# THEORIES — GAMES — ALGORITHMS

## Model Checking Games

Erich Grädel

graedel@rwth-aachen.de.

Aachen University

# 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

# 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 $L_\mu$):** 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 complete 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 $\sigma$ 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 $\sigma$: $f : \{v \in V_\sigma : vE \neq \varnothing\} \to V$ with $(v, f(v)) \in E$.

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

# Winning strategies and winning regions

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

$f$ is **winning from position $v$** if Player $\sigma$ 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 $\sigma$: $f : \{v \in V_\sigma : vE \neq \varnothing\} \to V$ with $(v, f(v)) \in E$.

$f$ is **winning from position $v$** if Player $\sigma$ 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 $\sigma$)

# Algorithms for finite games

## Theorem

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

# Algorithms for finite games

### Theorem

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

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 ∗)

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

**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**$(v, \sigma)$

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

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

  **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**$(u, \sigma)$

  **enddo**

# 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

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

# 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 ≡ deterministic space

Alternating space ≡ 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:** $\psi ::= R_i\overline{x} \mid \neg R_i\overline{x} \mid x = y \mid x \neq y \mid \psi \wedge \psi \mid \psi \vee \psi \mid \exists x\psi \mid \forall x\psi$

# 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 $\mathfrak{A} = (A, R_1, \dots, R_m)$, $R_i \subseteq A^{r_i}$)

# Evaluation game for FO

**FO:** $\psi ::= R_i\overline{x} \mid \neg R_i\overline{x} \mid x = y \mid x \neq y \mid \psi \land \psi \mid \psi \lor \psi \mid \exists x\psi \mid \forall x\psi$

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

**Positions:** $\varphi(\overline{a})$ $\varphi(\overline{x})$ subformula of $\psi$, $\overline{a} \in A^k$

# Evaluation game for FO

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

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

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

**Verifier moves:**

$$\varphi \vee \vartheta \longrightarrow \varphi, \qquad \varphi \vee \vartheta \longrightarrow \vartheta$$

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

**Falsifier moves:**

$$\varphi \wedge \vartheta \longrightarrow \varphi, \qquad \varphi \wedge \vartheta \longrightarrow \vartheta$$

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

# 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 $\mathfrak{A} = (A, R_1, \ldots, R_m)$, $R_i \subseteq A^{r_i}$)

**Positions:** $\varphi(\overline{a})$ $\varphi(\overline{x})$ subformula of $\psi$, $\overline{a} \in A^k$

**Verifier moves:**

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

**Falsifier moves:**

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

**Winning condition:** $\varphi$ atomic / negated atomic

$$\begin{array}{c} \text{Verifier} \\ \text{Falsifier} \end{array} \text{ wins at } \varphi(\overline{a}) \iff \mathfrak{A} \begin{array}{c} \models \\ \not\models \end{array} \varphi(\overline{a})$$

# 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 $\psi$.

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 $\psi$.

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($\psi$): 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 $\psi$.

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($\psi$): maximal number of free variables in subformulae

**Complexity of FO model checking:**

alternating time: $O(|\psi| \cdot \log |A|)$

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

# 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 $\psi$.

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($\psi$): maximal number of free variables in subformulae

**Complexity of FO model checking:**

alternating time: $O(|\psi| \cdot \log |A|)$

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

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

deterministic space: $O(|\psi| \cdot \log |A|)$

# Complexity of FO model checking

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

  PSPACE-complete (even for very small $\mathfrak{A}$, such as $\mathfrak{A} = \{0, 1\}$)

# Complexity of FO model checking

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

  PSPACE-complete (even for very small $\mathfrak{A}$, such as $\mathfrak{A} = \{0, 1\}$)

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** ($\psi$ fixed) : ALOGTIME $\subseteq$ LOGSPACE
- **Expression complexity** and **combined complexity:**

  PSPACE-complete (even for very small $\mathfrak{A}$, such as $\mathfrak{A} = \{0, 1\}$)

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

# Complexity of FO model checking

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

  PSPACE-complete (even for very small $\mathfrak{A}$, such as $\mathfrak{A} = \{0, 1\}$)

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($\mathrm{FO}^k$) 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

**Later:**

— **$\mathrm{FO}^2$** : formulae of width two

— **GF** : the guarded fragment of first-order logic

# ML: propositional modal logic

Transition systems = Kripke structures = labeled graphs

$$\mathcal{K} = (\underbrace{V}_{\substack{\text{states}\\\text{elements}}}, \underbrace{(E_a)_{a\in A}}_{\substack{\text{actions}\\\text{binary relations}}}, \underbrace{(P_i)_{i\in I}}_{\substack{\text{atomic propositions}\\\text{unary relations}}})$$

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

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

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

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

# Model checking game for ML

**Game** $\mathcal{G}(\mathcal{K}, \psi)$ for $\mathcal{K} = (V, (E_a)_{a \in A}, (P_i)_{i \in I})$ and $\psi \in \mathrm{ML}$

# Model checking game for ML

Game $\mathcal{G}(\mathcal{K}, \psi)$ for $\mathcal{K} = (V, (E_a)_{a \in A}, (P_i)_{i \in I})$ and $\psi \in \text{ML}$

**Positions:** $(\varphi, v)$ $\quad$ $\varphi$ subformula of $\psi$, $\quad v \in V$

**Verifier moves:**

$$(\varphi \vee \vartheta, v) \to (\varphi, v), \quad (\varphi \vee \vartheta, v) \to (\vartheta, v) \qquad (\langle a \rangle \varphi, v) \longrightarrow (\varphi, w), \quad w \in vE_a$$

**Falsifier moves:**

$$(\varphi \wedge \vartheta, v) \to (\varphi, v), \quad (\varphi \wedge \vartheta, v) \to (\vartheta, v) \qquad ([a]\varphi, v) \longrightarrow (w, \varphi), \quad w \in vE_a$$

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

# Model checking game for ML

Game $\mathcal{G}(\mathcal{K}, \psi)$ for $\mathcal{K} = (V, (E_a)_{a\in A}, (P_i)_{i\in I})$ and $\psi \in$ ML

**Positions:** $(\varphi, v)$ $\varphi$ subformula of $\psi$, $v \in V$

**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:**

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

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

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

$$\text{Verifier wins } \mathcal{G}(\mathcal{K}, \psi) \text{ from position } (\varphi, v) \iff \mathcal{K}, v \models \varphi$$

# Model checking game for ML

**Game** $\mathcal{G}(\mathcal{K}, \psi)$ for $\mathcal{K} = (V, (E_a)_{a \in A}, (P_i)_{i \in I})$ and $\psi \in \mathrm{ML}$

**Positions:** $(\varphi, v)$ $\quad \varphi$ subformula of $\psi$, $\quad v \in V$

**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:**

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

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

Verifier wins $\mathcal{G}(\mathcal{K}, \psi)$ from position $(\varphi, v)$ $\quad \Longleftrightarrow \quad \mathcal{K}, v \models \varphi$

$$\|\mathcal{G}(\mathcal{K}, \psi)\| = O(|\psi| \cdot \|\mathcal{K}\|)$$

# 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

# Model checking for propositional modal logic

**Theorem.** ModelCheck(ML) is PTIME-complete.

- solvable in time $O(|\psi| \cdot \|\mathcal{K}\|)$ via model checking game
- GAME (for strictly alternating games) $\leq_{\log}$ ModelCheck(ML)

$$\mathcal{G} = (V, E),\ v \quad \longmapsto \quad (\mathcal{G}, v), \psi_n \quad (n = |V|)$$

$$\psi_0 := \Box 0 \qquad \psi_{2m+1} = \Diamond \psi_{2m}, \qquad \psi_{2m+2} = \Box \psi_{2m+1}$$

$\mathcal{G}, v \models \psi_m \quad \Longleftrightarrow \quad$ Player 0 wins $\mathcal{G}$ from $v$ in $\leq m$ moves

# 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)

# 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 $\psi$ 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 $\psi$ 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,

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

# 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