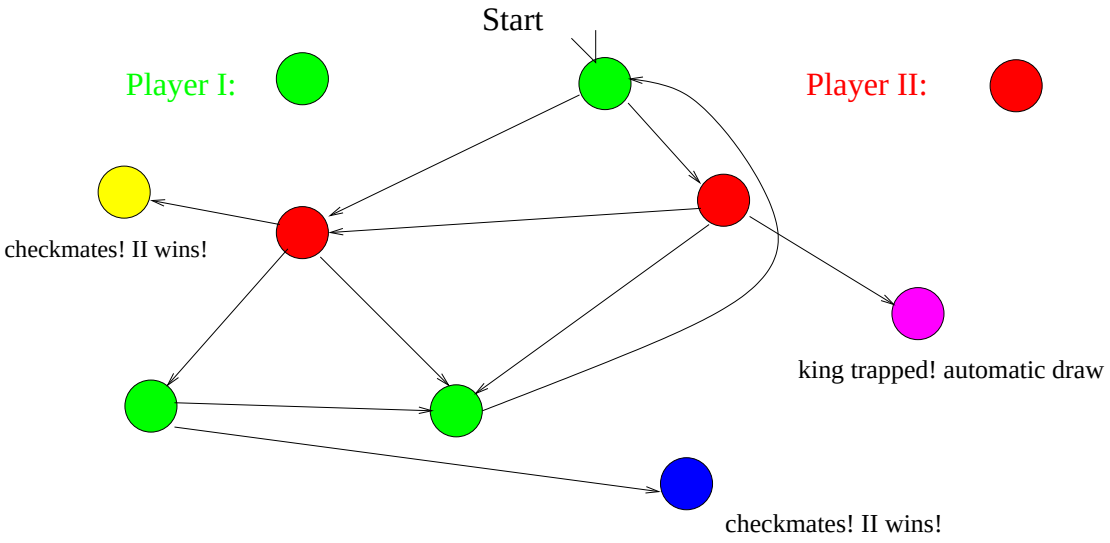
# Algorithmic Game Theory and Applications

# Lecture 13: Games on Graphs

Kousha Etessami

# unbounded chess




- Consider chess: the same exact "position" might recur in the game, but the "game tree" does not reflect this: no relationship between distinct nodes of the tree in a PI-game is indicated.
- There are finitely many positions in the game ($\leq 64^{32}$). After a certain depth, every "play" of the game contains a recurrence of a position.
- Consider "unbounded chess" without any artificial stopping conditions: a game that goes on for ever is by definition a draw.

  Is this win-lose-draw game determined?
  I.e., does Zermelo's theorem still hold?

# more serious motivation

- We can often model the dynamics of a system (e.g., a running program) as a state transition system or state machine.
- If the system interacts with an environment, transitions out of some states can be viewed as "controlled by the environment".
- Can the environment force the system, with any sequence of inputs, into a "error state"?
- Even for state machines without environments, questions like: "*can I reach the reset state from all reachable states of the system?*", can be formulated as a game on a graph.
- Such queries, and much more, can be formalized in certain "temporal logics". Temporal logics (TLs) are formal languages for describing relationships between the occurrence of events over time.

  Efficiently checking such queries against a system model (e.g., a state transition system) is the task of "model checking". Some key model checking tasks are intimately related to efficiently solving certain games on graphs.
  (To learn more, e.g., attend my short TPG course on "Automata-Theoretic Model Checking".)

# game graphs and their trees

A 2-player **game graph**, $G = (V, E, pl)$ consists of:

- A (finite) set $V$ of vertices.
- A set $E \subseteq V \times V$ of edges.
- A function $pl : V \mapsto \{1, 2\}$ mapping each vertex to the player whose turn it is to play at that vertex. Let $V_1 = pl^{-1}(1)$, and $V_2 = pl^{-1}(2)$.

A game graph $G$ together with a start vertex $v_0 \in V$, defines a game tree $T_{v_0}$ given by:

1. Action alphabet $\Sigma = V$. Thus $T_{v_0} \subseteq V^*$.
2. $\epsilon \in T_{v_0}$, and $wv'' \in T_{v_0}$, for $v'' \in V$, if and only if
   - $w = \epsilon$ and $(v_0, v'') \in E$, or
   - $w = w'v'$, for some $v' \in V$, and $(v', v'') \in E$.
   - We extend the player map $pl : V \mapsto \{1, 2\}$ to a map $pl' : T_{v_0} \mapsto \{1, 2\}$ as follows: $pl'(\epsilon) := pl(v_0)$, and $pl'(wv') := pl(v')$.

It is easy to confirm that $T_{v_0}$ is a game tree, where $Act(wv') = \{v'' \mid (v', v'') \in E\}$, and it's plays are precisely all paths in the graph $G$ starting from $v_0$.

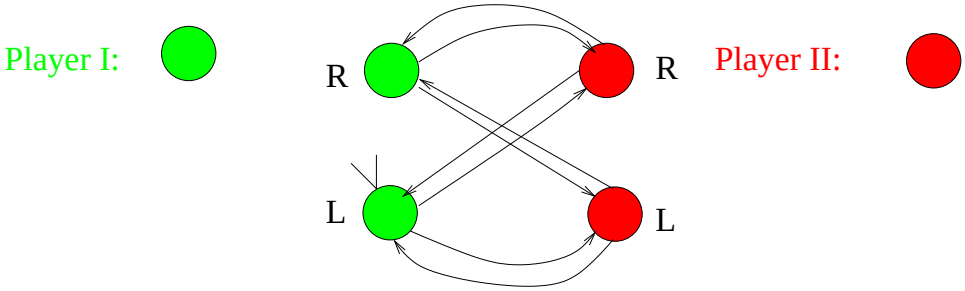
# games on graphs

A **game on a graph**, $\mathcal{G}_{v_0}$, is given by:

A finite game graph $G$,
a vertex $v_0 \in V$,
and a payoff function $u : \Psi_{T_{v_0}} \mapsto \mathbb{R}$,

These together define a 2-player zero-sum PI-game with game tree $T_{v_0}$ and player map $pl'$.

Note: We already know that even for win-lose payoff functions $u$, games on finite graphs are not in general determined, because the infinite binary tree $\{L, R\}^*$ is the game tree for the following game graph:




and we already know (lecture 11) that there are sets $Y$ of plays such that the win-lose game $\langle \{L, R\}^*, Y \rangle$ is not determined.

So, let's restrict the possible payoff functions.

# "history oblivious" payoffs

- Suppose there is some vertex $v'$ of graph $G$ that is a "dead end". E.g., in chess this could be a vertex "checkmate for Player I".
- There may be multiple ways to end up in vertex $v'$. But we would like the winner to be the same for any finite play $wv' \in V^*$. I.e., $u(wv') = u(w'v')$, for any finite plays $wv'$ and $w'v'$.
  So, the payoff is "history oblivious".
- What about for infinite plays $\pi$?
  We can think of $\pi$ as an infinite sequence $v_0v_1v_2v_3v_4v_5\ldots\ldots$, where each $v_i \in V$.
  We use the notation $\pi \in V^\omega$.
- For $\pi = v_0v_1\ldots$, let
  $\inf(\pi) = \{v \in V \mid \text{for } \infty\text{-many } i \in \mathbb{N},\ v_i = v\}$
- Let's call payoff function $u()$ history oblivious[1] (**h.o.**), if for all infinite plays $\pi$ and $\pi'$,
  $$\text{if } \inf(\pi) = \inf(\pi'), \text{ then } u(\pi) = u(\pi')$$
  and for all finite complete plays $wv$ and $w'v$,
  $$u(wv) = u(w'v).$$

Call a graph game h.o. if its payoffs are h.o.
We will only consider h.o. games (and often less).

[1] Technically, we are dispensing with more information here, beyond the info. in finite histories, but never mind the ill-chosen name.

# "finitistic" payoffs

- Note that in chess, if the play $\pi$ is infinite, then the play is always a draw, i.e., $u(\pi) = 0$.

- Let's call an h.o. payoff function finitistic if for all infinite plays $\pi$ and $\pi'$, $u(\pi) = u(\pi')$.
  Let's call a game on a graph $\mathcal{G}_{v_0}$ finitistic if its payoff function is.

  So, in win-lose-draw finitistic games, infinite plays are either all wins, all losses, or all draws, for player 1.

  **Question:** Are all finitistic games on graphs determined?
  **Answer:** Yes.....

  In fact, more it true: for finitistic games there is always a memoryless strategy for each player that achieves the value of the game, and we can efficiently compute these strategies.

# memoryless strategies and determinacy

**Definition** For a game $\mathcal{G}_{v_0}$, a strategy $s_i$ for player $i$ is a **memoryless strategy** if
for all $wv, w'v \in \mathit{Pl}'_i$, $s_i(wv) = s_i(w'v)$, and
if $wv_0 \in \mathit{Pl}'_i$ then $s_i(wv_0) = s_i(\epsilon)$.

I.e., the strategy always makes the same move from a vertex, regardless of the history of how it got there.

Let $\mathit{MLS}_i$ denote the set of memoryless strategies for player $i$. Note $\mathit{MLS}_i$ is a finite set, even if $S_i$ is not. In particular, if $m = |\mathit{Pl}_i|$ is the number of vertices belonging to player $i$, then $|\mathit{MLS}_i| \leq |\Sigma|^m$.

**Definition** $\mathcal{G}_{v_0}$ is **memorylessly determined** if both players have memoryless strategies that achieve "the value" of the game. I.e.,

$$\max_{s_1 \in \mathit{MLS}_1} \inf_{s_2 \in S_2} u(s_1, s_2) = \min_{s_2 \in \mathit{MLS}_2} \sup_{s_1 \in S_1} u(s_1, s_2)$$

**Theorem A** Finitistic games on finite graphs are memorylessly determined. Moreover, there is an efficient (P-time) algorithm to compute memoryless value-achieving strategies in such games.

# the win-lose case: easy "fixed point" algorithm

We first prove the theorem for finitistic win-lose games via an easy "bottom up" fixed point algorithm.

Input: Game graph $G = (V, E, pl, v_0)$.
We assume w.l.o.g. that all infinite plays are a win for player 2 (the other case is symmetric).
A "dead end" is a vertex with no outgoing edge.

$Good := \{v \in V \mid v$ a dead end that wins for player 1$\}$.
$Bad := \{v \in V \mid v$ a dead end that wins for player 2$\}$.

1. Initialize: $Win_1 := Good$; $St_1 := \emptyset$;
2. **Repeat**
   Foreach $v \notin Win_1$:
   If $(pl(v) = 1 \ \ \& \ \ \exists\, (v, v') \in E : \ v' \in Win_1)$
   $Win_1 := Win_1 \cup \{v\}$; $St_1 := St_1 \cup \{v \mapsto v'\}$;
   If $(pl(v) = 2 \ \ \& \ \ \forall\, (v, v') \in E : \ v' \in Win_1)$
   $Win_1 := Win_1 \cup \{v\}$;
   **Until** The set $Win_1$ does not change;

Player 1 has a Win.-Strategy iff $v_0 \in Win_1$. If so, $St_1$ is a memoryless winning strategy for player 1.

# why does this work?

**Proof of Theorem A:** (for the win-lose case)

- First, we claim that for each $v \in Win_1$, $St_1$ is a winning strategy for player 1 in the game $\mathcal{G}_v$ (i.e., the game that starts at node $v$).

  Suppose $v \in Win_1$. It must have entered $Win_1$ after, say, $m$ iterations of the repeat loop. By induction on $m$, if player 1 plays according to (partial) strategy $St_1$, then it is guaranteed a win in the game $\mathcal{G}_v$ within $m$ moves. Note that $St_1$ may be partial: it may only tell us how to move from some vertices. This won't matter.
  Base case: $m = 0$, $v \in Good$.
  Inductively: either $v$ is player 1's vertex or 2's.
  If it is player 1's, then $St_1(v) = v'$, where $(v, v') \in E$ and $v' \in Win_1$, and furthermore $v'$ entered $Win_1$ by $m-1$ iterations. By induction $St_1$ wins for player 1 from $v'$ in $m-1$ moves.
  If $v$ is player 2's, then we know that for all $(v, v') \in E$, $v' \in Win_1$, and furthermore $v'$ entered $Win_1$ by $\leq m-1$ iterations. Thus, no matter what move player 2 makes, in 1 move, by induction, we will be at a vertex $v' \in Win_1$ where player wins with $St_1$ within $m-1$ moves.

- Now consider $v \notin Win_1$ when algorithm halts. For each $v' \in pl^{-1}(2)$, if $\exists\ (v', v'') \in E$, with $v'' \notin Win_1$, then pick one such $v''$, and let $St_2 := St_2 \cup \{v' \mapsto v''\}$. $St_2$ may also be partial. We claim $St_2$ is a memoryless winning strategy for player 2 in every game $\mathcal{G}_v$, where $v \notin Win_1$. Suppose $St_2$ is not a winning strategy for some $v \notin Win_1$. Then player 1 must be able to win by reaching a $Good$ vertex within say, $m$ moves from $v$ against $St_2$. Let's show this is a contradiction.
  Base case: m= 0, but then $v \in Good$. $\Rightarrow\Leftarrow$.
  Inductively: either $v$ is player 1's or player 2's.
  If player 1's, then $\forall (v, v') \in E$, $v' \notin Win_1$, because otherwise by the algorithm $v \in Win_1$. Suppose player 1's winning strategy is to play $(v, v') \in E$. It must have a win within $m-1$ moves from $v' \notin Win_1$ against $St_2$. $\Rightarrow\Leftarrow$.
  If it is player 2's move, then one possibility is $v \in Bad$, $(\Rightarrow\Leftarrow)$. Otherwise, $St_2(v) = v'$ must be defined: since $v \notin Win_1$, there must exist $(v, v') \in E$ with $v' \notin Win_1$. Otherwise, by the algorithm, $v \in Win_1$.
  By induction, player 1 must have a $(m-1)$-winning strategy from $v' \notin Win_1$.$\Rightarrow\Leftarrow$.

# generalizing to finitistic zero-sum

The generalization is not hard:
In a finitistic game, there can only be a bounded number, $r \leq |V| + 1$, of distinct payoffs $u(\pi)$,

$$j_1 < j_2 < j_3 < \ldots < j_r$$

and one of these, say $j_k$, is the payoff $u(\pi)$ for all infinite plays $\pi$. Suppose, w.l.o.g., that $k < r$. (If instead $1 < k$, then we work symmetrically with respect to player 2. If $1 = k = r$, then all payoffs are equal and there is nothing to do.)
Consider a new win-lose game where player 1 wins if it attains payoff $j_r$, and loses if its payoff is any less. Use the fixed point algorithm on this game to find a memoryless (partial) strategy for player 1 that is winning from vertices in $Win_1$ where payoff $j_r$ can be obtained. We can then eliminate $Win_1$ vertices and the payoff $j_r$. We get a new finitistic zero-sum game, with payoffs $j_1 < \ldots < j_{r-1}$. Repeat!! (Homework asks you to solve some of these.)

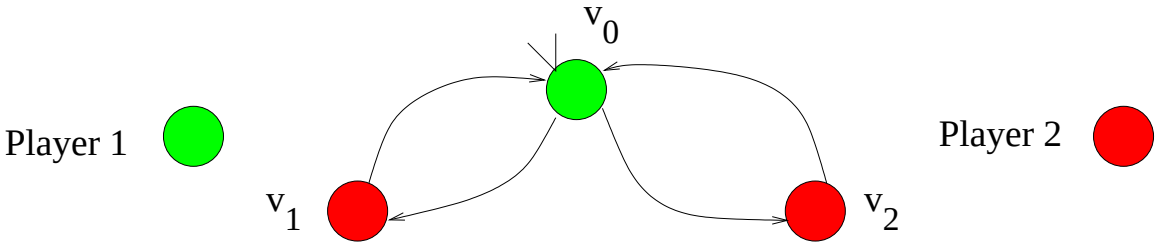
# non-finitistic win-lose h.o. games a.k.a., Muller games

- We will only be interested in win-lose h.o. games.

  By attaching a "self-loop" to every dead-end vertex, every play becomes infinite, and we can define the "payoffs" via a set $\mathcal{F} \subseteq 2^V$, where

  $$\mathcal{F} = \{F \subseteq V \mid \text{player 1 wins if } \inf(\pi) = F\}$$

  We call $\mathcal{F}$ the (Muller) winning condition. Let's call such win-lose h.o. games Muller games.
- **Question:** Are all Muller games determined?
  **Answer:** Yes.....
- **Question:** Are all Muller games memorylessly determined?
  **Answer:** No!
  Consider the following Muller game,
  $\mathcal{F} = \{\{v_0, v_1, v_2\}\}$:




  Does Player 1 have a winning strategy?
  Does it have a memoryless winning strategy?

# remarks

- Muller games and restricted variants of them are important in applications to model checking. We can't do them full justice here.
- Every Muller game can be converted to an "equivalent" (but potentially exponentially larger) game with a limited kind of Muller winning condition called a parity condition. These so called parity games *are* memorylessly determined.
- Can we find winning strategies in parity games efficiently (in P-time)?
  This is a tantalizing open problem.
  It follows from memoryless determinacy that finding winning strategies for them is in **NP** $\cap$ **co-NP**: we can guess a memoryless strategy for either player and efficiently verify that it is a winning strategy.
- Next time you will learn more about parity games.
- A survey text on all this is:
  ["*Automata, Logics, and Infinite Games*",
  *edited by E. Grädel, W. Thomas, T. Wilke, 2002*].

# food for thought: back to LP

Consider the following LP, for solving a finitistic win-lose game with game graph $G$. (Suppose w.l.o.g., player 1 loses if the play is infinite.)
Let $V = \{v_1, \ldots, v_n\}$ be the vertices of $G$. We will have one LP variable $x_i$ for each vertex $v_i \in V$.

**Minimize** $x_m$
**Subject to:**
$0 \leq x_i \leq 1$, for $i = 1, \ldots, n$;
$x_i = 1$, for $v_i$ a winning dead end for player 1.
$x_i = 0$, for $v_i$ a losing dead end for player 1.
For each $x_i$ where $pl(v_i) = 1$,
$x_i \geq x_j$, for each $(v_i, v_j) \in E$.
For each $x_i$ where $pl(v_i) = 2$,
and $\{v_{j_1}, \ldots, v_{j_r}\} = \{v' \mid (v_i, v') \in E\}$,
$x_i \leq x_{j_k}$, for $k = 1, \ldots, r$ , and
$x_i \geq x_{j_1} + \ldots + x_{j_r} - (r - 1)$

–Convince yourself: the optimal value of this LP is 1 iff player 1 has a winning strategy in $\mathcal{G}_{v_m}$. (Incidental corollary: the LP problem is "P-hard".)
–Now, what if instead of 2 players, player 1 was playing "alone against nature"? Could you formulate an LP for 1's optimal payoff? This would be a simple instance of a "Markov Decision Process".