Principles of Computer Game Design and Implementation

Lecture 20

Outline for today

- Sense-Think-Act Cycle:
  - Thinking
  - Acting

Agents and Virtual Player

- Agents, no virtual player
  - Shooters, racing, …
- Virtual player, no agents
  - Chess, …
- Both
  - Strategy games, team sport games, …

Agents

- Act as
  - enemies, allies, neutral characters
- Constantly go through a
  - *Sense – Think – Act* cycle
    - Sometimes can learn new behaviours
- Example: first-person shooter enemies, other car drivers, units in strategies

Sense-Think-Act Cycle: Thinking

- Sensed information gathered
- Must process sensed information
- Two primary methods
  - Process using pre-coded expert knowledge
  - Use search to find an optimal solution

Thinking: Expert Knowledge

- Many different systems
  - Finite-state machines
  - Production systems
  - Decision trees
  - Logical inference
- Encoding expert knowledge is appealing because it’s relatively easy
  - Can ask just the right questions
  - As simple as if-then statements
- Problems with expert knowledge
  - Not very scalable

Thinking: Search

- Employs search algorithm to find an optimal or near-optimal solution
- E.g.
  - A* pathfinding
  - Game search

Thinking: Machine Learning

- If imparting expert knowledge and search are both not reasonable/possible, then machine learning might work
- Examples:
  - Reinforcement learning
  - Neural networks
  - Decision tree learning
- Not often used by game developers
  - complexity of learning techniques
  - reproducibility and quality control
  - impossible to test if it performs correctly and locate bugs.

Thinking: Flip-Flopping Decisions

- Must prevent flip-flopping of decisions
- Reaction times might help keep it from happening every frame
- Must make a decision and stick with it
  - Until situation changes enough
  - Until enough time has passed

Sense-Think-Act Cycle: Acting

- Sensing and thinking steps invisible to player
- Acting is how player witnesses intelligence
- Numerous agent actions, for example:
  - Change locations
  - Pick up object
  - Play animation
  - Play sound effect
  - Converse with player
  - Fire weapon

Acting: Showing Intelligence

- Adeptness and subtlety of actions impact perceived level of intelligence
- Enormous burden on asset generation
- Agent can only express intelligence in terms of vocabulary of actions
- Current games have huge sets of animations/assets
  - Must use scalable solutions to make selections

Extra Step in Cycle: Learning and Remembering

- Optional 4th step
- Not necessary in many games
  - Agents don't live long enough
  - Game design might not desire it

Learning

- Remembering outcomes and generalizing to future situations
- Simplest approach: gather statistics
  - If 80% of time player attacks from left
  - Then expect this likely event
- Adapts to player behavior

Remembering

- Remember hard facts
  - Observed states, objects, or players
- For example
  - Where was the player last seen?
  - What weapon did the player have?
  - Where did I last see a health pack?
- Memories should fade
  - Helps keep memory requirements lower
  - Simulates poor, imprecise, selective human memory

Remembering within the World

- All memory doesn't need to be stored in the agent – can be stored in the world
- For example:
  - Agents get slaughtered in a certain area
  - Area might begin to "smell of death"
    - Agent's path planning will avoid the area
  - Simulates group memory

Virtual Player Example: Game Playing

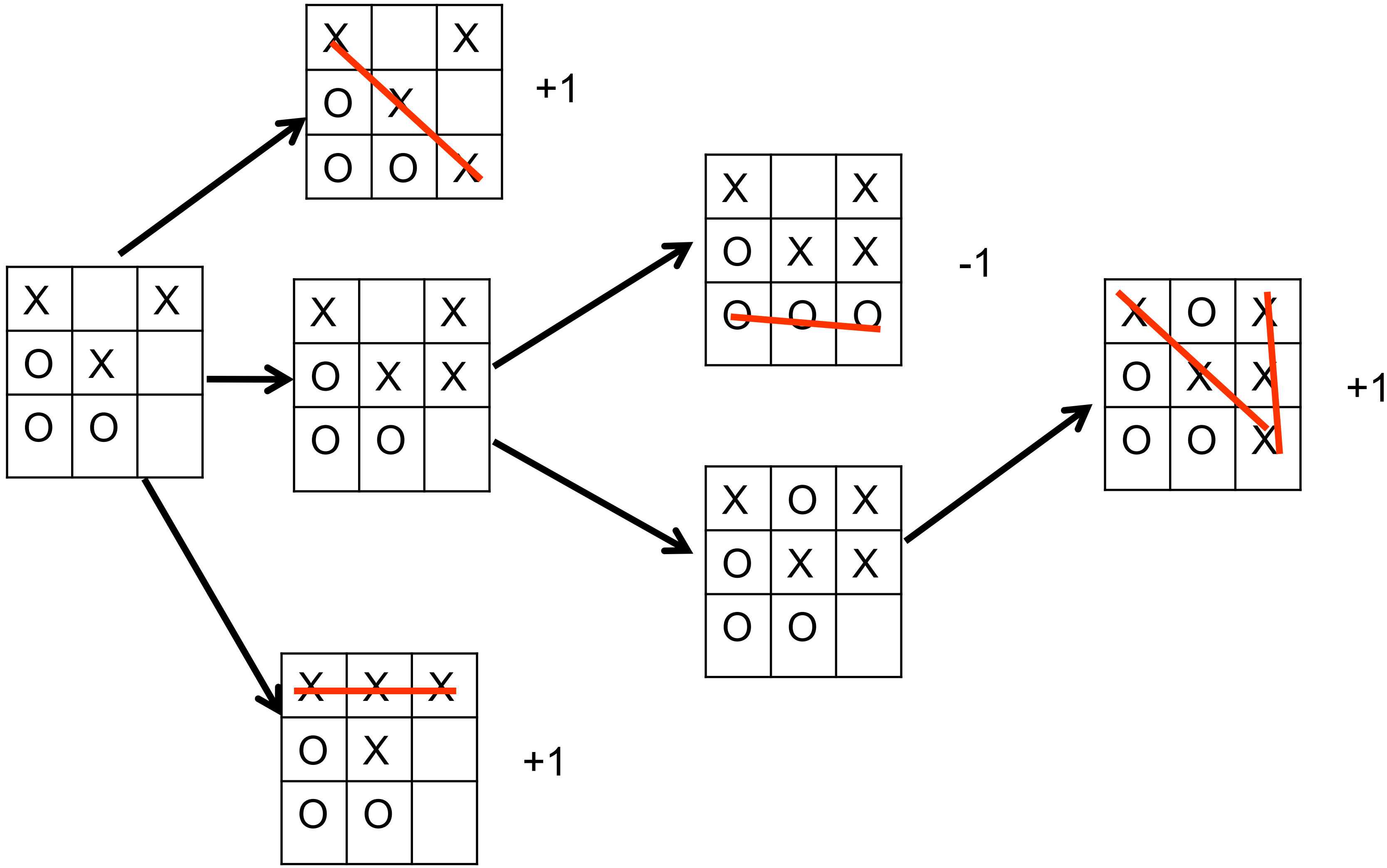
- Recall from COMP219:

	deterministic	chance
perfect information	chess, checkers, go, othello	backgammon monopoly
imperfect information	battleships, blind tictactoe	bridge, poker, scrabble nuclear war

Problem Formulation

- Initial state
  - Initial board position, player to move.
- Successor function
  - Returns list of (move, state) pairs, one per legal move.
- Terminal test
  - Determines when the game is over.
- Utility function
  - Numeric value for terminal states
  - E.g. Chess +1, -1, 0
  - E.g. Backgammon +192 to -192

Game Tree



Game Tree

- Each level labelled with player to move
- Each level represents a ***ply***
  - Half a turn
- Represents what happens with competing agents

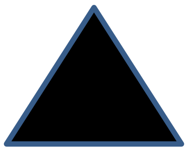
Minimax Value

Formally:

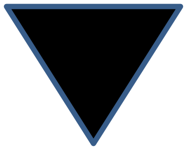
$$\text{MinimaxValue}(n) = \begin{cases} \text{Utility}(n) & \text{Terminal} \\ \max_{s \in \text{Successors}(n)} \text{MinimaxValue}(s) & \text{MAX} \\ \min_{s \in \text{Successors}(n)} \text{MinimaxValue}(s) & \text{MIN} \end{cases}$$

Minimax Algorithm

- Calculate minimax value of each node recursively
- Depth-first exploration of tree
- Game tree as *minimax tree*
- *Max Node:*



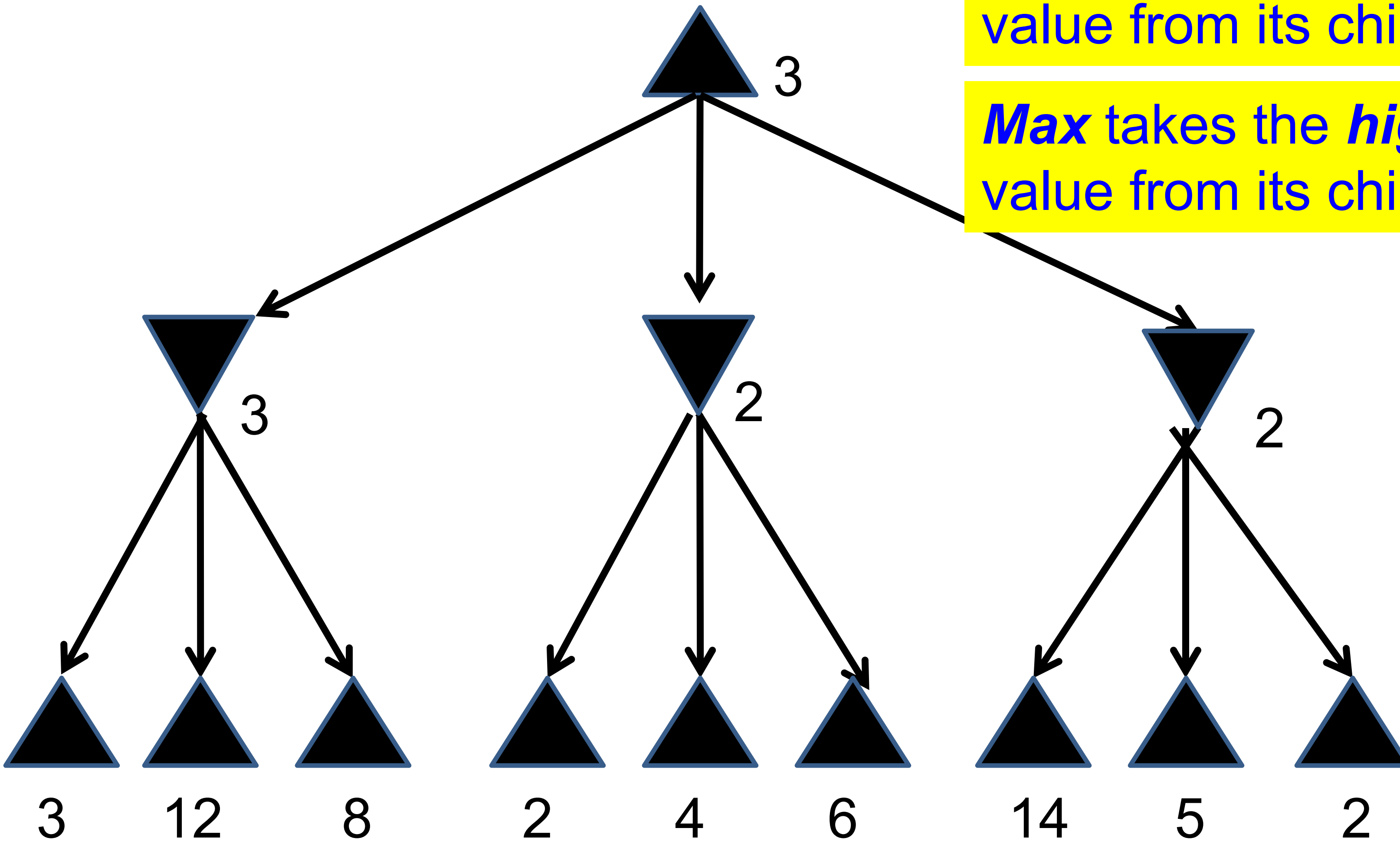
- *Min Node*



Minimax Tree

Min **takes the** ***lowest*** value from its children

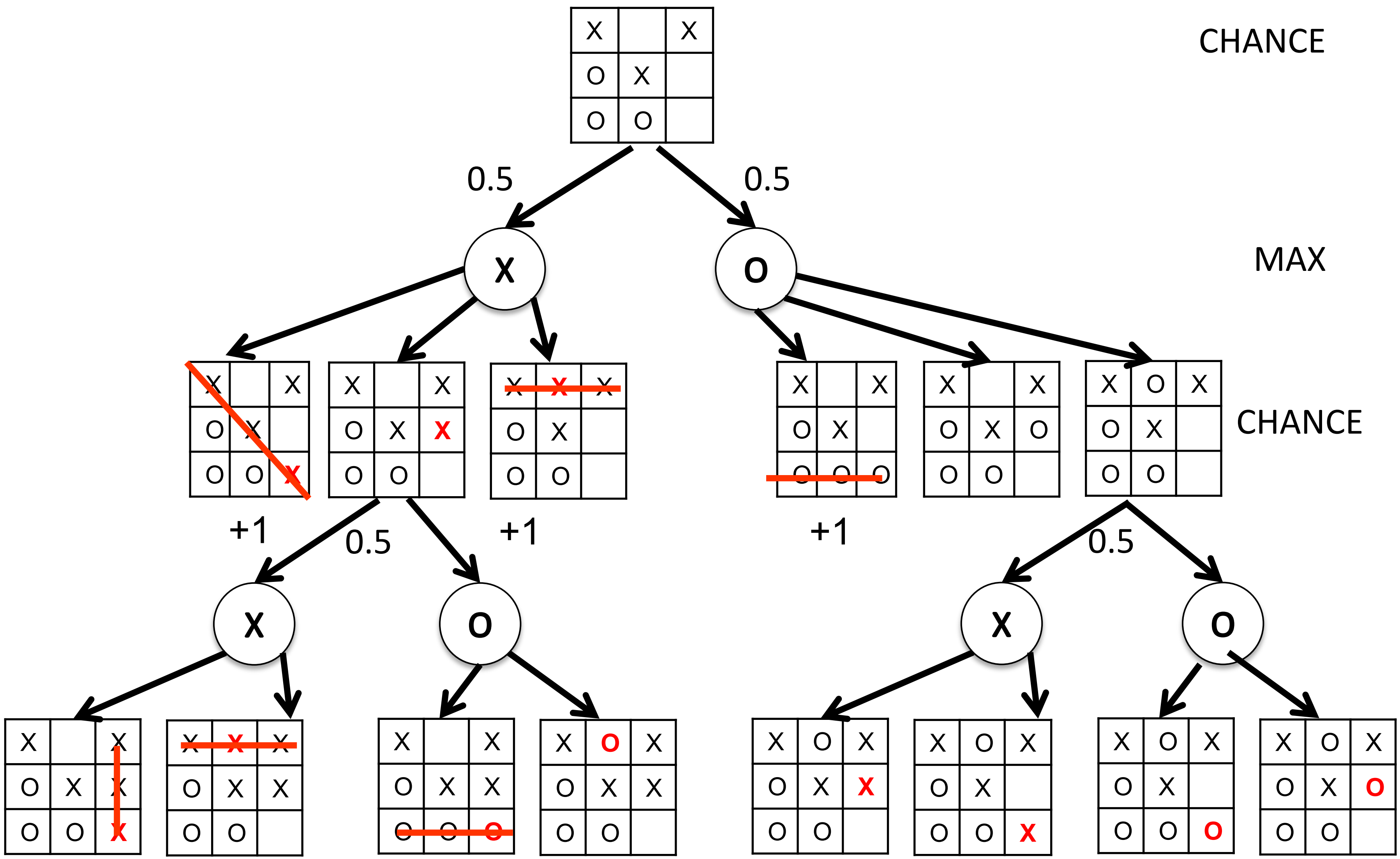
Max takes the ***highest*** value from its children



Extension: Nondeterministic Games

- Consider Naughts and Crosses game with an element of chance:
- **Before** each move, a player tosses a coin
  - Head: you play crosses
  - Tail: you play naughts

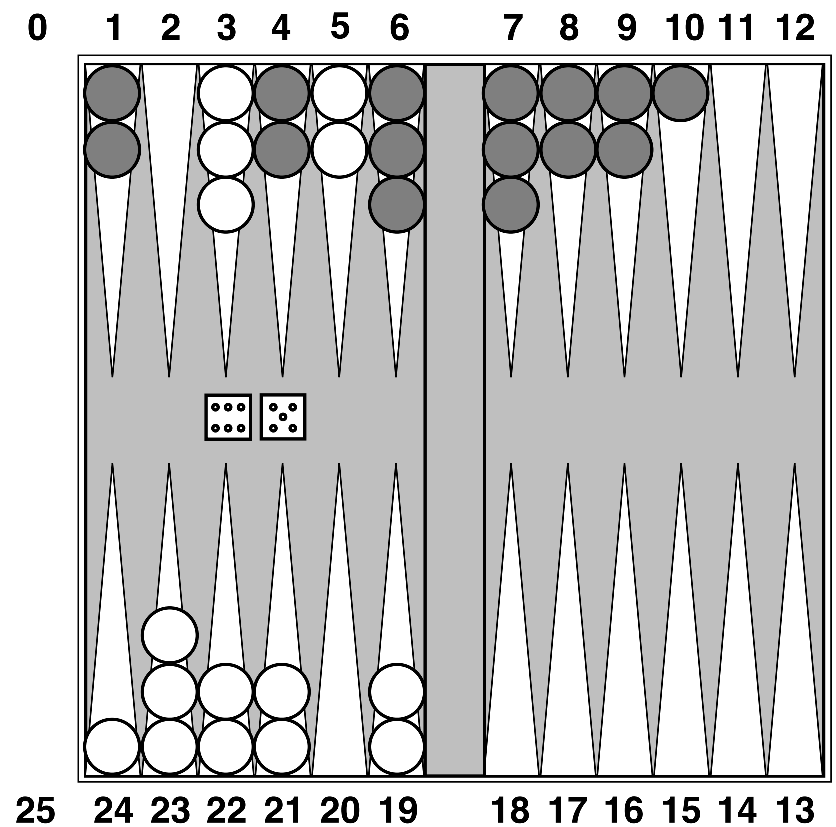
Game Tree With Chance Nodes



Backgammon

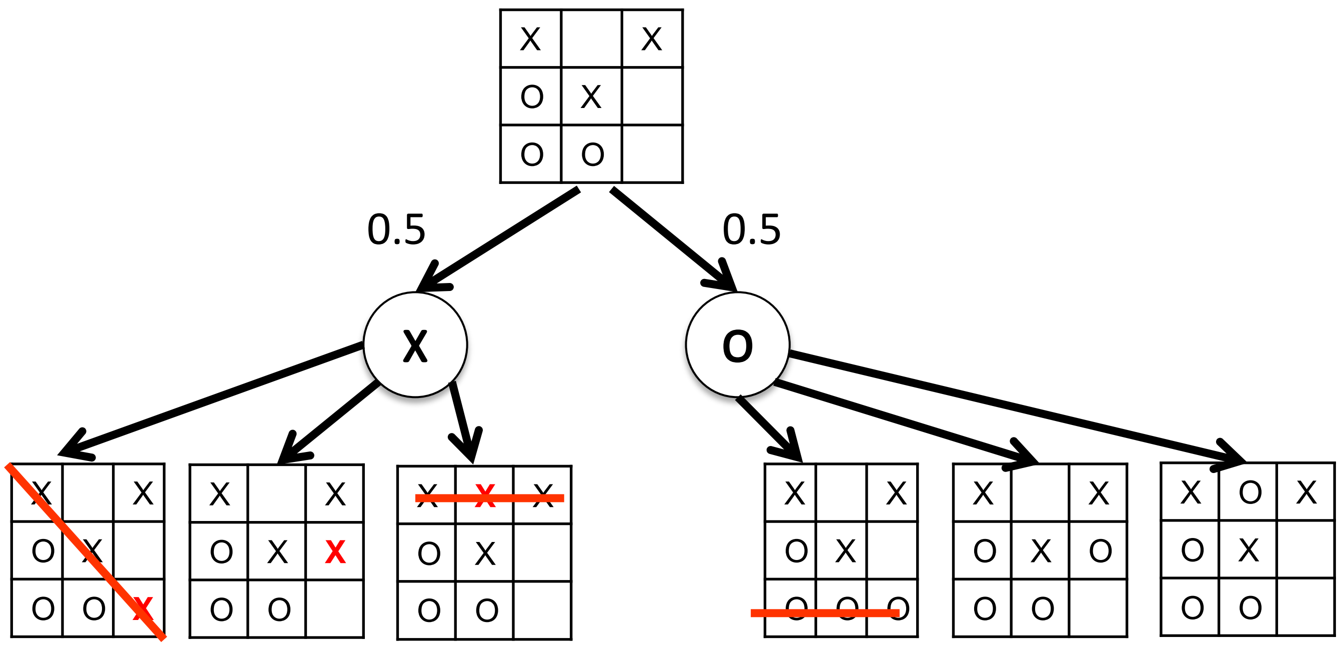
- Admittedly, this Naughts and Crosses modification is weird

- Backgammon is a better example of a chance game



ExpectiMinimax

$$\mathrm{EMV}(n) = \begin{cases} \mathrm{Utility}(n) & \text{Terminal} \\ \max_{s \in \mathrm{Successors}(n)} \mathrm{EMV}(s) & \text{MAX} \\ \min_{s \in \mathrm{Successors}(n)} \mathrm{EMV}(s) & \text{MIN} \\ \sum_{s \in \mathrm{Successors}(n)} \mathsf{Prob}(s)\mathrm{EMV}(s) & \text{CHOICE} \end{cases}$$



Playing Cards

- Chance + Imperfect information

- Idea: Chance nodes for all possible deals
  - (compatible with the revealed information)

- Use ExpectiMinimax

Summary

- Game artificial intelligence differs in that it sets a different goal
  - Appear intelligent rather than be one
- Game agent & Virtual player
  - Virtual player is closer to traditional AI
  - Game agents correspond to the modern view on AI
- Next, we look more on agents than on the VP.