Introduction to Theoretical Computer Science

Lecture 10 [bonus]: Games

Dr. Liam O'Connor

University of Edinburgh

Semester 1, 2023/2024

Logical Games

- Consider a game played between two players, **A**belard, written $\forall$ and **E**loise, written $\exists$.

Logical Games

- Consider a game played between two players, **A**belard, written $\forall$ and **E**loise, written $\exists$.
- The game involves alternately choosing elements of a *domain* Ω. As they choose, they produce a sequence of elements $a_0, a_1, a_2, \ldots$

Logical Games

- Consider a game played between two players, **A**belard, written $\forall$ and **E**loise, written $\exists$.
- The game involves alternately choosing elements of a *domain* Ω. As they choose, they produce a sequence of elements $a_0, a_1, a_2, \ldots$
- An infinite sequence of such elements is called a *play*. (w.l.o.g. we generalise finite to infinite sequences)

Logical Games

- Consider a game played between two players, **A**belard, written $\forall$ and **E**loise, written $\exists$.
- The game involves alternately choosing elements of a *domain* Ω. As they choose, they produce a sequence of elements $a_0, a_1, a_2, \ldots$
- An infinite sequence of such elements is called a *play*. (w.l.o.g. we generalise finite to infinite sequences)
- There are disjoint sets $\mathcal{W}_\exists$ and $\mathcal{W}_\forall$, which contain the *winning plays* for $\exists$ and $\forall$ respectively.

Logical Games

- Consider a game played between two players, **A**belard, written $\forall$ and **E**loise, written $\exists$.
- The game involves alternately choosing elements of a *domain* Ω. As they choose, they produce a sequence of elements $a_0, a_1, a_2, \ldots$
- An infinite sequence of such elements is called a *play*. (w.l.o.g. we generalise finite to infinite sequences)
- There are disjoint sets $\mathcal{W}_\exists$ and $\mathcal{W}_\forall$, which contain the *winning plays* for $\exists$ and $\forall$ respectively.
- A logical game is *total* if all plays are in $\mathcal{W}_\exists$ or $\mathcal{W}_\forall$.

Logical Games

- Consider a game played between two players, **A**belard, written $\forall$ and **E**loise, written $\exists$.
- The game involves alternately choosing elements of a *domain* Ω. As they choose, they produce a sequence of elements $a_0, a_1, a_2, \ldots$
- An infinite sequence of such elements is called a *play*. (w.l.o.g. we generalise finite to infinite sequences)
- There are disjoint sets $\mathcal{W}_\exists$ and $\mathcal{W}_\forall$, which contain the *winning plays* for $\exists$ and $\forall$ respectively.
- A logical game is *total* if all plays are in $\mathcal{W}_\exists$ or $\mathcal{W}_\forall$.
- A logical game is *well-founded* if every play is determined to be in $\mathcal{W}_\exists$ or $\mathcal{W}_\forall$ based on a finite prefix.

Logical Games

- Consider a game played between two players, **A**belard, written $\forall$ and **E**loise, written $\exists$.
- The game involves alternately choosing elements of a *domain* Ω. As they choose, they produce a sequence of elements $a_0, a_1, a_2, \ldots$
- An infinite sequence of such elements is called a *play*. (w.l.o.g. we generalise finite to infinite sequences)
- There are disjoint sets $\mathcal{W}_\exists$ and $\mathcal{W}_\forall$, which contain the *winning plays* for $\exists$ and $\forall$ respectively.
- A logical game is *total* if all plays are in $\mathcal{W}_\exists$ or $\mathcal{W}_\forall$.
- A logical game is *well-founded* if every play is determined to be in $\mathcal{W}_\exists$ or $\mathcal{W}_\forall$ based on a finite prefix.
- A logical game is *finite* if there is an n such that all plays are determined to be in $\mathcal{W}_\exists$ or $\mathcal{W}_\forall$ based on a finite prefix of length n.

Winning Strategies

A logical game is *determined* if one or the other players have a *winning strategy*.

Definition

A *winning strategy* is a series of moves for a player p such that, *regardless of the moves of the other player*, the resulting play will be in $\mathcal{W}_p$.

Winning Strategies

A logical game is *determined* if one or the other players have a *winning strategy*.

Definition

A *winning strategy* is a series of moves for a player p such that, *regardless of the moves of the other player*, the resulting play will be in $\mathcal{W}_p$.

Any problem in Σ^0_n can be expressed as finding an $\exists$-winning strategy for a finite game of length n (see previous lecture).

$$\varphi \equiv \exists x. \forall y. \exists z.\ R(x, y, z)$$

Winning Strategies

A logical game is *determined* if one or the other players have a *winning strategy*.

Definition

A *winning strategy* is a series of moves for a player p such that, *regardless of the moves of the other player*, the resulting play will be in $\mathcal{W}_p$.

Any problem in Σ_n^0 can be expressed as finding an $\exists$-winning strategy for a finite game of length n (see previous lecture).

$$\varphi \equiv \exists x.\forall y.\exists z.\ R(x, y, z)$$

$\exists$-winning strategy: we have a proof of φ.

$\forall$-winning strategy: we have a counterexample to φ.

Determined Games

Theorem

Every well-founded game is determined.

- Suppose $\forall$ has no winning strategy for the game. That is, $\forall$ has no winning strategy from the initial position of the game.

Determined Games

Theorem

Every well-founded game is determined.

- Suppose $\forall$ has no winning strategy for the game. That is, $\forall$ has no winning strategy from the initial position of the game.
- If $\forall$ moves, then the next position must also give no winning strategy, or there would have been a winning strategy from the previous position.

Determined Games

Theorem

Every well-founded game is determined.

- Suppose $\forall$ has no winning strategy for the game. That is, $\forall$ has no winning strategy from the initial position of the game.
- If $\forall$ moves, then the next position must also give no winning strategy, or there would have been a winning strategy from the previous position.
- If $\exists$ moves, she must have a move that does not put $\forall$ into a winning strategy, or otherwise the previous position would have a $\forall$-winning strategy.

Determined Games

Theorem

Every well-founded game is determined.

- Suppose $\forall$ has no winning strategy for the game. That is, $\forall$ has no winning strategy from the initial position of the game.
- If $\forall$ moves, then the next position must also give no winning strategy, or there would have been a winning strategy from the previous position.
- If $\exists$ moves, she must have a move that does not put $\forall$ into a winning strategy, or otherwise the previous position would have a $\forall$-winning strategy.
- Thus, inductively, the entire run will never put $\forall$ in a winning position. Thus, $\exists$ has won.

Hintikka Games

Duality

The *dual* of a game G, written $\overline{G}$, is the game where $\forall$ and $\exists$ are transposed in both the rules for playing and for winning.

Hintikka Games

Duality

The *dual* of a game G, written $\overline{G}$, is the game where $\forall$ and $\exists$ are transposed in both the rules for playing and for winning.

We can give a meaning to first-order logic using *Hintikka games*. Define $G[\varphi]$ for all first-order formulae φ:

- $G[\forall x.P]$ = $\forall$ picks an x and the game proceeds as $G[P]$.

Hintikka Games

Duality

The *dual* of a game G, written $\overline{G}$, is the game where $\forall$ and $\exists$ are transposed in both the rules for playing and for winning.

We can give a meaning to first-order logic using *Hintikka games*. Define $G[\varphi]$ for all first-order formulae φ:

- $G[\forall x.P]$ = $\forall$ picks an x and the game proceeds as $G[P]$.
- $G[\exists x.P]$ = $\exists$ picks an x and the game proceeds as $G[P]$.

Hintikka Games

Duality

The *dual* of a game G, written $\overline{G}$, is the game where $\forall$ and $\exists$ are transposed in both the rules for playing and for winning.

We can give a meaning to first-order logic using *Hintikka games*. Define $G[\varphi]$ for all first-order formulae φ:

- $G[\forall x.P]$ = $\forall$ picks an x and the game proceeds as $G[P]$.
- $G[\exists x.P]$ = $\exists$ picks an x and the game proceeds as $G[P]$.
- $G[P \wedge Q]$ = $\forall$ picks if the game proceeds as $G[P]$ or $G[Q]$.

Hintikka Games

Duality

The *dual* of a game G, written $\overline{G}$, is the game where $\forall$ and $\exists$ are transposed in both the rules for playing and for winning.

We can give a meaning to first-order logic using *Hintikka games*. Define $G[\varphi]$ for all first-order formulae φ:

- $G[\forall x.P]$ = $\forall$ picks an x and the game proceeds as $G[P]$.
- $G[\exists x.P]$ = $\exists$ picks an x and the game proceeds as $G[P]$.
- $G[P \wedge Q]$ = $\forall$ picks if the game proceeds as $G[P]$ or $G[Q]$.
- $G[P \vee Q]$ = $\exists$ picks if the game proceeds as $G[P]$ or $G[Q]$.

Hintikka Games

Duality

The *dual* of a game G, written $\overline{G}$, is the game where $\forall$ and $\exists$ are transposed in both the rules for playing and for winning.

We can give a meaning to first-order logic using *Hintikka games*. Define $G[\varphi]$ for all first-order formulae φ:

- $G[\forall x.P]$ = $\forall$ picks an x and the game proceeds as $G[P]$.
- $G[\exists x.P]$ = $\exists$ picks an x and the game proceeds as $G[P]$.
- $G[P \wedge Q]$ = $\forall$ picks if the game proceeds as $G[P]$ or $G[Q]$.
- $G[P \vee Q]$ = $\exists$ picks if the game proceeds as $G[P]$ or $G[Q]$.
- $G[\neg P] = \overline{G[P]}$

Hintikka Games

Duality

The *dual* of a game G, written $\overline{G}$, is the game where $\forall$ and $\exists$ are transposed in both the rules for playing and for winning.

We can give a meaning to first-order logic using *Hintikka games*. Define $G[\varphi]$ for all first-order formulae φ:

- $G[\forall x.P]$ = $\forall$ picks an x and the game proceeds as $G[P]$.
- $G[\exists x.P]$ = $\exists$ picks an x and the game proceeds as $G[P]$.
- $G[P \wedge Q]$ = $\forall$ picks if the game proceeds as $G[P]$ or $G[Q]$.
- $G[P \vee Q]$ = $\exists$ picks if the game proceeds as $G[P]$ or $G[Q]$.
- $G[\neg P] = \overline{G[P]}$
- $\top$ is winning for $\exists$. $\perp$ is winning for $\forall$.

Hintikka Games

Duality

The *dual* of a game G, written $\overline{G}$, is the game where $\forall$ and $\exists$ are transposed in both the rules for playing and for winning.

We can give a meaning to first-order logic using *Hintikka games*. Define $G[\varphi]$ for all first-order formulae φ:

- $G[\forall x.P]$ = $\forall$ picks an x and the game proceeds as $G[P]$.
- $G[\exists x.P]$ = $\exists$ picks an x and the game proceeds as $G[P]$.
- $G[P \wedge Q]$ = $\forall$ picks if the game proceeds as $G[P]$ or $G[Q]$.
- $G[P \vee Q]$ = $\exists$ picks if the game proceeds as $G[P]$ or $G[Q]$.
- $G[\neg P] = \overline{G[P]}$
- $\top$ is winning for $\exists$. $\bot$ is winning for $\forall$.

A formula φ holds iff $\exists$ has a winning strategy for $G[\varphi]$.

Logics for Infinite Games

We can specify infinite (or unbounded) games using *fixed-point logics*. There are a lot of subtleties here that I can talk about later if time.

Logics for Infinite Games

We can specify infinite (or unbounded) games using *fixed-point logics*. There are a lot of subtleties here that I can talk about later if time.

For now, let's just add a *least fixed point* formula construct $[\mathbf{lfp}_{R(\vec{x})}.\varphi]$, with the equivalence:

$$[\mathbf{lfp}_{R(\vec{x})}.\,\varphi](\vec{y}) \quad \equiv \quad \varphi\left[{}^{\vec{y}}/{}_{\vec{x}}\right]\left[{}^{[\mathbf{lfp}_{R(\vec{x})}.\,\varphi](\vec{z})}/{}_{R(\vec{z})}\right]$$

Example (Solitaire Games)

Given a graph consisting of a connectness predicate $E(a, b)$, the cycle-finding game can be stated as:

$$[\mathbf{lfp}_{R(u,v)}.\, E(u, v) \vee (\exists w.\, E(u, w) \wedge R(w, v))]$$

Logics for Infinite Games

We can specify infinite (or unbounded) games using *fixed-point logics*. There are a lot of subtleties here that I can talk about later if time.

For now, let's just add a *least fixed point* formula construct $[\mathbf{lfp}_{R(\vec{x})}.\varphi]$, with the equivalence:

$$[\mathbf{lfp}_{R(\vec{x})}.\,\varphi](\vec{y}) \quad \equiv \quad \varphi\left[{}^{\vec{y}}/_{\vec{x}}\right]\left[{}^{[\mathbf{lfp}_{R(\vec{x})}.\,\varphi](\vec{z})}/_{R(\vec{z})}\right]$$

Example (Solitaire Games)

Given a graph consisting of a connectness predicate $E(a, b)$, the cycle-finding game can be stated as:

$$[\mathbf{lfp}_{R(u,v)}.\, E(u, v) \vee (\exists w.\, E(u, w) \wedge R(w, v))]$$

Why is this a solitaire game?

Back and Forth Games

Back and forth games can be viewed as a game to construct a comparison between two structures A and B.

Back and Forth Games

Back and forth games can be viewed as a game to construct a comparison between two structures A and B.

- The two players are called **S**poiler and **D**uplicator.

Back and Forth Games

Back and forth games can be viewed as a game to construct a comparison between two structures A and B.

- The two players are called **S**poiler and **D**uplicator.
- **S** first picks an element of A.
- **D** picks a “matching” element of B.

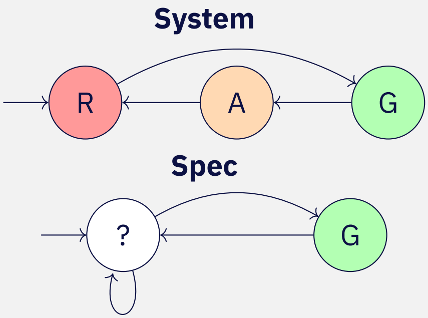
Back and Forth Games

Back and forth games can be viewed as a game to construct a comparison between two structures A and B.

- The two players are called **S**poiler and **D**uplicator.
- **S** first picks an element of A.
- **D** picks a “matching” element of B.
- **S** wins if he picks an element that **D** cannot match.
- **D** wins if she can continue matching **S**’s moves forever.

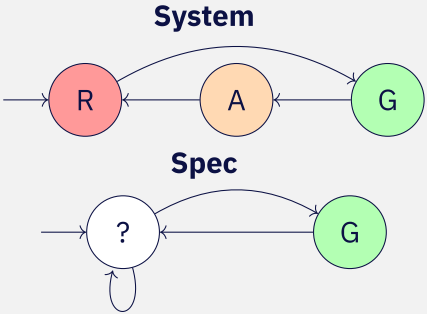
Simulation Games

Consider a traffic light system and its specification:



Simulation Games

Consider a traffic light system and its specification:



Abstraction

Showing that the system meets the spec requires a *simulation relation*: a winning strategy for a back and forth game where **S** picks system moves and **D** picks matching spec moves.

Simulation Relations

Definition

A *simulation* of an automaton C by an automaton A is defined as a relation $\mathcal{S} \subseteq Q_C \times Q_A$ which satisfies:

- If $s \mathrel{\mathcal{S}} t$ then $L_C(s) \cap L_A = L_A(t)$

Simulation Relations

Definition

A *simulation* of an automaton C by an automaton A is defined as a relation $\mathcal{S} \subseteq Q_C \times Q_A$ which satisfies:

- If $s \ \mathcal{S} \ t$ then $L_C(s) \cap L_A = L_A(t)$
- If $s \ \mathcal{S} \ t$ and $s \xrightarrow{a} s'$ (with $a \in \Sigma_C, s' \in Q_C$) then there exists a $t' \in Q_A$ such that $t \xrightarrow{a} t'$ and $s' \ \mathcal{R} \ t'$.

Simulation Relations

Definition

A *simulation* of an automaton C by an automaton A is defined as a relation $\mathcal{S} \subseteq Q_C \times Q_A$ which satisfies:

- If $s \,\mathcal{S}\, t$ then $L_C(s) \cap L_A = L_A(t)$
- If $s \,\mathcal{S}\, t$ and $s \xrightarrow{a} s'$ (with $a \in \Sigma_C, s' \in Q_C$) then there exists a $t' \in Q_A$ such that $t \xrightarrow{a} t'$ and $s' \,\mathcal{R}\, t'$.

The automaton A is an *abstraction* of the concrete automaton C iff a A simulates C. This is sometimes written $A \sqsubseteq C$.

Simulation relations are the foundation of abstraction – a key technique in formal modelling and verification.

Model Equivalence

Question

When do two automata represent the same system?

Model Equivalence

Question

When do two automata represent the same system?
hmm...

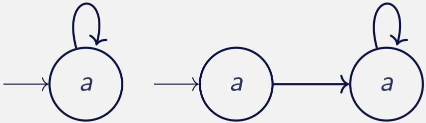
Is it (only) when $A = B$ (*graph isomorphism*)?

Model Equivalence

Question

When do two automata represent the same system?
hmm...

Is it (only) when $A = B$ (*graph isomorphism*)?



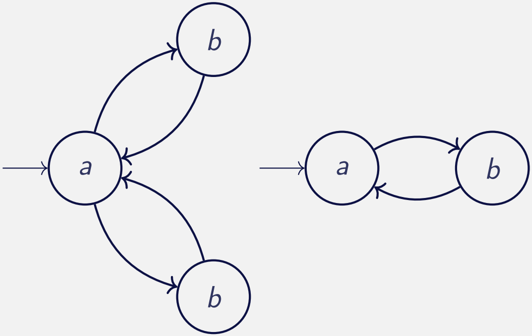
Nope!

Tree Equivalence?

Is it (only) when the two automata have the same computation tree?

Tree Equivalence?

Is it (only) when the two automata have the same computation tree?



Also no!

Bisimulations

Definition

A (strong) *bisimulation* between two automata A and B is defined as a relation $\mathcal{R} \subseteq Q_A \times Q_B$ which satisfies:

- If $s \mathcal{R} t$ then $L_A(s) = L_B(t)$

Bisimulations

Definition

A (strong) *bisimulation* between two automata A and B is defined as a relation $\mathcal{R} \subseteq Q_A \times Q_B$ which satisfies:

- If $s \,\mathcal{R}\, t$ then $L_A(s) = L_B(t)$
- If $s \,\mathcal{R}\, t$ and $s \xrightarrow{a} s'$ (with $a \in \Sigma_A, s' \in Q_A$) then there exists a $t' \in Q_B$ such that $t \xrightarrow{a} t'$ and $s' \,\mathcal{R}\, t'$.

Bisimulations

Definition

A (strong) *bisimulation* between two automata A and B is defined as a relation $\mathcal{R} \subseteq Q_A \times Q_B$ which satisfies:

- If $s \mathcal{R} t$ then $L_A(s) = L_B(t)$
- If $s \mathcal{R} t$ and $s \xrightarrow{a} s'$ (with $a \in \Sigma_A, s' \in Q_A$) then there exists a $t' \in Q_B$ such that $t \xrightarrow{a} t'$ and $s' \mathcal{R} t'$.
- If $s \mathcal{R} t$ and $t \xrightarrow{a} t'$ (with $a \in \Sigma_B, t' \in Q_B$) then there exists a $s' \in Q_A$ such that $s \xrightarrow{a} s'$ and $s' \mathcal{R} t'$.

Bisimulations

Definition

A (strong) *bisimulation* between two automata A and B is defined as a relation $\mathcal{R} \subseteq Q_A \times Q_B$ which satisfies:

- If $s \mathcal{R} t$ then $L_A(s) = L_B(t)$
- If $s \mathcal{R} t$ and $s \xrightarrow{a} s'$ (with $a \in \Sigma_A, s' \in Q_A$) then there exists a $t' \in Q_B$ such that $t \xrightarrow{a} t'$ and $s' \mathcal{R} t'$.
- If $s \mathcal{R} t$ and $t \xrightarrow{a} t'$ (with $a \in \Sigma_B, t' \in Q_B$) then there exists a $s' \in Q_A$ such that $s \xrightarrow{a} s'$ and $s' \mathcal{R} t'$.

Two automata are *bisimulation equivalent* or *bisimilar* iff there exists a bisimulation between their initial states.

Bisimulations

Definition

A (strong) *bisimulation* between two automata A and B is defined as a relation $\mathcal{R} \subseteq Q_A \times Q_B$ which satisfies:

- If $s \mathcal{R} t$ then $L_A(s) = L_B(t)$
- If $s \mathcal{R} t$ and $s \xrightarrow{a} s'$ (with $a \in \Sigma_A, s' \in Q_A$) then there exists a $t' \in Q_B$ such that $t \xrightarrow{a} t'$ and $s' \mathcal{R} t'$.
- If $s \mathcal{R} t$ and $t \xrightarrow{a} t'$ (with $a \in \Sigma_B, t' \in Q_B$) then there exists a $s' \in Q_A$ such that $s \xrightarrow{a} s'$ and $s' \mathcal{R} t'$.

Two automata are *bisimulation equivalent* or *bisimilar* iff there exists a bisimulation between their initial states.

Let's find bisimulations for the previous examples.

Bisimulation Games

We can turn our simulation games into bisimulation games by allowing the locus of control to switch between the two players.

Bisimulation Games

We can turn our simulation games into bisimulation games by allowing the locus of control to switch between the two players.

Bisimulation Games

- **S** goes first and picks a move from either system A or system B.

Bisimulation Games

We can turn our simulation games into bisimulation games by allowing the locus of control to switch between the two players.

Bisimulation Games

- **S** goes first and picks a move from either system A or system B.
- If **S** picked a move from system A, **D** must pick a matching move from system B, and vice versa.

Bisimulation Games

We can turn our simulation games into bisimulation games by allowing the locus of control to switch between the two players.

Bisimulation Games

- **S** goes first and picks a move from either system A or system B.
- If **S** picked a move from system A, **D** must pick a matching move from system B, and vice versa.
- Then, **S** picks another move...

Bisimulation Games

We can turn our simulation games into bisimulation games by allowing the locus of control to switch between the two players.

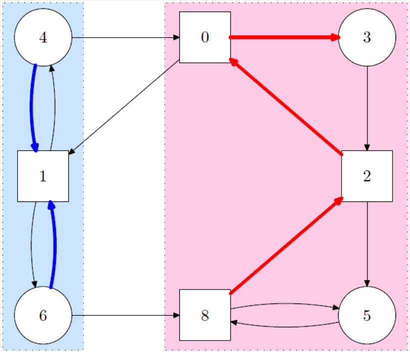
Bisimulation Games

- **S** goes first and picks a move from either system A or system B.
- If **S** picked a move from system A, **D** must pick a matching move from system B, and vice versa.
- Then, **S** picks another move...
- If **S** can find a move that **D** cannot match, **S** wins.
- **D** wins if it can match all moves selected by **S**.

Parity Games

Definition

A parity game is played between two players on a directed graph. Player 0 chooses moves from circular nodes and Player 1 chooses for square nodes. Player n wins an infinite play if the highest number infinitely visited in the play $\equiv n \; [\mathbf{mod}\ 2]$.



Parity Games

- Parity games can be used to give a model-checking algorithm for a type of logic called *modal μ-calculus*, commonly used to express properties of systems.
- Validity and satisfiability for many other modal logics is reducible to parity game solving.
- Parity games are history-free determined.
- Zielonka gives an algorithm for solving parity games.
- **Open question**: Can parity games be solved in polynomial time?