CS440/ECE448 Lecture 34: Games of Chance and Imperfect Information

Mark Hasegawa-Johnson, 4/2020

Including slides by Svetlana Lazebnik

A contemporary backgammon set. Public domain photo by Manuel Hegner, 2013, https://commons.wikimedia.org/w/index.php?curid=25006945

A game of Texas Hold'em in progress. Copyright US Navy, released for public distribution 2009, https://commons.wikimedia.org/w/index.php?curid=8361356

Types of game environments

	Deterministic	**Stochastic**
Perfect information (fully observable)	Chess, checkers, go	Backgammon, monopoly
Imperfect information (partially observable)	Battleship	Scrabble, poker, bridge

Content of today's lecture

- Stochastic games: the Expectiminimax algorithm
- Imperfect information: belief states

Stochastic games

How can we incorporate dice throwing into the game tree?

Minimax

State evolves deterministically (when a player acts, that action uniquely determines the following state).

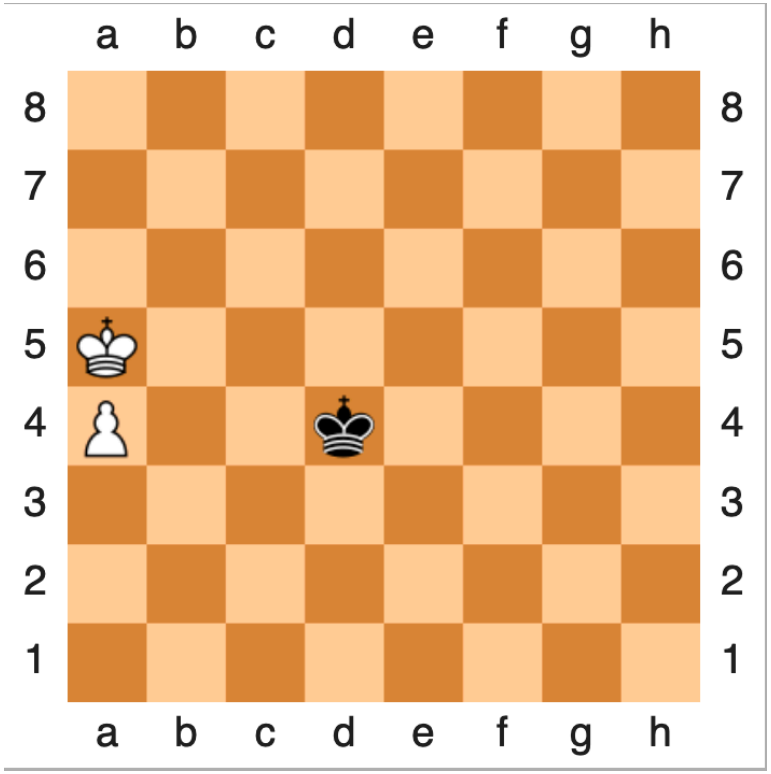
Current state is visible to both players.

Each player tries to maximize his or her own reward:

- **Maximize** (over all possible moves I can make) the
- **Minimum** (over all possible moves Min can make) of the resulting utility:

$$U(s) = \max_{s' \in C(s)} U(s')$$

$$U(s') = \min_{s'' \in C(s')} U(s'')$$



Bellman's Equation

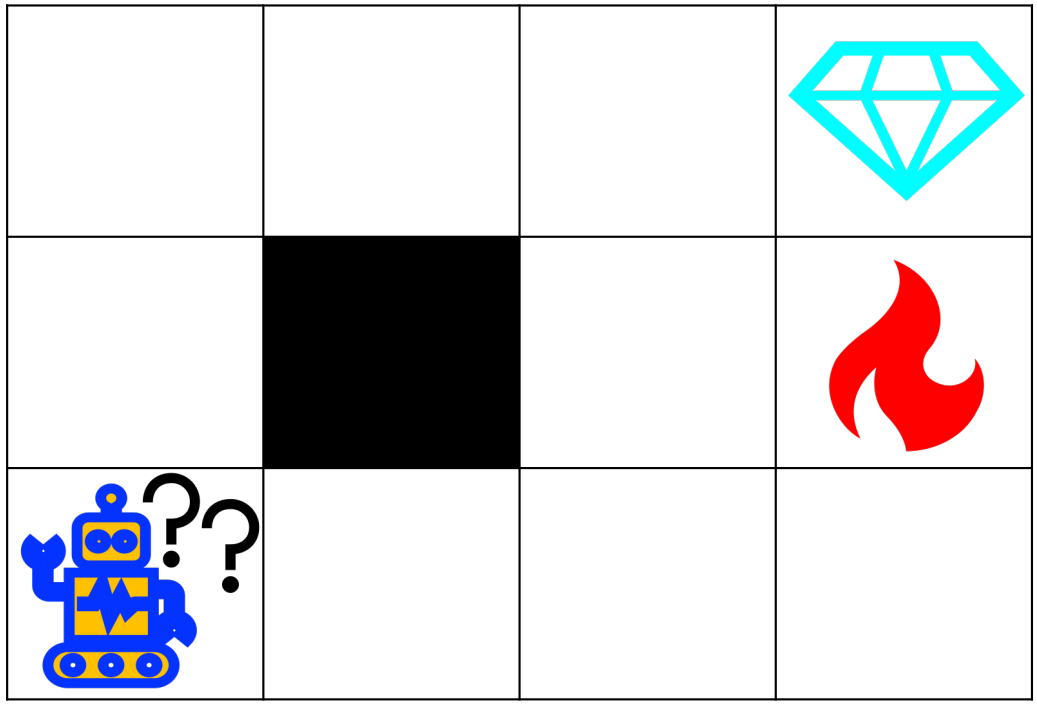
State evolves **stochastically** (when a player acts, that action influences the state transition probability).

Current state is visible to the player.

The player tries to maximize his or her own reward:

- **Maximize** (over all possible moves I can make) the
- **Expected value** (over all possible successor states) of the resulting utility:

$$U(s) = R(s) + \gamma \max_a \sum_{s'} P(s'|s,a)U(s')$$



Expectiminimax

State evolves **stochastically** (when a player acts, that action influences the state transition probability).

Current state is visible to both players.

Each player tries to maximize his or her own reward:

- **Maximize** (over all possible moves I can make) the
- **Minimum** (over all possible moves Min can make) of the
- **Expected value** (over all possible successor states) of the resulting utility:

$$U(s) = \max_{a} \sum_{s'} P(s'|s,a)U(s')$$

$$U(s') = \min_{a'} \sum_{s''} P(s''|s',a')U(s'')$$

Expectiminimax: notation

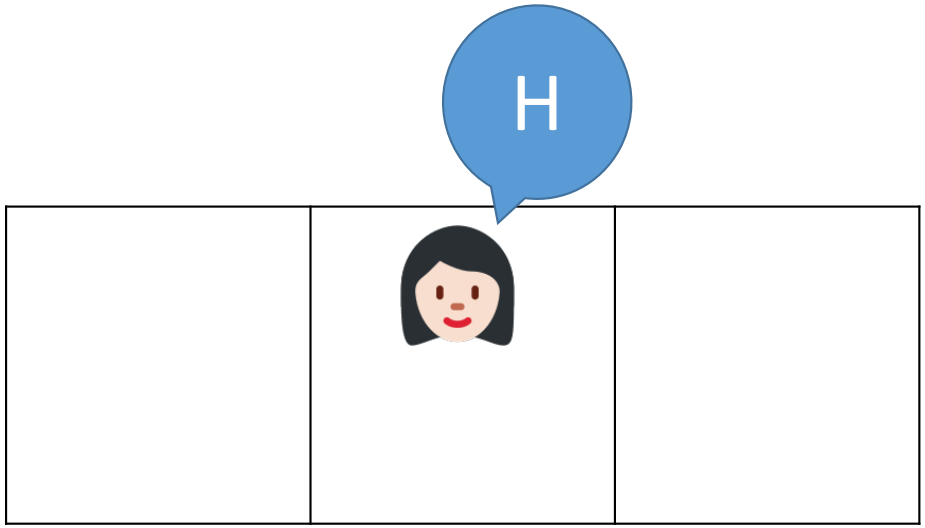
= MAX node. $U(s) = \max_{a \in A(s)} Q(s,a)$

= MIN node. $U(s) = \min_{a \in A(s)} Q(s,a)$

= Chance node. $Q(s,a) = \sum_{s'} P(s'|s,a)U(s')$

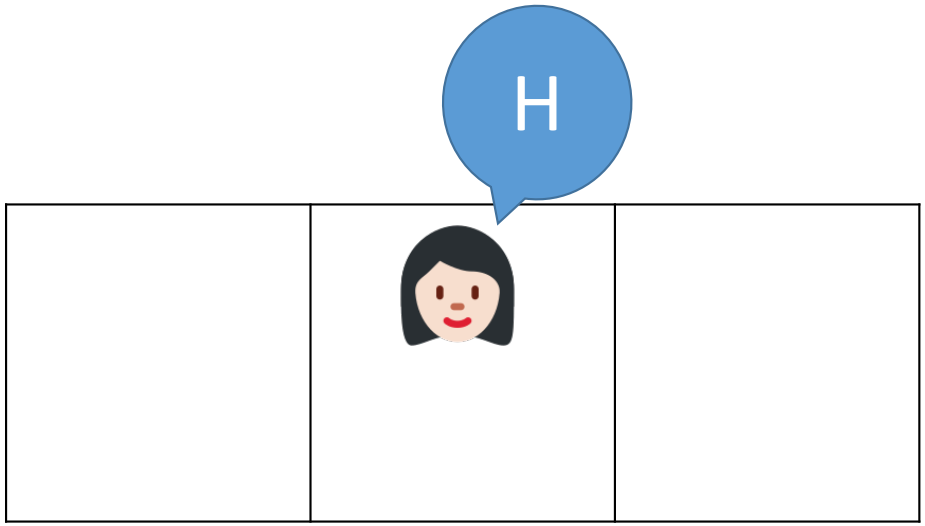
Expectiminimax example

- MIN: Min decides whether to count heads (action H) or tails (action T) as a forward movement.



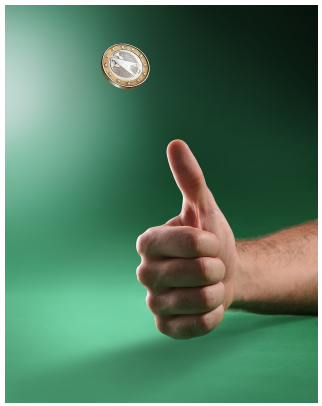
Expectiminimax example

- MIN: Min decides whether to count heads (action H) or tails (action T) as a forward movement.

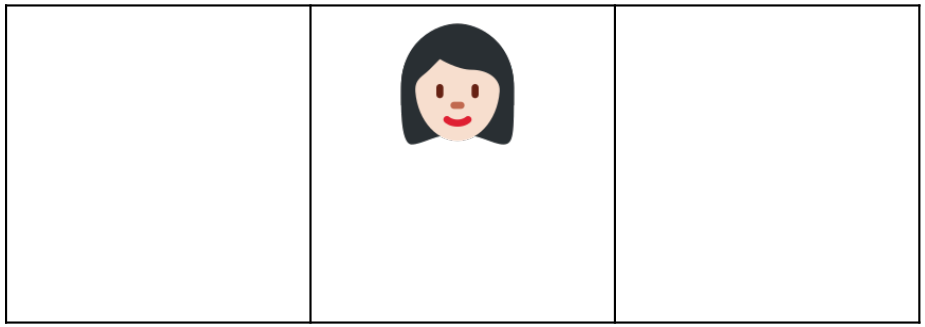


Expectiminimax example

- MIN: Min decides whether to count heads (action H) or tails (action T) as a forward movement.
- Chance: she flips a coin and moves her game piece in the direction indicated.



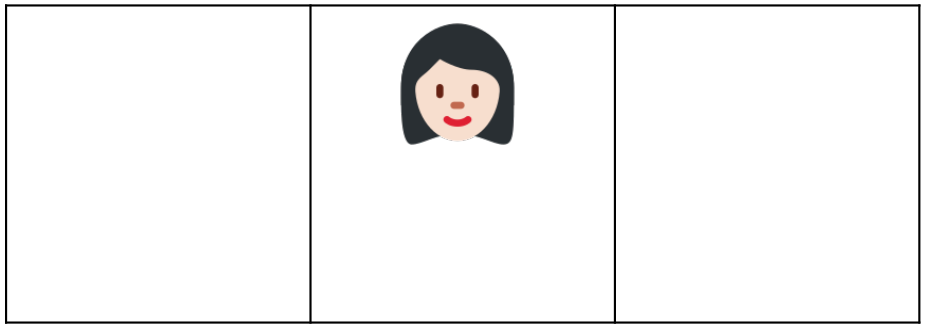
By ICMA Photos - Coin Toss, CC BY-SA 2.0, https://commons.wikimedia.org/w/index.php?curid=71147286



Emojis by Twitter, CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=59974366

Expectiminimax example

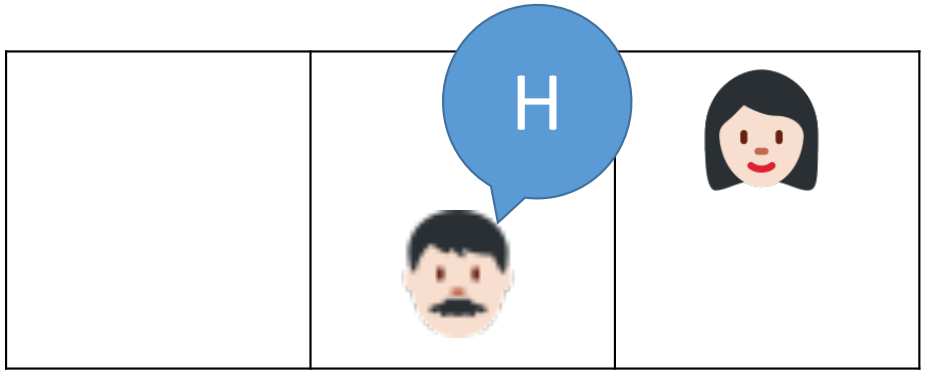
- MIN: Min decides whether to count heads (action H) or tails (action T) as a forward movement.
- Chance: she flips a coin and moves her game piece in the direction indicated.



Expectiminimax example

- MIN: Min decides whether to count heads (action H) or tails (action T) as a forward movement.
- Chance: she flips a coin and moves her game piece in the direction indicated.
- MAX: Max decides whether to count heads (action H) or tails (action T) as a forward movement.

By NJR ZA - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=4228918



Emojis by Twitter, CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=59974366

Expectiminimax example

- MIN: Min decides whether to count heads (action H) or tails (action T) as a forward movement.
- Chance: she flips a coin and moves her game piece in the direction indicated.
- MAX: Max decides whether to count heads (action H) or tails (action T) as a forward movement.
- Chance: he flips a coin and moves his game piece in the direction indicated.

By NJR ZA - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=4228918

By NJR ZA - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=4228918

Emojis by Twitter, CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=59974366

Expectiminimax example

- MIN: Min decides whether to count heads (action H) or tails (action T) as a forward movement.
- Chance: she flips a coin and moves her game piece in the direction indicated.
- MAX: Max decides whether to count heads (action H) or tails (action T) as a forward movement.
- Chance: he flips a coin and moves his game piece in the direction indicated.

Reward: $2 to the winner, $0 for a draw.

By NJR ZA - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=4228918

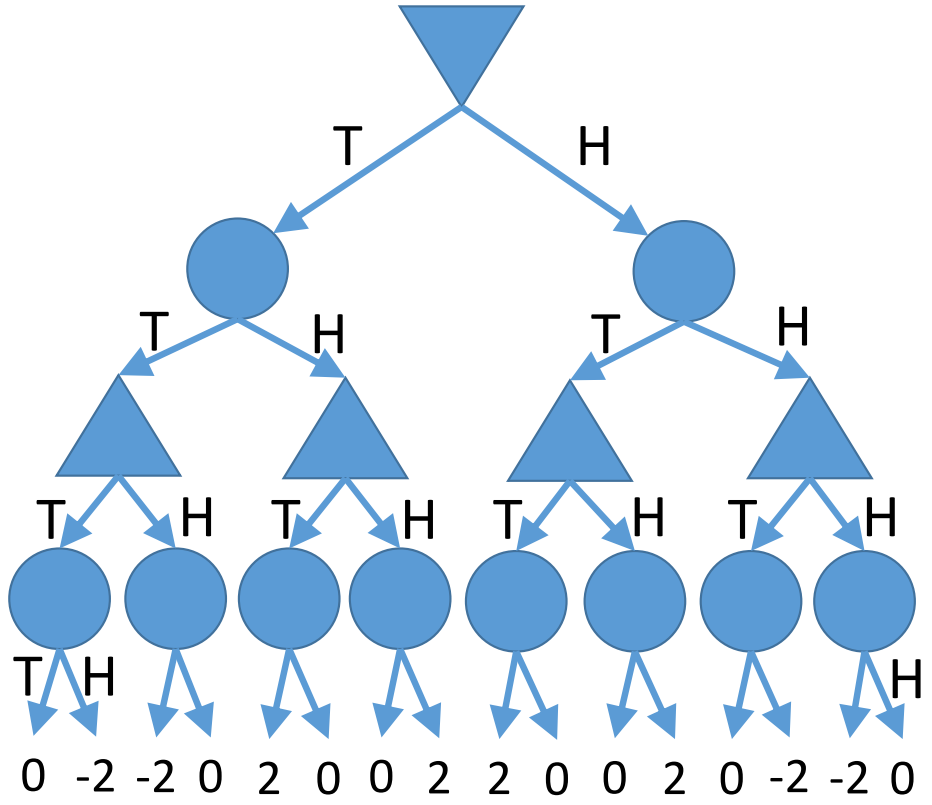
By NJR ZA - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=4228918

Emojis by Twitter, CC BY 4.0, https://commons.wikimedia.org/w/index.php?curid=59974366.
$2 By Bureau of Engraving and Printing: U.S. Department of the Treasury - own scanned, Public Domain, https://commons.wikimedia.org/w/index.php?curid=56299470

Expectiminimax example

- MIN: Min decides whether to count heads (action H) or tails (action T) as a forward movement.
- Chance: she flips a coin and moves her game piece in the direction indicated.
- MAX: Max decides whether to count heads (action H) or tails (action T) as a forward movement.
- Chance: he flips a coin and moves his game piece in the direction indicated.

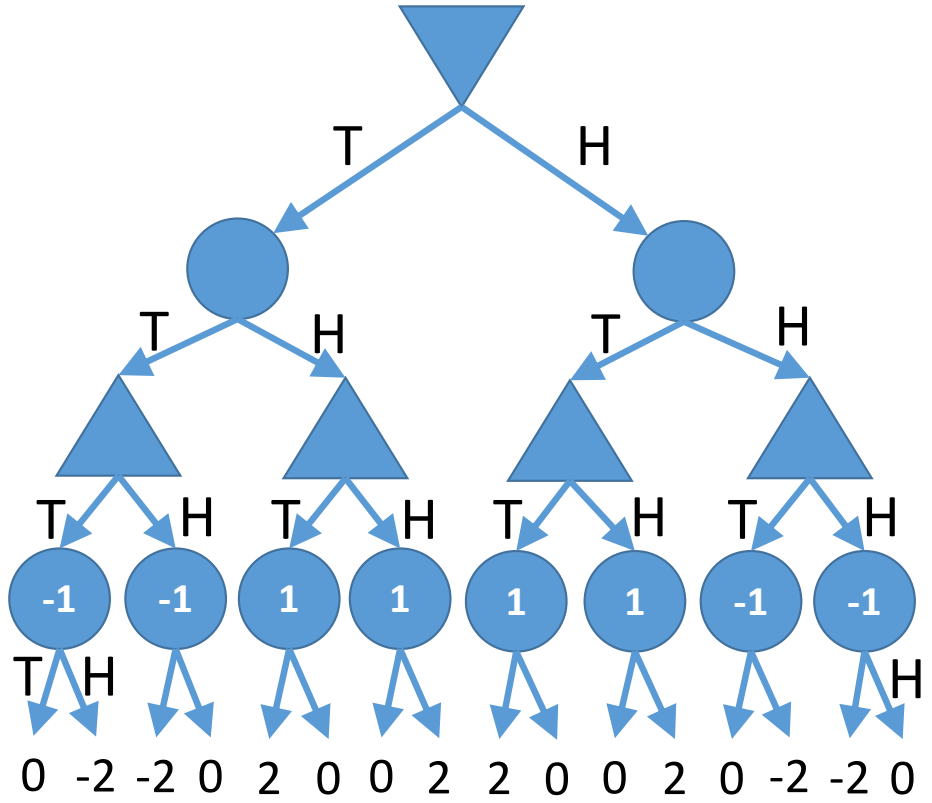
Reward: $2 to the winner, $0 for a draw.



Expectiminimax example

Chance node:

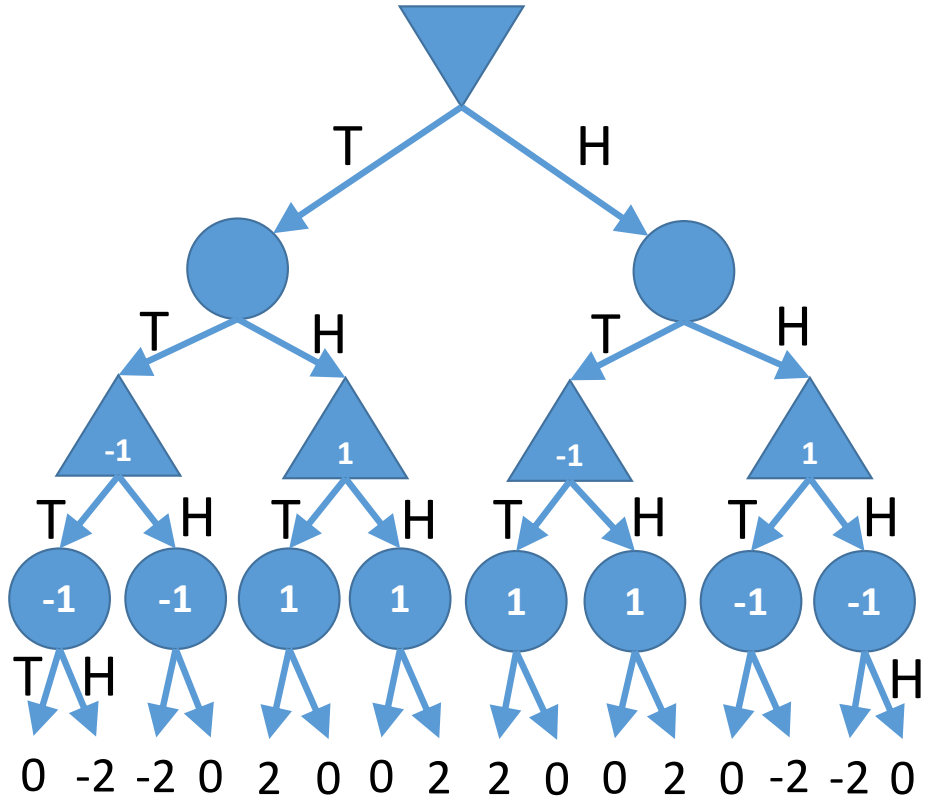
$$Q(s,a) = \sum_{s'} P(s'|s,a)U(s')$$



Expectiminimax example

Max node:

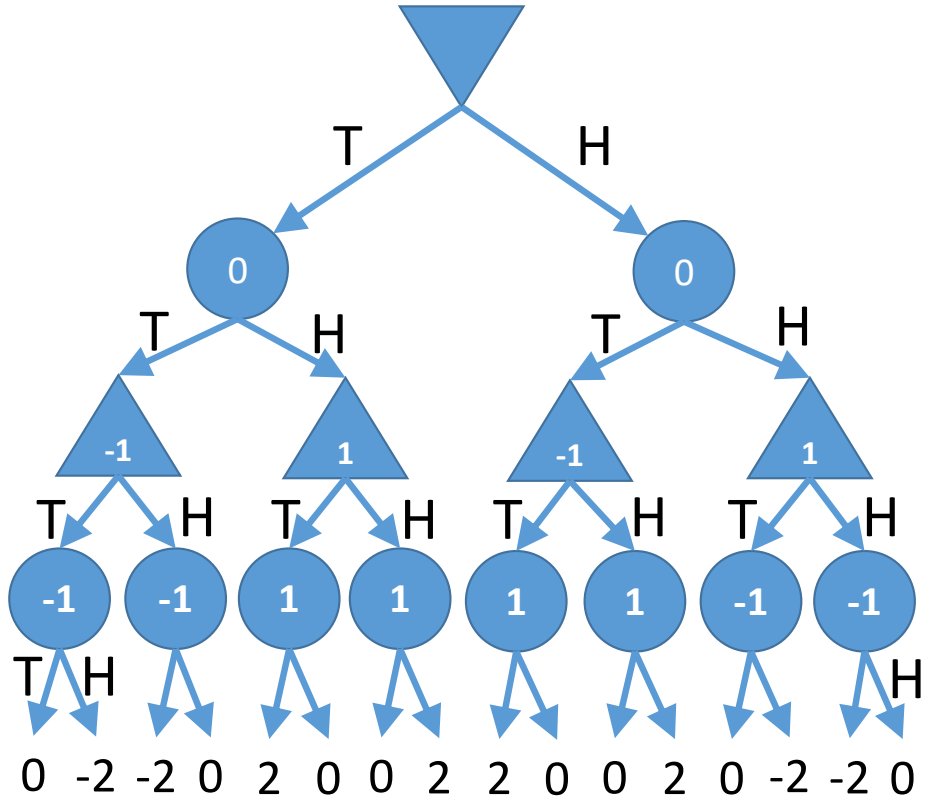
$$U(s) = \max_{a \in A(s)} Q(s, a)$$



Expectiminimax example

Chance node:

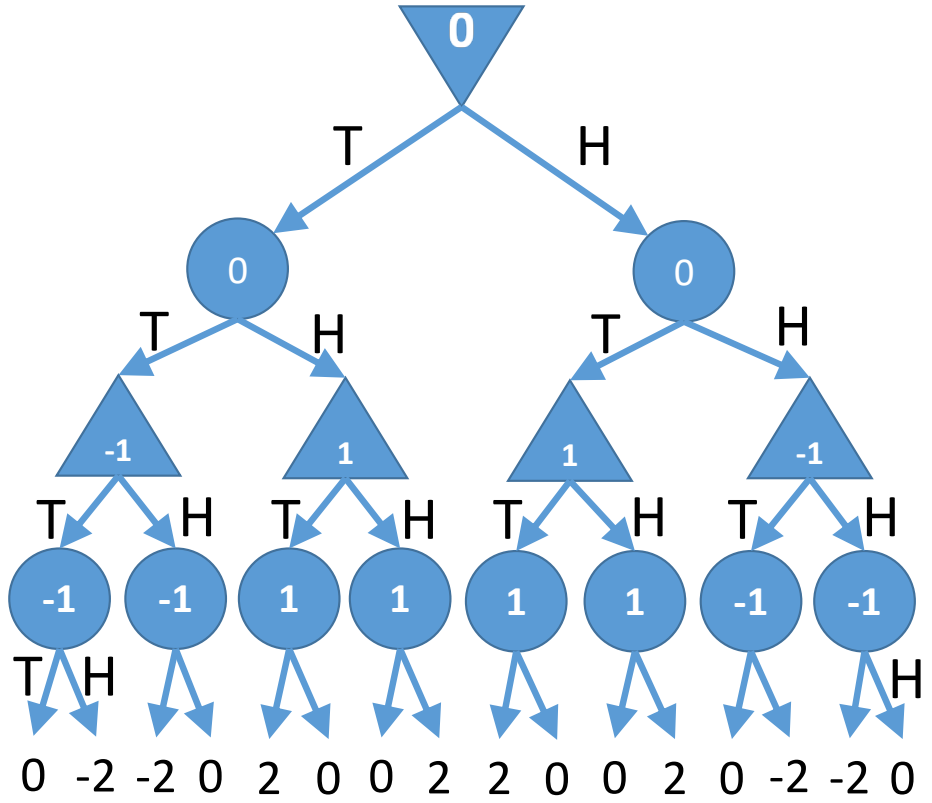
$$Q(s,a) = \sum_{s'} P(s'|s,a)U(s')$$



Expectiminimax example

Min node:

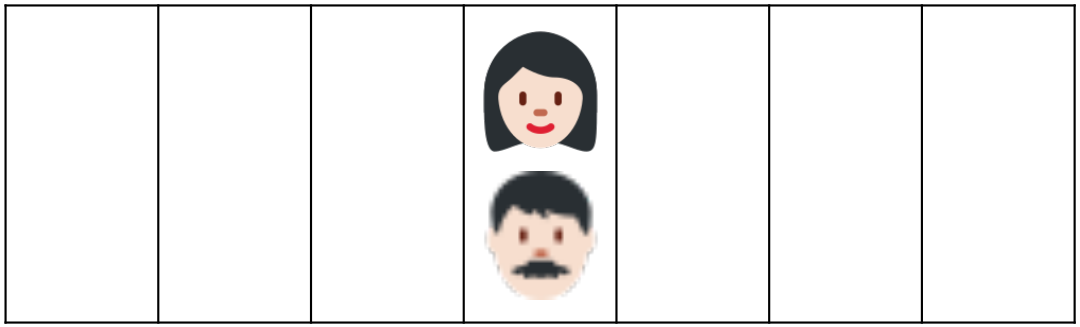
$$U(s) = \min_{a \in A(s)} Q(s, a)$$



Expectiminimax example #2

- MIN: Min decides whether she's going to move $D - 3$ or $3 - D$ steps forward, where D is the roll of the dice.
- Chance: she rolls the dice and moves her game piece in the direction indicated.
- MAX: Max decides whether he's going to move $D - 3$ or $3 - D$ steps forward, where D is the roll of the dice.
- Chance: he rolls the dice and moves his game piece in the direction indicated.

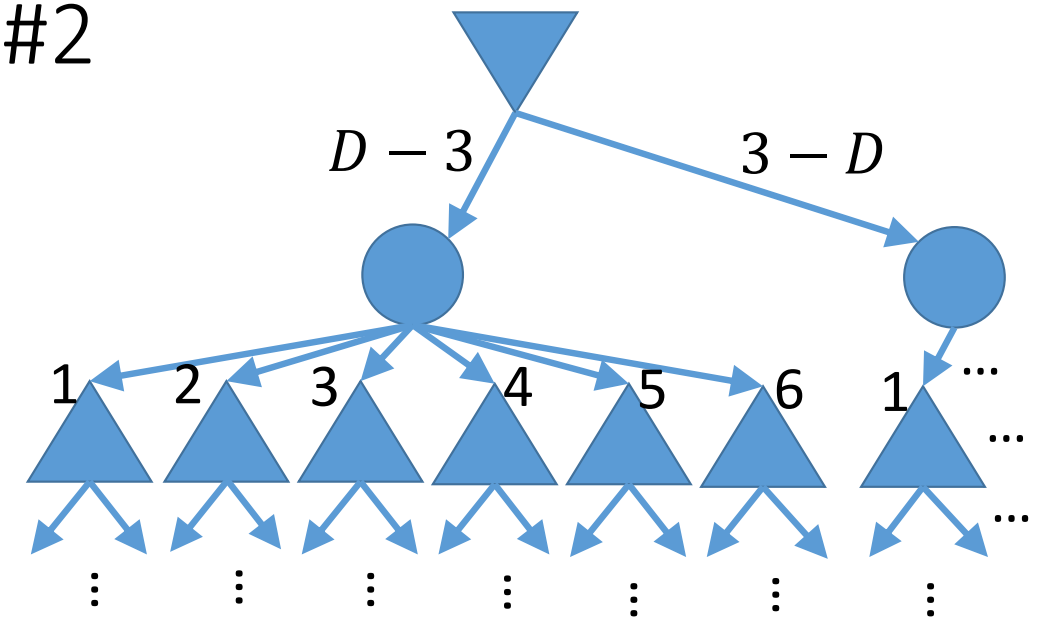
Reward: loser pays the winner a number of dollars equal to the number of spaces difference.



Expectiminimax example #2

- MIN: Min decides whether she's going to move $D - 3$ or $3 - D$ steps forward, where D is the roll of the dice.
- Chance: she rolls the dice and moves her game piece in the direction indicated.
- MAX: Max decides whether he's going to move $D - 3$ or $3 - D$ steps forward, where D is the roll of the dice.
- Chance: he rolls the dice and moves his game piece in the direction indicated.

Reward: loser pays the winner a number of dollars equal to the number of spaces difference.



Expectiminimax summary

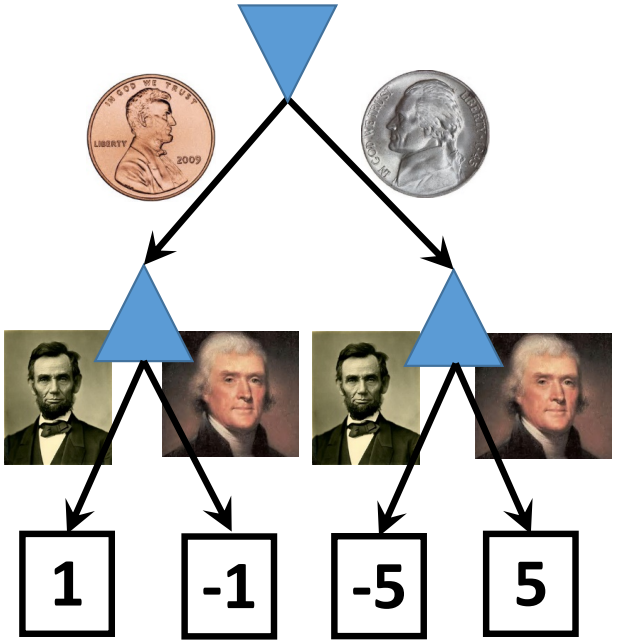
- All of the same methods are useful:
  - Alpha-Beta pruning
  - Evaluation function
  - Quiescence search, Singular move
- Computational complexity is pretty bad
  - Branching factor of the random choice can be high
  - Twice as many “levels” in the tree

Content of today's lecture

- Stochastic games: the Expectiminimax algorithm
- Imperfect information: belief states

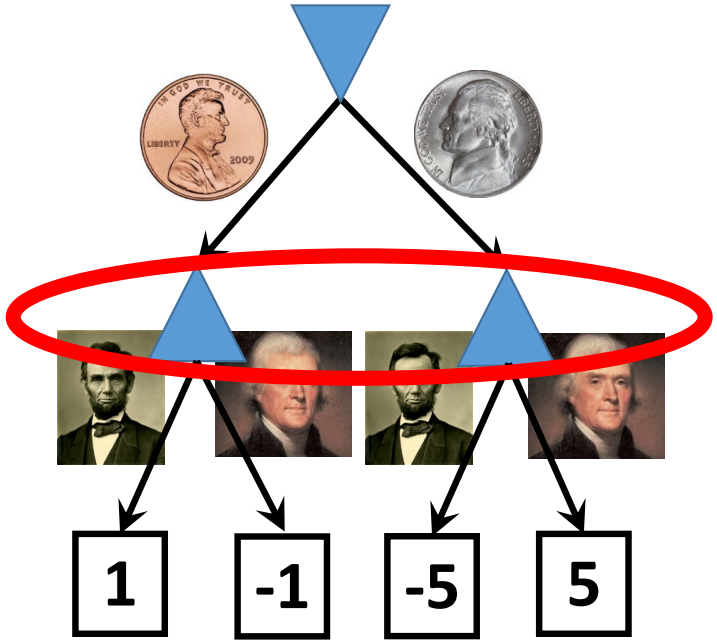
Imperfect information example

- Min chooses a coin.
- I say the name of a U.S. President.
  - If I guessed right, she gives me the coin.
  - If I guessed wrong, I have to give her a coin to match the one she has.



Imperfect information example

- The problem: I don't know which state I'm in. I only know it's one of these two.

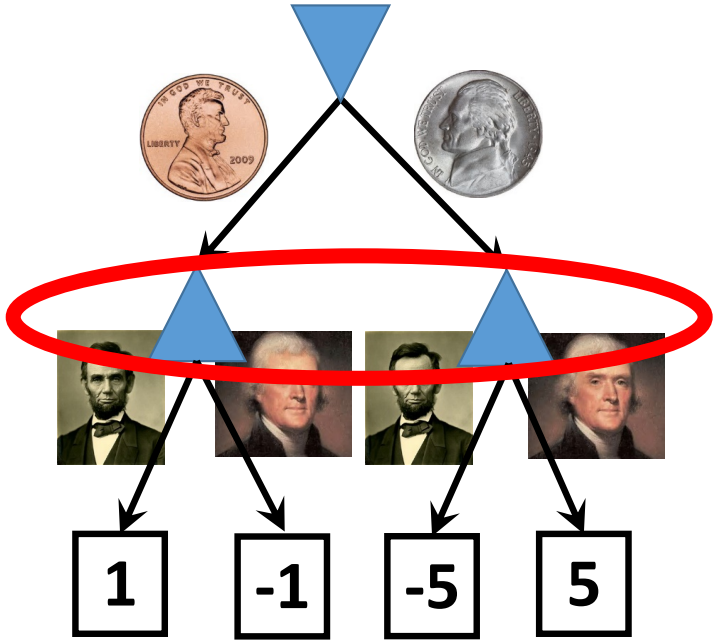


Imperfect information example

The equivalent of the minimax question, in this environment, is:

1. Is there any strategy I can use that will ***guarantee*** that I win a positive reward? (Minimax strategy)

2. If I assume a probability distribution over the set of possible states, what is the strategy that maximizes my ***expected*** reward? (Expectiminimax strategy)



Belief states

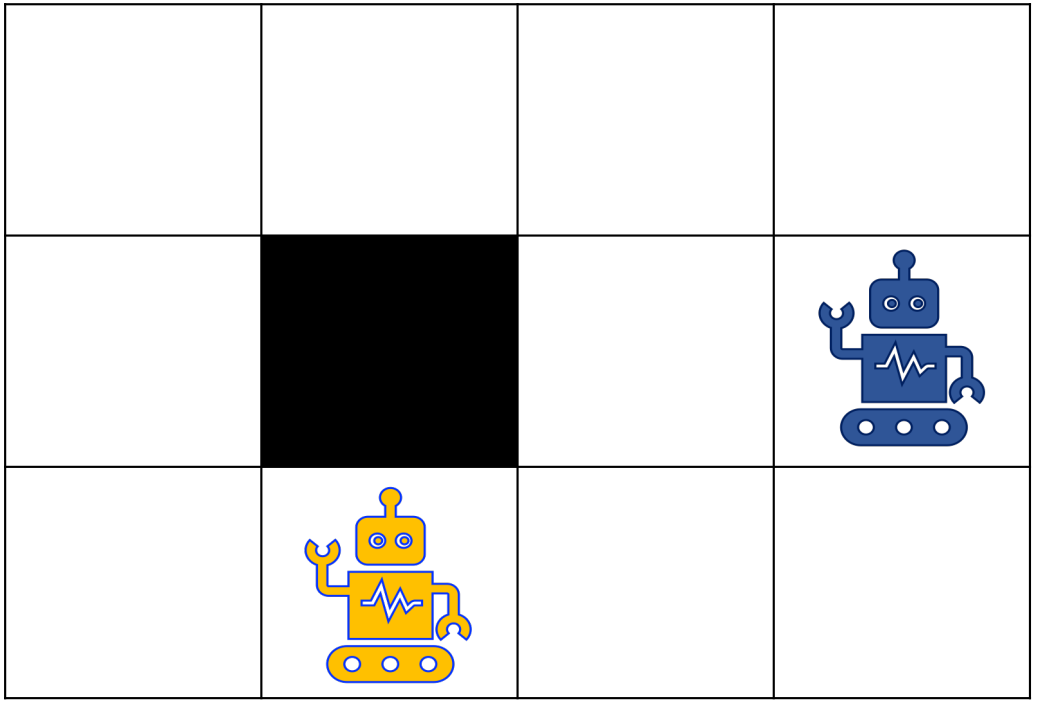
If the environment is only partially observable, then an agent's "belief state," b, is the set of all states, s, that are currently possible, given the agent's past and current actions, a, and observations, o.

Example: Maze War

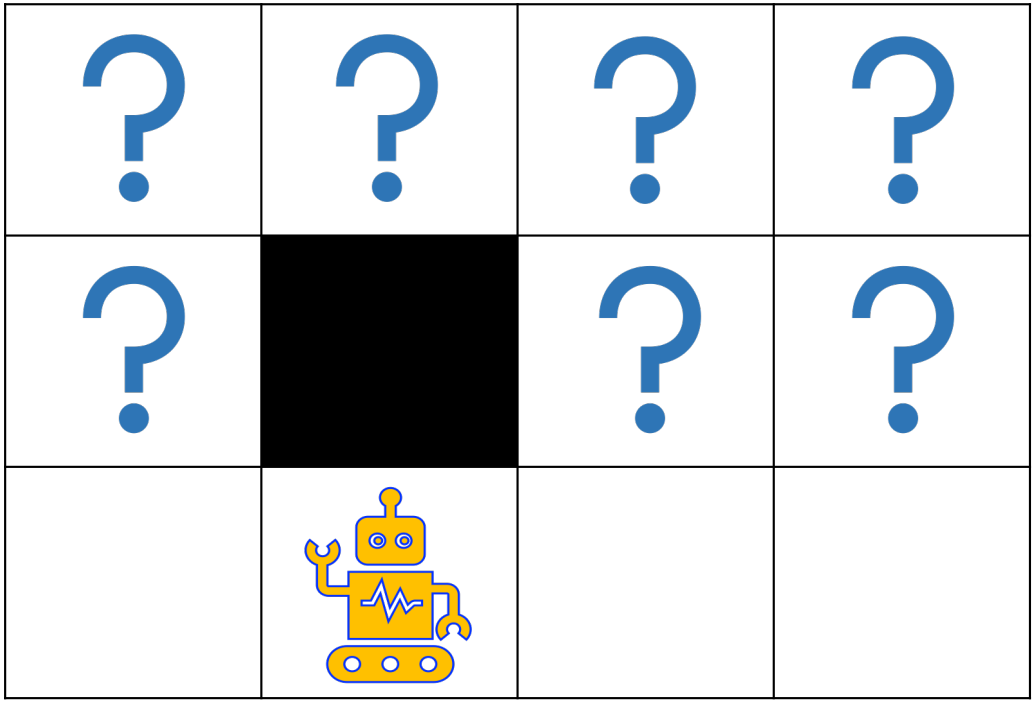
Orange robot and blue robot can see one another if there is no wall in the way. They can't see around corners.

A complete description of the current game state specifies the locations of both robots, e.g., $s = [O: (2,1), B: (4,2)]$.

The loser is the robot who moves to a position from which it can be seen by the other robot.



Example: Maze War



Orange robot knows:

- Orange robot is in (2,1)
- Blue robot might be in (1,2), (1,3), (2,3), …

we can say that

Orange robot's current belief state is

$$b = [O: (2,1), B \in \{(1,2), (1,3), (2,3), (3,2), (3,3), (4,2), (4,3)\}]$$

Belief state update equations

When the robot in belief state b performs action a and then sees observation o, it can then update its belief state using a two-part algorithm:

- $PREDICT(b, a)$ is the set of all states s' that could be reached by performing action a in any state s in the current belief state:

$$PREDICT(b, a) = \{s' : s \in b, s' = RESULT(s, a)\}$$

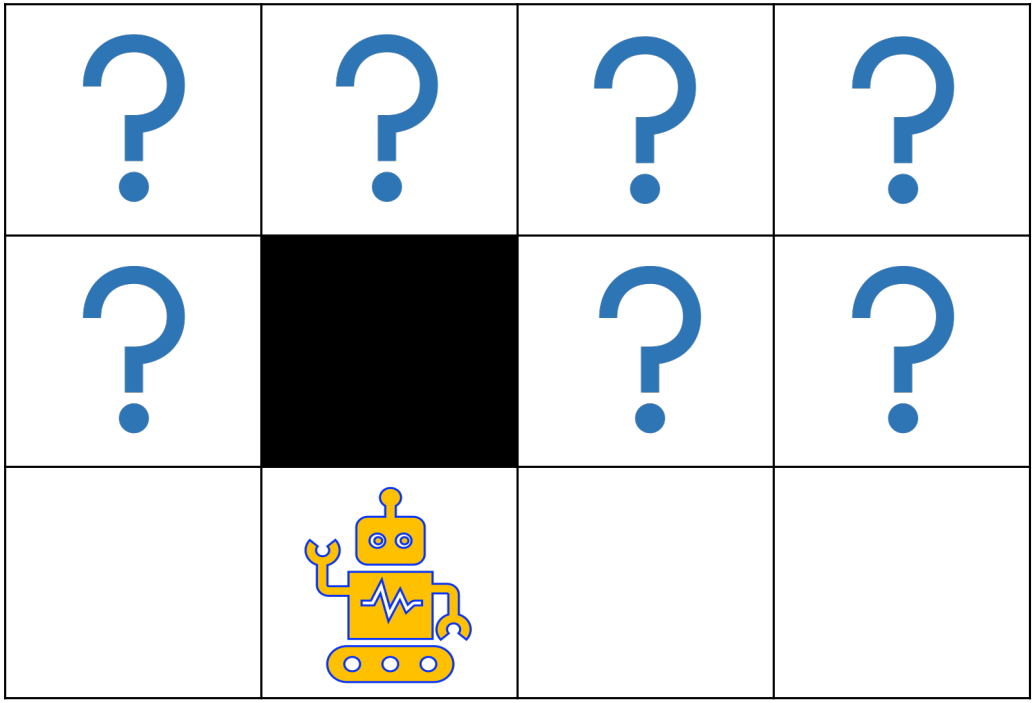
- $UPDATE(b, o)$ is the set of states in b from which it is possible to perceive o:

$$UPDATE(b, o) = \{s : s \in b, o = PERCEPT(s)\}$$

Example: Maze War

Orange robot moves as shown.

Blue robot is not observed.

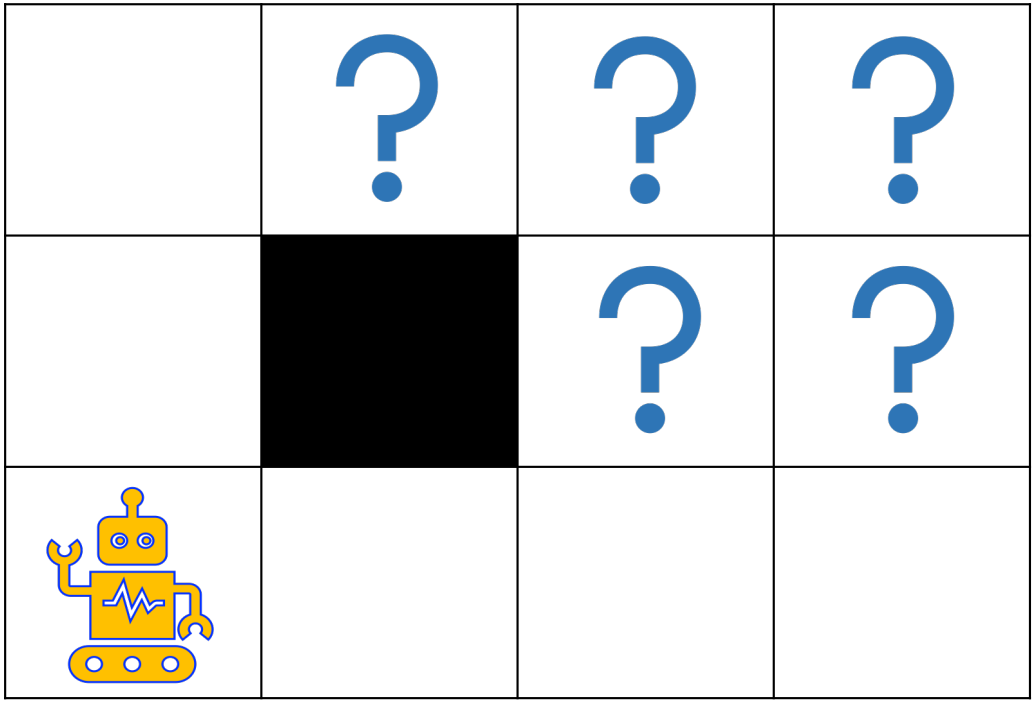


$$PREDICT(b, a) = [O: (1,1), B \in \{(1,2), (1,3), (2,3), (3,2), (3,3), (4,2), (4,3)\}]$$

Example: Maze War

Orange robot moves as shown.

Blue robot is not observed.



$$PREDICT(b, a) = [O: (1,1), B \in \{(1,2), (1,3), (2,3), (3,2), (3,3), (4,2), (4,3)\}]$$

$$UPDATE(PREDICT(b, a), o) = [O: (1,1), B \in \{(2,3), (3,2), (3,3), (4,2), (4,3)\}]$$

Stochastic games of imperfect information

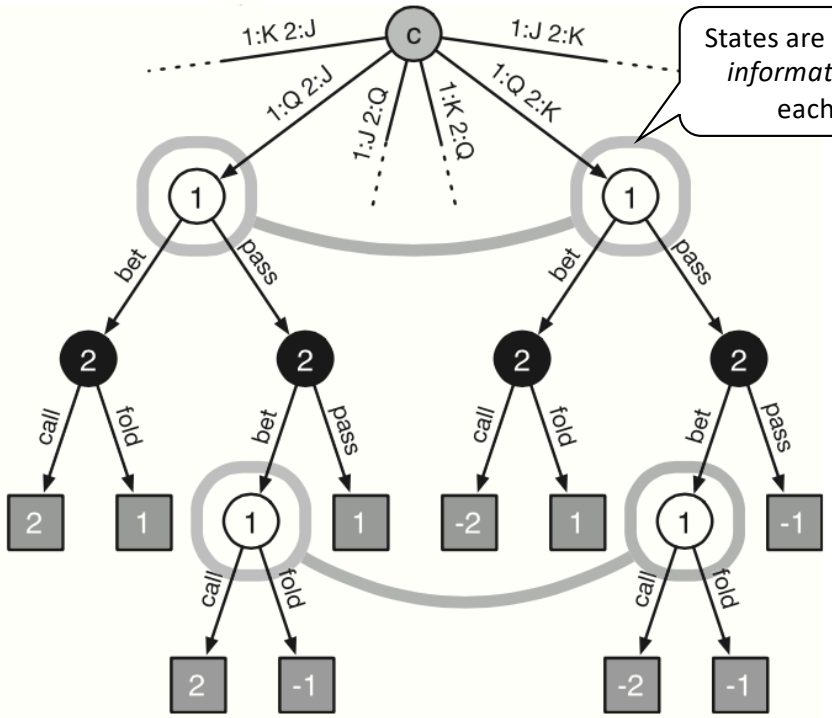


Fig. 1. Portion of the extensive-form game representation of three-card Kuhn poker (*16*). Player 1 is dealt a queen (Q), and the opponent is given either the jack (J) or king (K). Game states are circles labeled by the player acting at each state ("c" refers to chance, which randomly chooses the initial deal). The arrows show the events the acting player can choose from, labeled with their in-game meaning. The leaves are square vertices labeled with the associated utility for player 1 (player 2's utility is the negation of player 1's). The states connected by thick gray lines are part of the same information set; that is, player 1 cannot distinguish between the states in each pair because they each represent a different unobserved card being dealt to the opponent. Player 2's states are also in information sets, containing other states not pictured in this diagram.

Source

Game AI: Origins

- Minimax algorithm: Ernst Zermelo, 1912
- Chess playing with evaluation function, quiescence search, selective search:
  Claude Shannon, 1949 (paper)
- Alpha-beta search: John McCarthy, 1956
- Checkers program that learns its own evaluation function by playing against itself: Arthur Samuel, 1956 (Rodney Brooks blog post)

Game AI: State of the art

- Observable & Deterministic:
  - **Checkers:** solved in 2007
  - **Chess:** Deep learning machine teaches itself chess in 72 hours, plays at International Master Level (arXiv, September 2015)
  - **Go**: AlphaGo beats Lee Sedol, 2015
- Observable & Stochastic:
  - **Backgammon:** TD-Gammon system (1992) used *reinforcement learning* to learn a good evaluation function
- Partially Observable and Stochastic:
  - **Poker**
    - Heads-up limit hold'em poker is solved (2015)
      - Simplest variant played competitively by humans
      - Smaller number of states than checkers, but partial observability makes it difficult
      - *Essentially weakly solved* = cannot be beaten with statistical significance in a lifetime of playing
    - CMU's Libratus system beats four of the best human players at no-limit Texas Hold'em poker (2017)

Content of today's lecture

- Stochastic games: the Expectiminimax algorithm

$$U(s) = \max_a \sum_{s'} P(s'|s,a)U(s')$$

$$U(s') = \min_{a'} \sum_{s''} P(s''|s',a')U(s'')$$

- Imperfect information: belief states

$$PREDICT(b,a) = \{s' : s \in b, s' = RESULT(s,a)\}$$

$$UPDATE(b,o) = \{s : s \in b, o = PERCEPT(s)\}$$