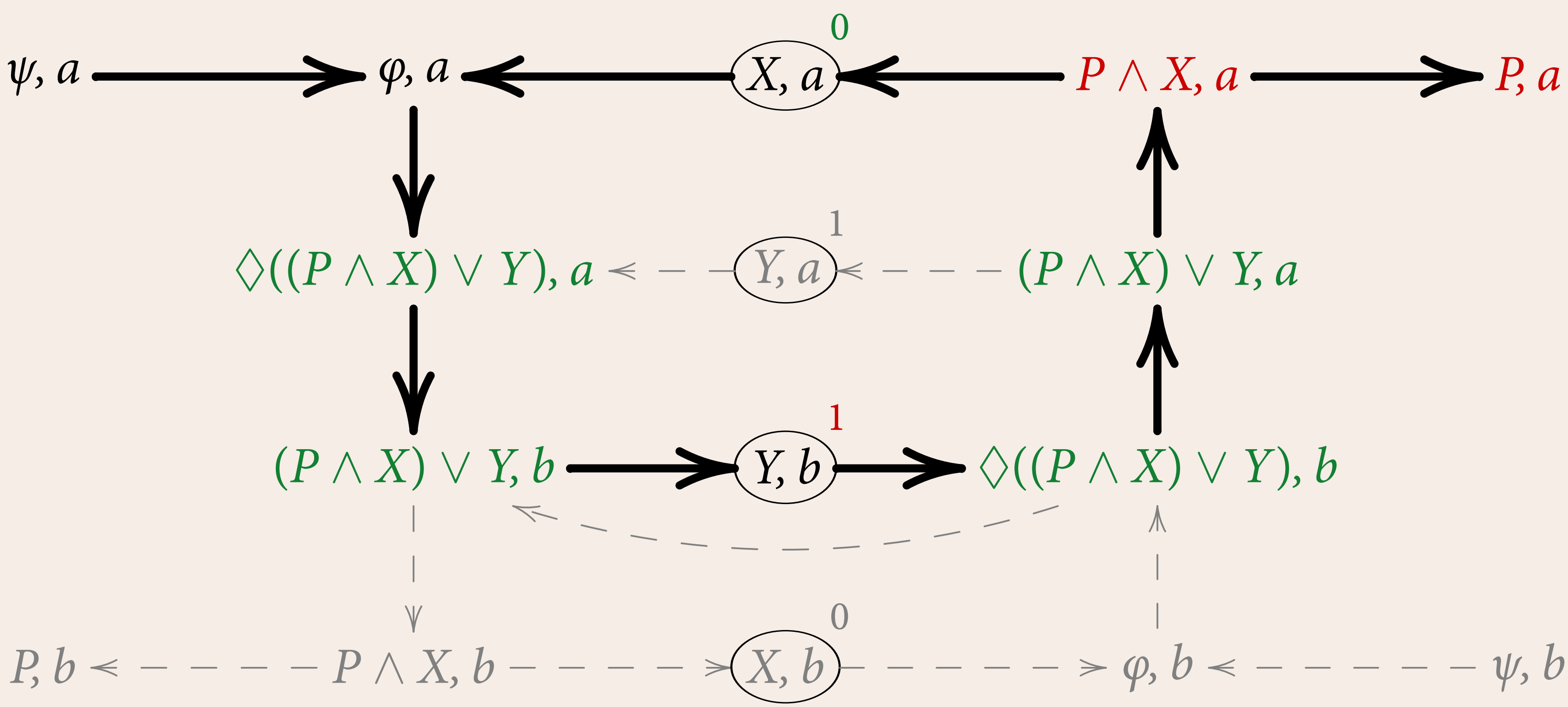
Model checking game with nested cycles: Example

Bad cycles for Verifier: Least priority is odd



Model checking game with nested cycles: Example

Winning strategy for Verifier



Defining winning regions of parity games in L_μ

Describe parity game with d priorities by transition system

$\mathcal{G} = (V, E, E_0, \dots, E_{d-1}, A_0, \dots, A_{d-1})$ where

$$E_i = \{u : \Omega(u) = i \text{ and Ego (Player 0) moves from } u\}$$

$$A_i = \{u : \Omega(u) = i \text{ and Alter (Player 1) moves from } u\}$$

Defining winning regions of parity games in L_μ

Describe parity game with d priorities by transition system

$\mathcal{G} = (V, E, E_0, \ldots, E_{d-1}, A_0, \ldots, A_{d-1})$ where

$$E_i = \{u : \Omega(u) = i \text{ and Ego (Player 0) moves from } u\}$$

$$A_i = \{u : \Omega(u) = i \text{ and Alter (Player 1) moves from } u\}$$

Define the formula

$$\mathrm{Win}_d := \nu X_0\, \mu X_1\, \nu X_2 \cdots \lambda X_{d-1} \bigvee_i \Big((E_i \wedge \Diamond X_i) \vee (A_i \wedge \Box X_i) \Big)$$

Defining winning regions of parity games in L_μ

Describe parity game with d priorities by transition system

$\mathcal{G} = (V, E, E_0, \dots, E_{d-1}, A_0, \dots, A_{d-1})$ where

$$E_i = \{u : \Omega(u) = i \text{ and Ego (Player 0) moves from } u\}$$

$$A_i = \{u : \Omega(u) = i \text{ and Alter (Player 1) moves from } u\}$$

Define the formula

$$\mathrm{Win}_d := \nu X_0\, \mu X_1\, \nu X_2 \cdots \lambda X_{d-1} \bigvee_i \Big((E_i \wedge \Diamond X_i) \vee (A_i \wedge \Box X_i) \Big)$$

Theorem. Player 0 wins $\mathcal{G}$ from position u $\iff$ $\mathcal{G}, u \models \mathrm{Win}_d$.

Defining winning regions of parity games in L_μ

Describe parity game with d priorities by transition system

$\mathcal{G} = (V, E, E_0, \dots, E_{d-1}, A_0, \dots, A_{d-1})$ where

$$E_i = \{u : \Omega(u) = i \text{ and Ego (Player 0) moves from } u\}$$

$$A_i = \{u : \Omega(u) = i \text{ and Alter (Player 1) moves from } u\}$$

Define the formula

$$\mathrm{Win}_d := \nu X_0\, \mu X_1\, \nu X_2 \cdots \lambda X_{d-1} \bigvee_i \Big((E_i \wedge \Diamond X_i) \vee (A_i \wedge \Box X_i) \Big)$$

Theorem. Player 0 wins $\mathcal{G}$ from position u $\iff$ $\mathcal{G}, u \models \mathrm{Win}_d$.

Proof. The model checking game for Win_d on $\mathcal{G}$ coincides (up to the presence of additional 'stupid' moves) with the game $\mathcal{G}$ itself !