

GAME PLAYING

CHAPTER 5, SECTIONS 1–5

Outline

- ◇ Perfect play
- ◇ Resource limits
- ◇ α - β pruning
- ◇ Games of chance
- ◇ Games of imperfect information

Games vs. search problems

“Unpredictable” opponent \Rightarrow solution is a **strategy**
specifying a move for every possible opponent reply

Time limits \Rightarrow unlikely to find goal, must approximate

Plan of attack:

- Computer considers possible lines of play (Babbage, 1846)
- Algorithm for perfect play (Zermelo, 1912; Von Neumann, 1944)
- Finite horizon, approximate evaluation (Zuse, 1945; Wiener, 1948; Shannon, 1950)
- First chess program (Turing, 1951)
- Machine learning to improve evaluation accuracy (Samuel, 1952–57)
- Pruning to allow deeper search (McCarthy, 1956)

Types of games

deterministic

chance

perfect information

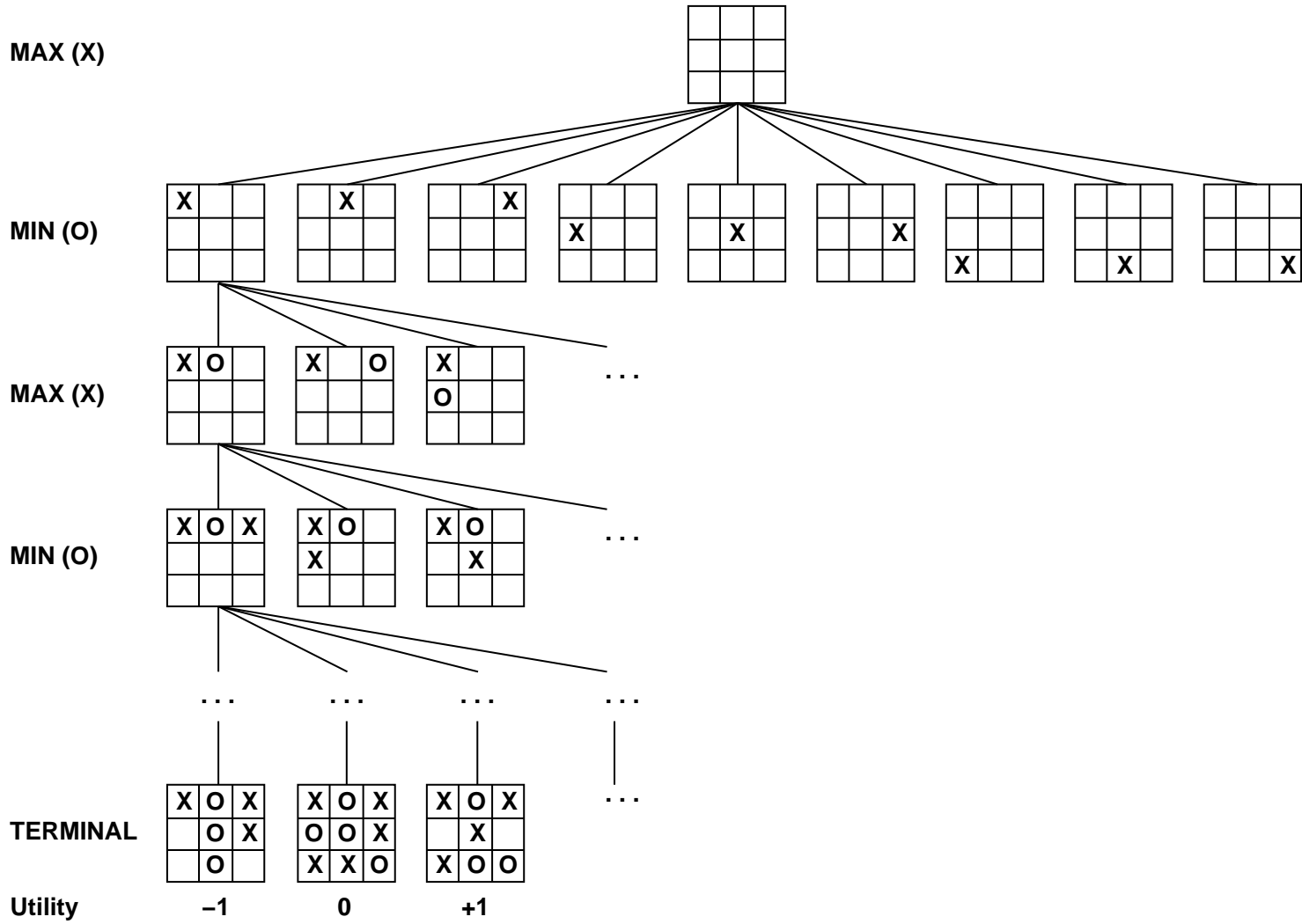
**chess, checkers,
go, othello**

**backgammon
monopoly**

imperfect information

**bridge, poker, scrabble
nuclear war**

Game tree (2-player, deterministic, turns)

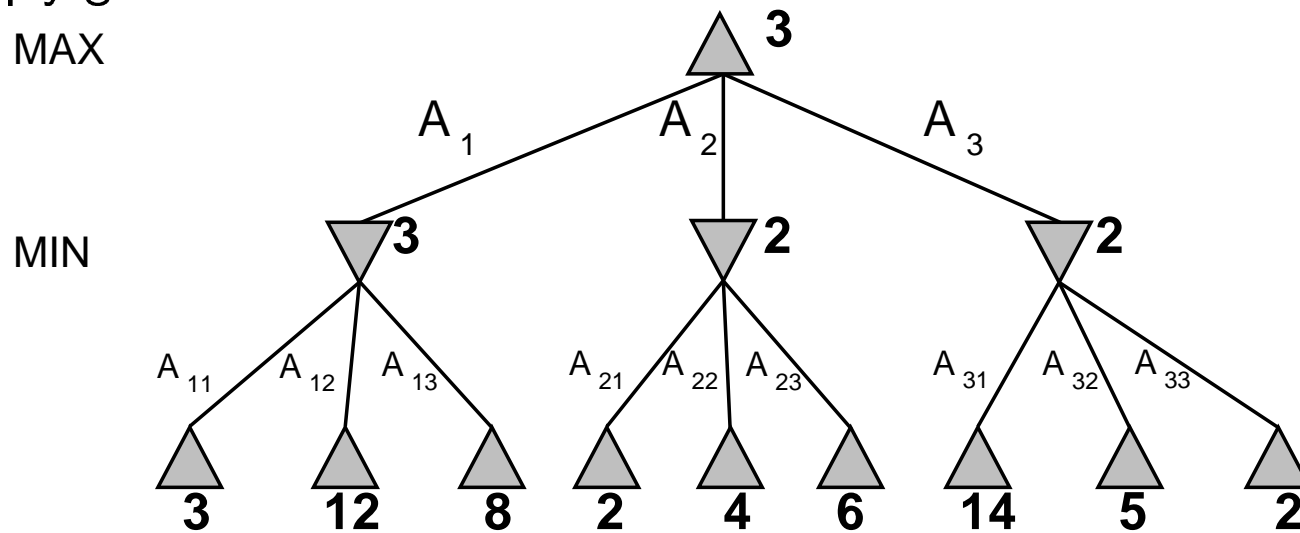


Minimax

Perfect play for deterministic, perfect-information games

Idea: choose move to position with highest *minimax value*
= best achievable payoff against best play

E.g., 2-ply game:



Minimax algorithm

function MINIMAX-DECISION(*state, game*) *returns an action*
 action, state ← the *a, s* in SUCCESSORS(*state*)
 such that MINIMAX-VALUE(*s, game*) is maximized
return *action*

function MINIMAX-VALUE(*state, game*) *returns a utility value*
 if TERMINAL-TEST(*state*) **then**
 return UTILITY(*state*)
 else if MAX is to move in *state* **then**
 return the highest MINIMAX-VALUE of SUCCESSORS(*state*)
 else
 return the lowest MINIMAX-VALUE of SUCCESSORS(*state*)

Properties of minimax

Complete??

Properties of minimax

Complete?? Only if tree is finite (chess has specific rules for this).

NB a finite strategy can exist even in an infinite tree!

Optimal??

Properties of minimax

Complete?? Yes, if tree is finite (chess has specific rules for this)

Optimal?? Yes, against an optimal opponent. Otherwise??

Time complexity??

Properties of minimax

Complete?? Yes, if tree is finite (chess has specific rules for this)

Optimal?? Yes, against an optimal opponent. Otherwise??

Time complexity?? $O(b^m)$

Space complexity??

Properties of minimax

Complete?? Yes, if tree is finite (chess has specific rules for this)

Optimal?? Yes, against an optimal opponent. Otherwise??

Time complexity?? $O(b^m)$

Space complexity?? $O(bm)$ (depth-first exploration)

For chess, $b \approx 35$, $m \approx 100$ for “reasonable” games
 \Rightarrow exact solution completely infeasible

Resource limits

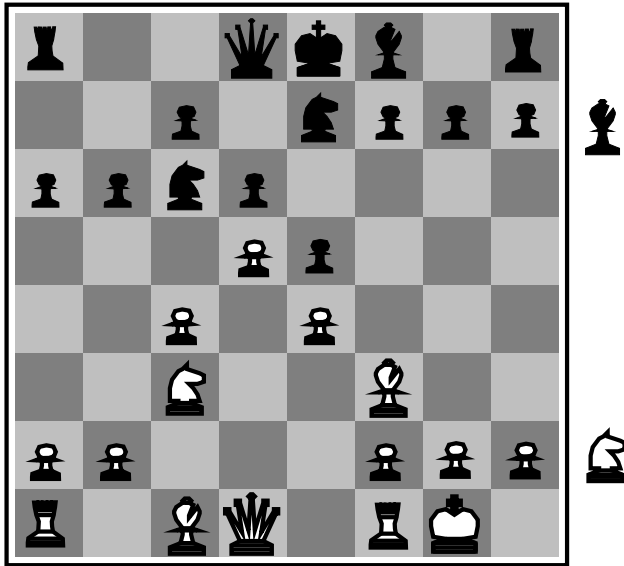
Suppose we have 100 seconds, explore 10^4 nodes/second

$\Rightarrow 10^6$ nodes per move

Standard approach:

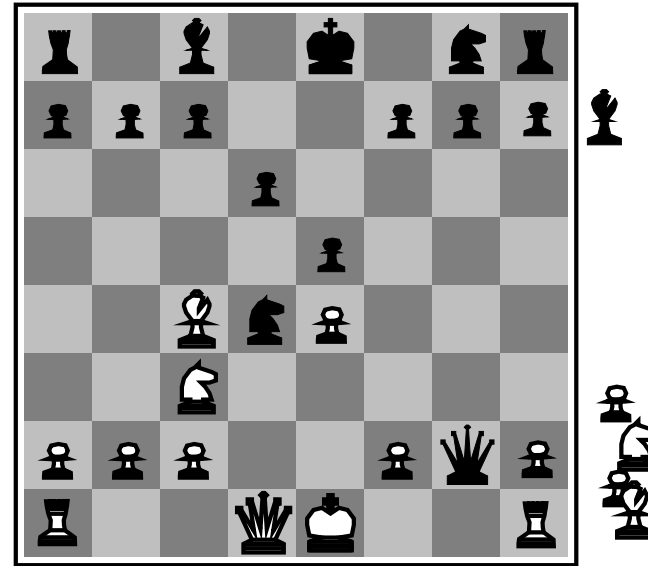
- *cutoff test*
e.g., depth limit (perhaps add *quiescence search*)
- *evaluation function*
= estimated desirability of position

Evaluation functions



Black to move

White slightly better



White to move

Black winning

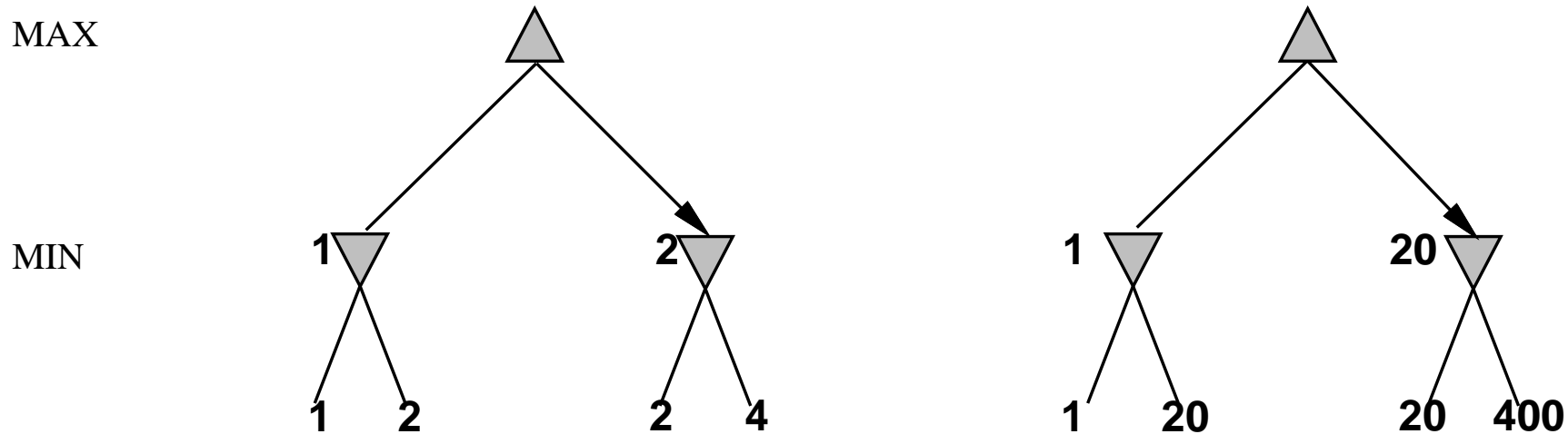
For chess, typically *linear* weighted sum of *features*

$$Eval(s) = w_1 f_1(s) + w_2 f_2(s) + \dots + w_n f_n(s)$$

e.g., $w_1 = 9$ with

$f_1(s) = (\text{number of white queens}) - (\text{number of black queens}),$ etc.

Digression: Exact values don't matter



Behaviour is preserved under any *monotonic* transformation of `EVAL`

Only the order matters:

payoff in deterministic games acts as an *ordinal utility* function

Cutting off search

MINIMAXCUTOFF is identical to MINIMAXVALUE except

1. TERMINAL? is replaced by CUTOFF?
2. UTILITY is replaced by EVAL

Does it work in practice?

$$b^m = 10^6, \quad b = 35 \quad \Rightarrow \quad m = 4$$

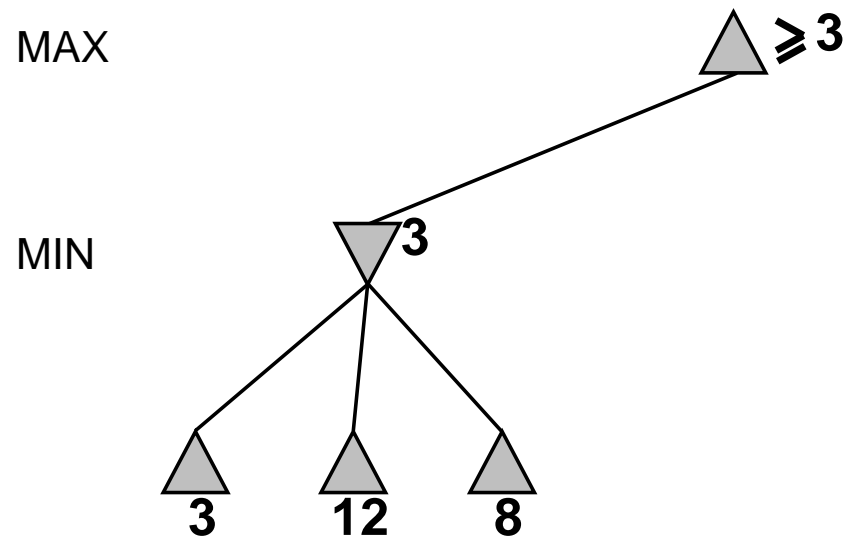
4-ply lookahead is a hopeless chess player!

4-ply \approx human novice

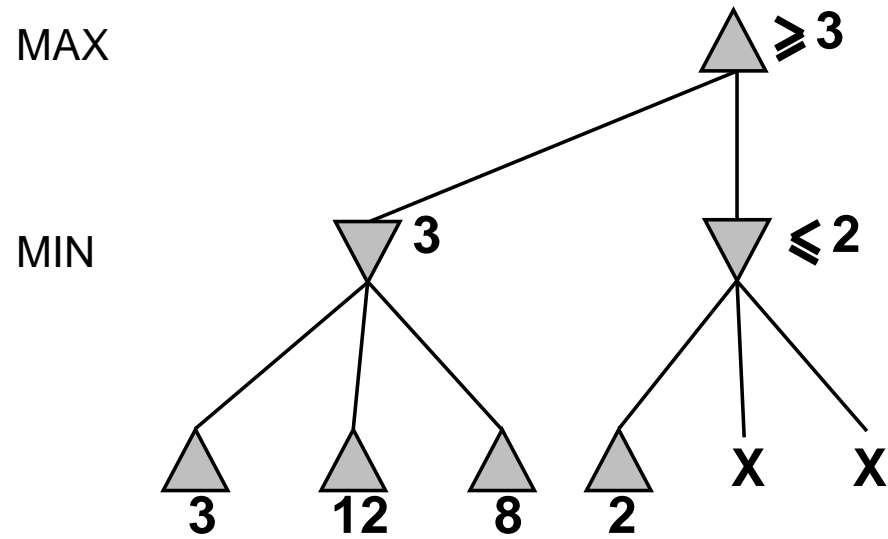
8-ply \approx typical PC, human master

12-ply \approx Deep Blue, Kasparov

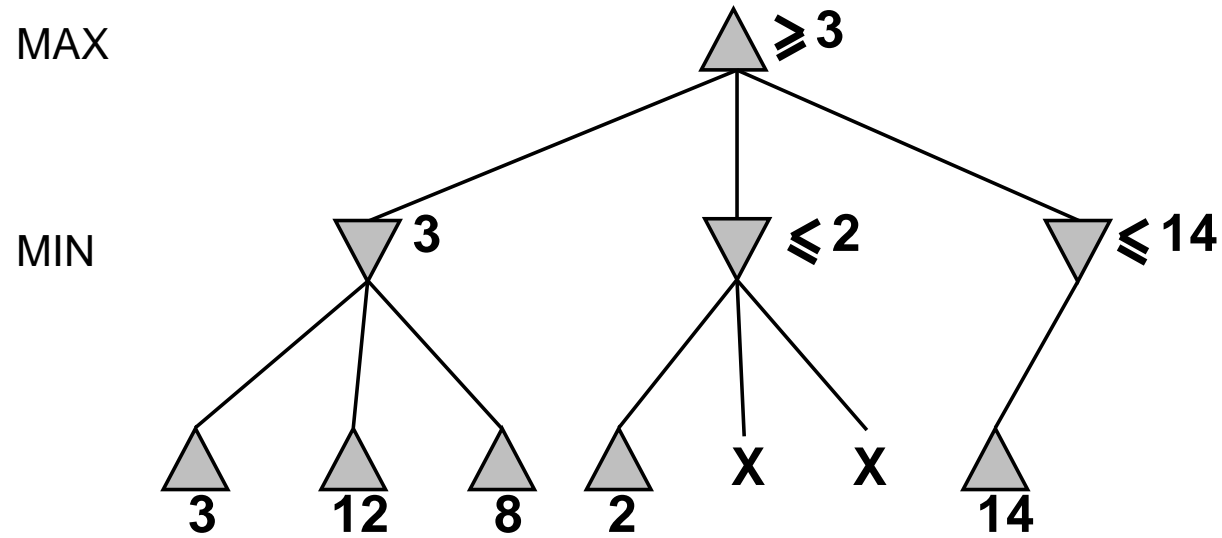
α - β pruning example



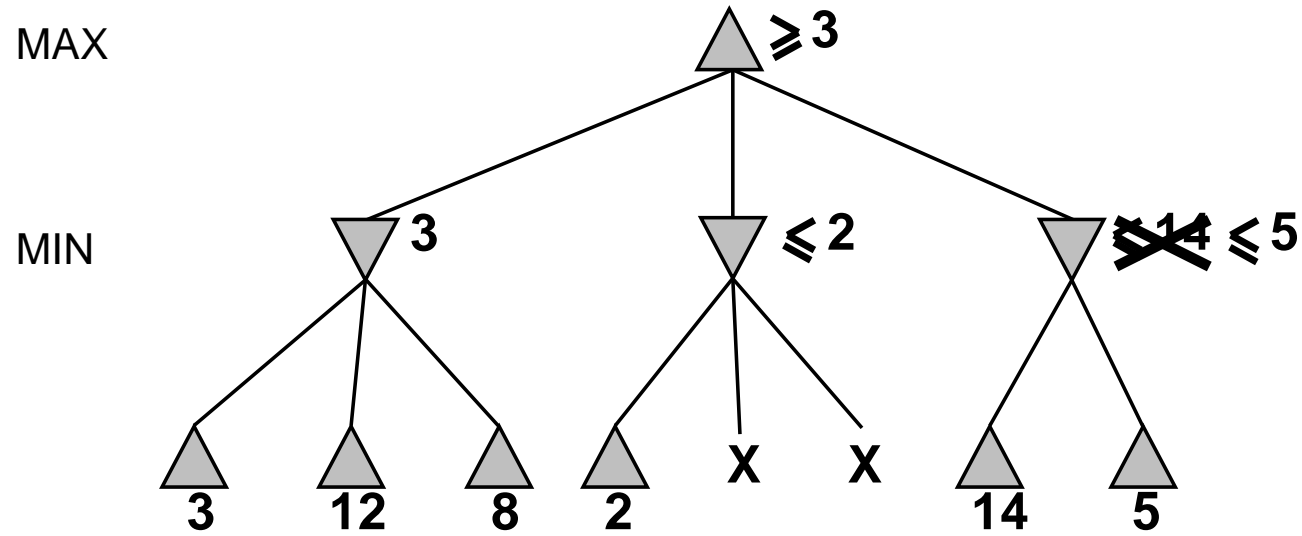
α - β pruning example



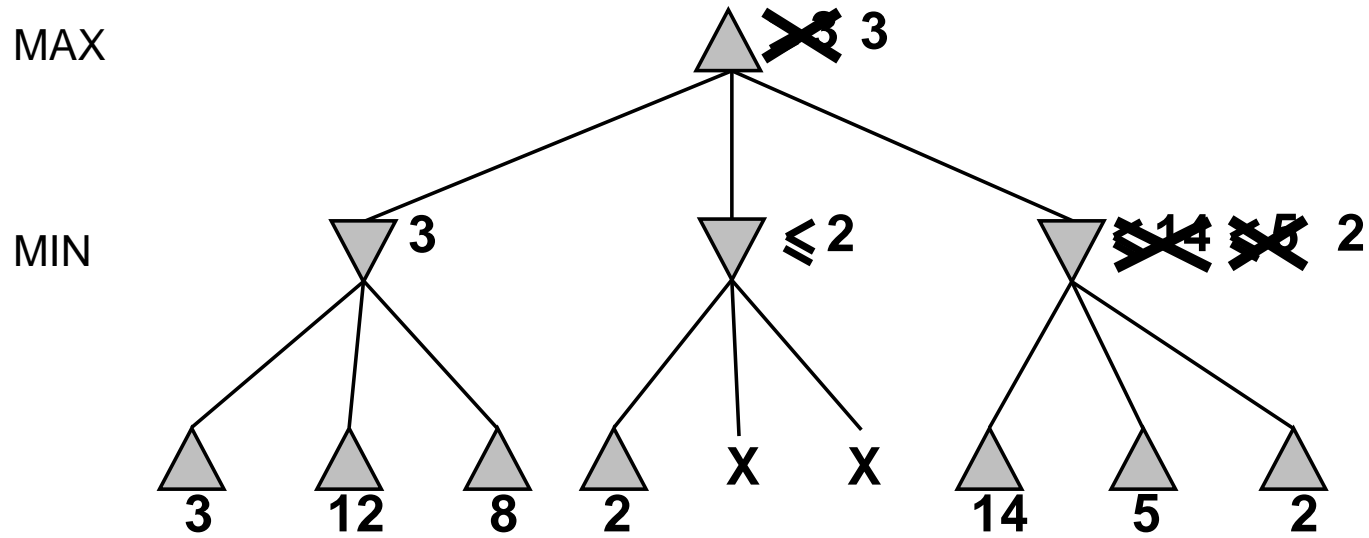
α - β pruning example



α - β pruning example



α - β pruning example



Properties of α - β

Pruning *does not* affect final result

Good move ordering improves effectiveness of pruning

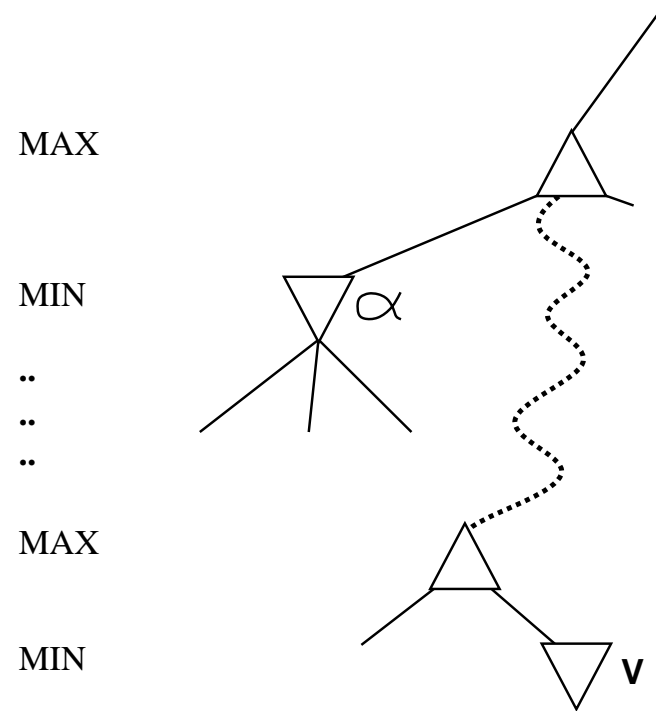
With “perfect ordering,” time complexity = $O(b^{m/2})$

⇒ *doubles* depth of search

⇒ can easily reach depth 8 and play good chess

A simple example of the value of reasoning about which computations are relevant (a form of *metareasoning*)

Why is it called α - β ?



α is the best value (to MAX) found so far off the current path

If V is worse than α , MAX will avoid it \Rightarrow prune that branch

Define β similarly for MIN

The α - β algorithm

function ALPHA-BETA-SEARCH(*state*, *game*) **returns** an action
action, *state* \leftarrow the *a*, *s* **in** SUCCESSORS[*game*](*state*)
 such that MIN-VALUE(*s*, *game*, $-\infty$, $+\infty$) is maximized
return *action*

function MAX-VALUE(*state*, *game*, α , β) **returns** the minimax value of *state*
if CUTOFF-TEST(*state*) **then return** EVAL(*state*)
for each *s* **in** SUCCESSORS(*state*) **do**
 $\alpha \leftarrow \max(\alpha, \text{MIN-VALUE}(s, \text{game}, \alpha, \beta))$
 if $\alpha \geq \beta$ **then return** β
return α

function MIN-VALUE(*state*, *game*, α , β) **returns** the minimax value of *state*
if CUTOFF-TEST(*state*) **then return** EVAL(*state*)
for each *s* **in** SUCCESSORS(*state*) **do**
 $\beta \leftarrow \min(\beta, \text{MAX-VALUE}(s, \text{game}, \alpha, \beta))$
 if $\beta \leq \alpha$ **then return** α
return β

Deterministic games in practice

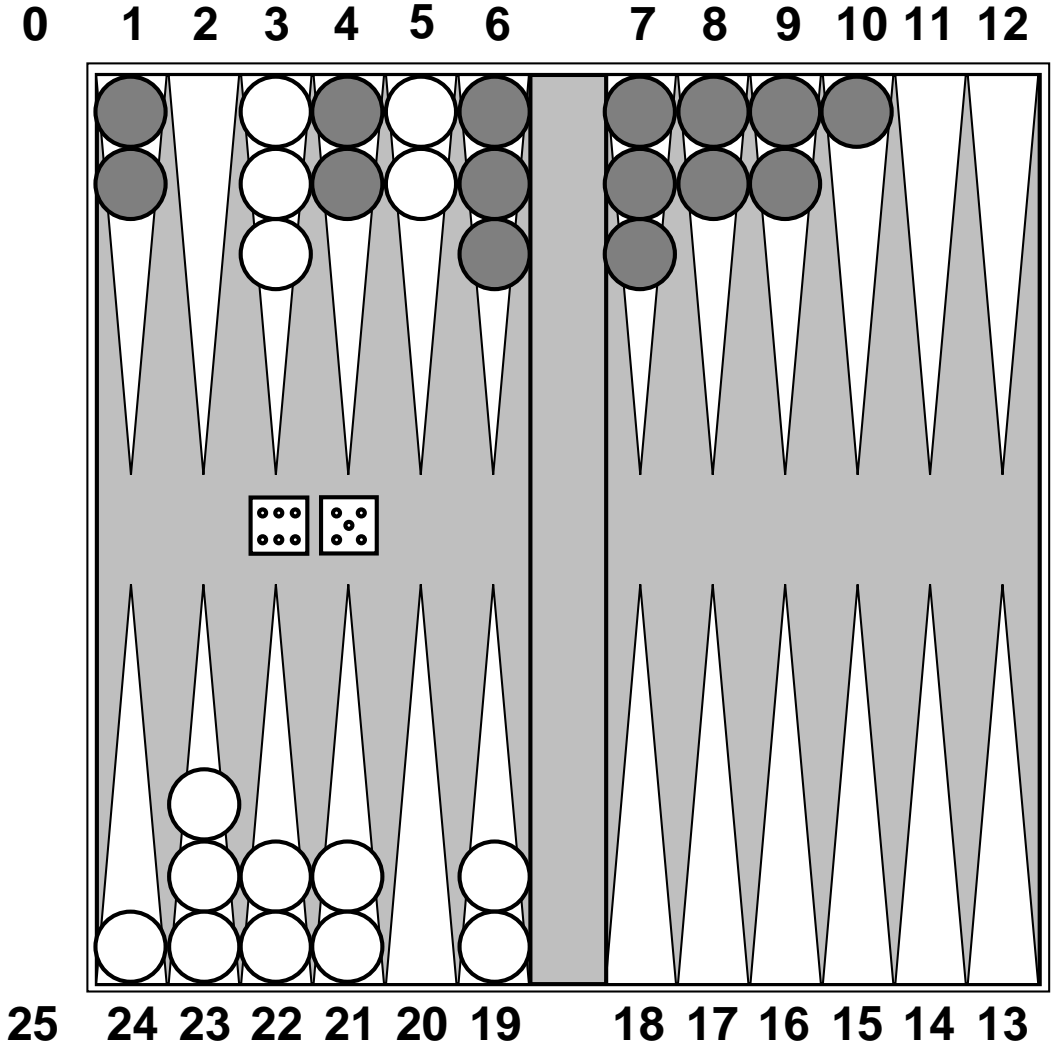
Checkers: Chinook ended 40-year-reign of human world champion Marion Tinsley in 1994. Used an endgame database defining perfect play for all positions involving 8 or fewer pieces on the board, a total of 443,748,401,247 positions.

Chess: Deep Blue defeated human world champion Gary Kasparov in a six-game match in 1997. Deep Blue searches 200 million positions per second, uses very sophisticated evaluation, and undisclosed methods for extending some lines of search up to 40 ply.

Othello: human champions refuse to compete against computers, who are too good.

Go: human champions refuse to compete against computers, who are too bad. In go, $b > 300$, so most programs use pattern knowledge bases to suggest plausible moves.

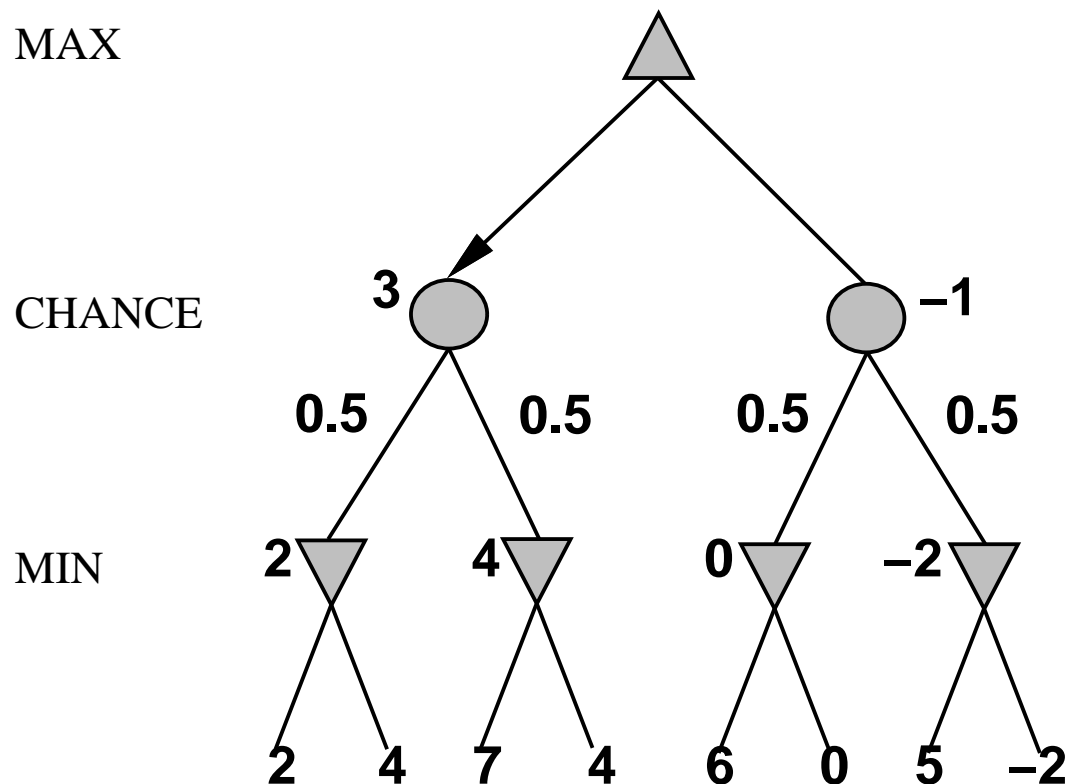
Nondeterministic games: backgammon



Nondeterministic games in general

In nondeterministic games, chance introduced by dice, card-shuffling

Simplified example with coin-flipping:



Algorithm for nondeterministic games

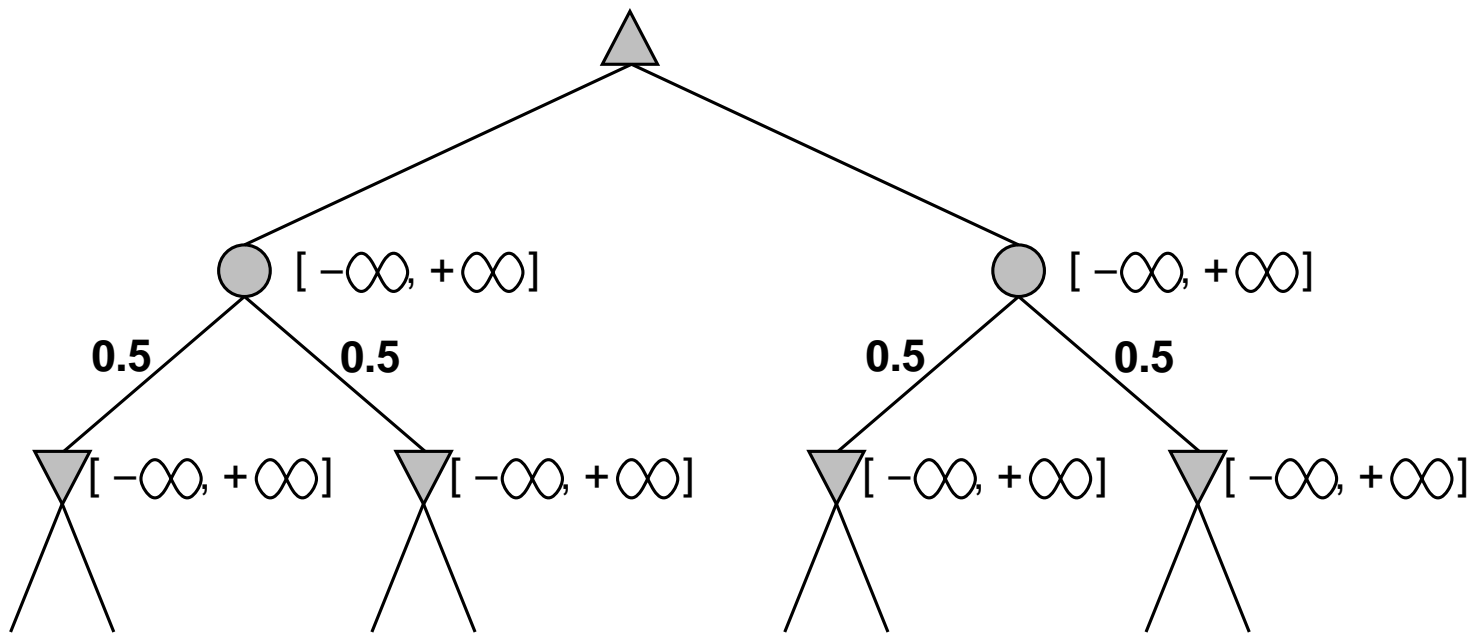
EXPECTIMINIMAX gives perfect play

Just like MINIMAX, except we must also handle chance nodes:

```
...
if state is a MAX node then
    return the highest EXPECTIMINIMAX-VALUE of SUCCESSORS(state)
if state is a MIN node then
    return the lowest EXPECTIMINIMAX-VALUE of SUCCESSORS(state)
if state is a chance node then
    return average of EXPECTIMINIMAX-VALUE of SUCCESSORS(state)
...
```

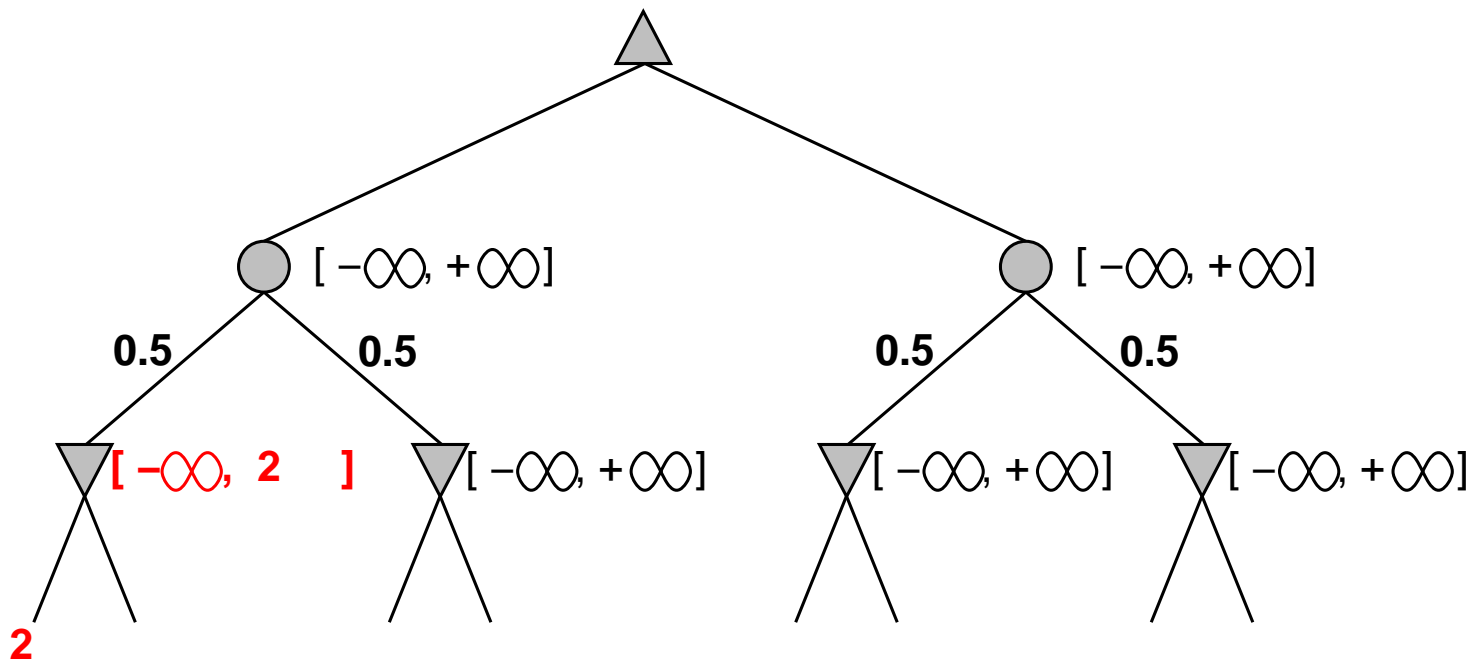
Pruning in nondeterministic game trees

A version of α - β pruning is possible:



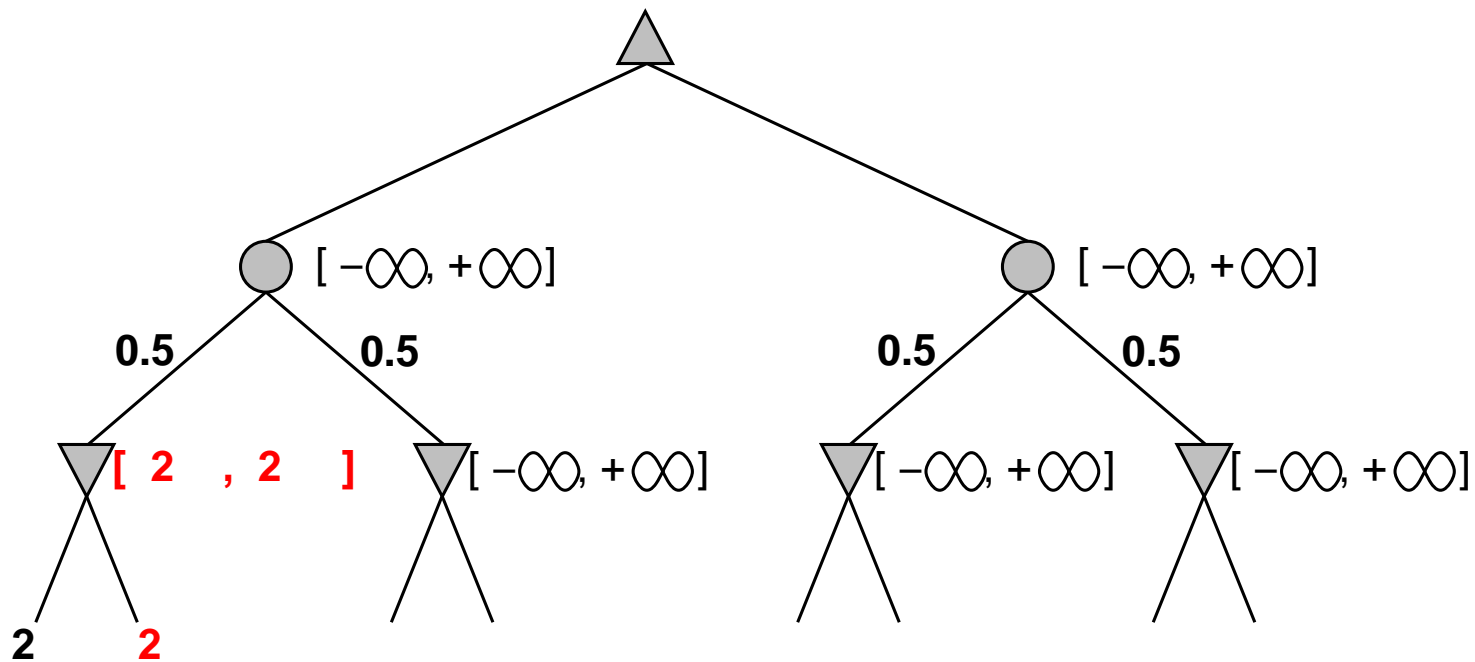
Pruning in nondeterministic game trees

A version of α - β pruning is possible:



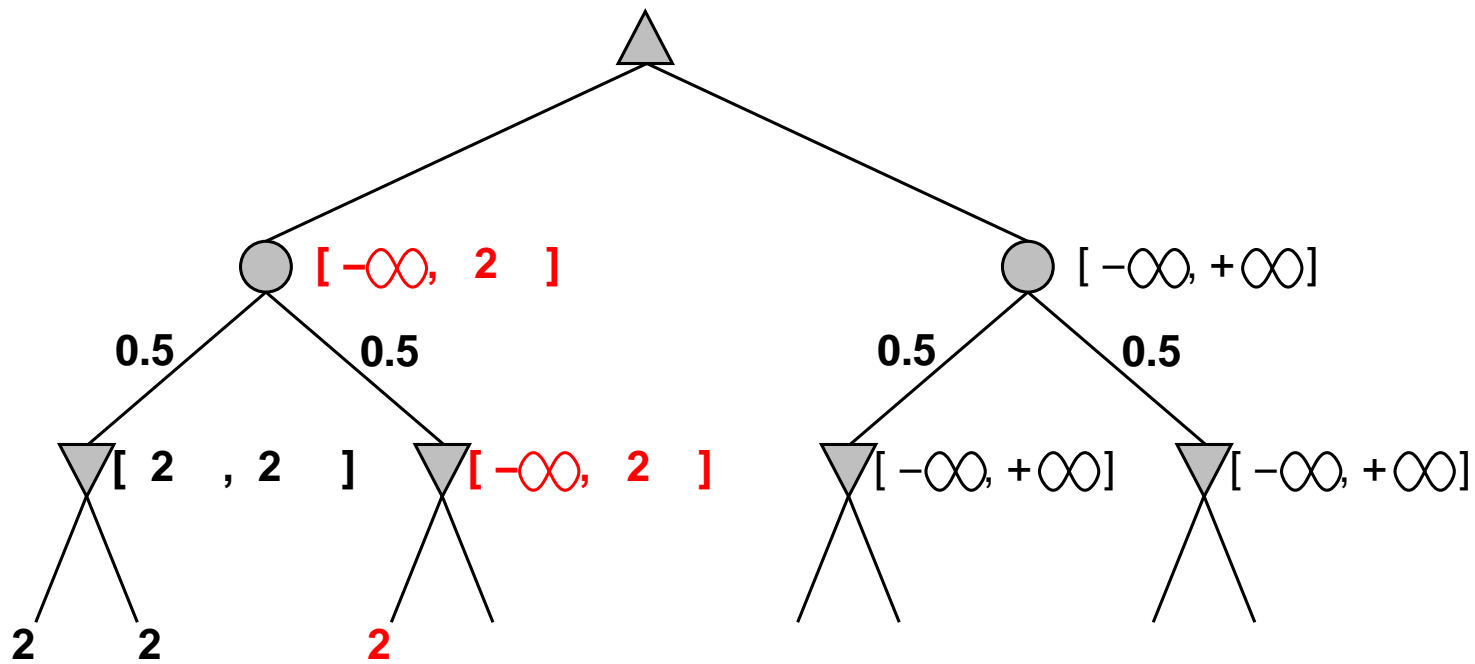
Pruning in nondeterministic game trees

A version of α - β pruning is possible:



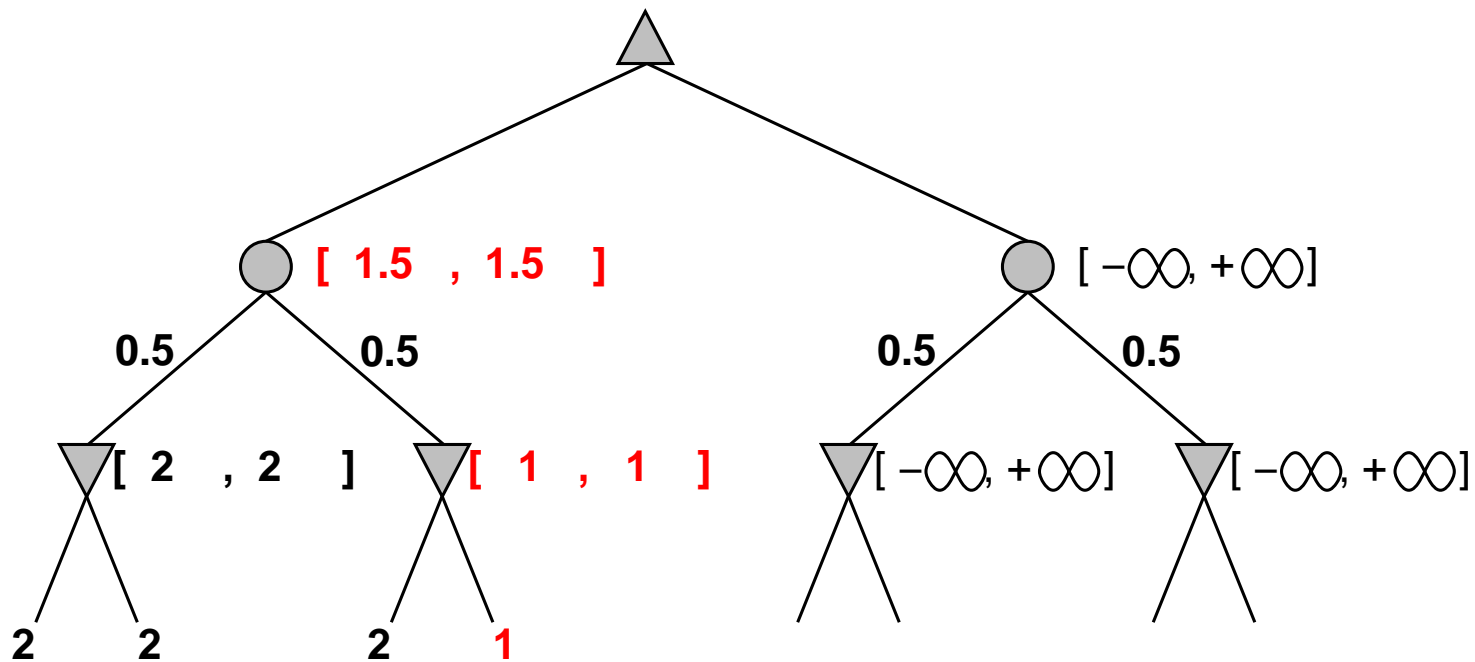
Pruning in nondeterministic game trees

A version of α - β pruning is possible:



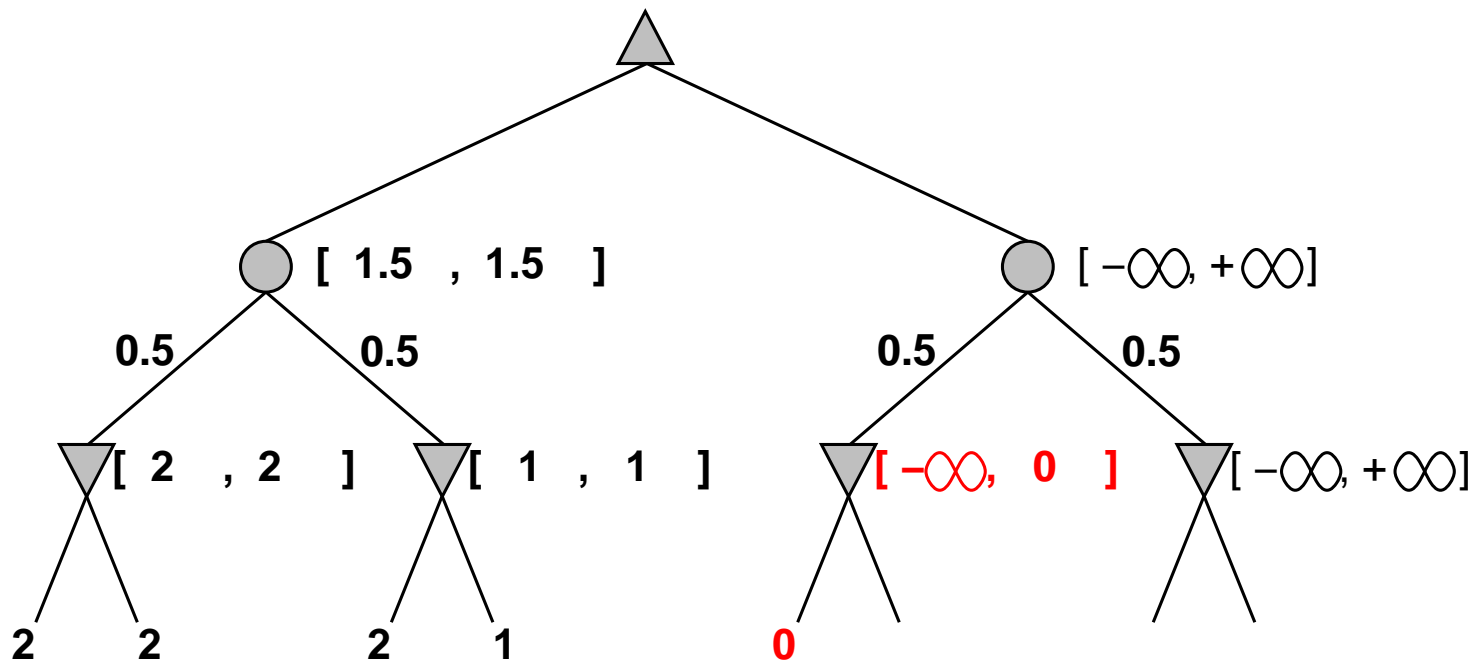
Pruning in nondeterministic game trees

A version of α - β pruning is possible:



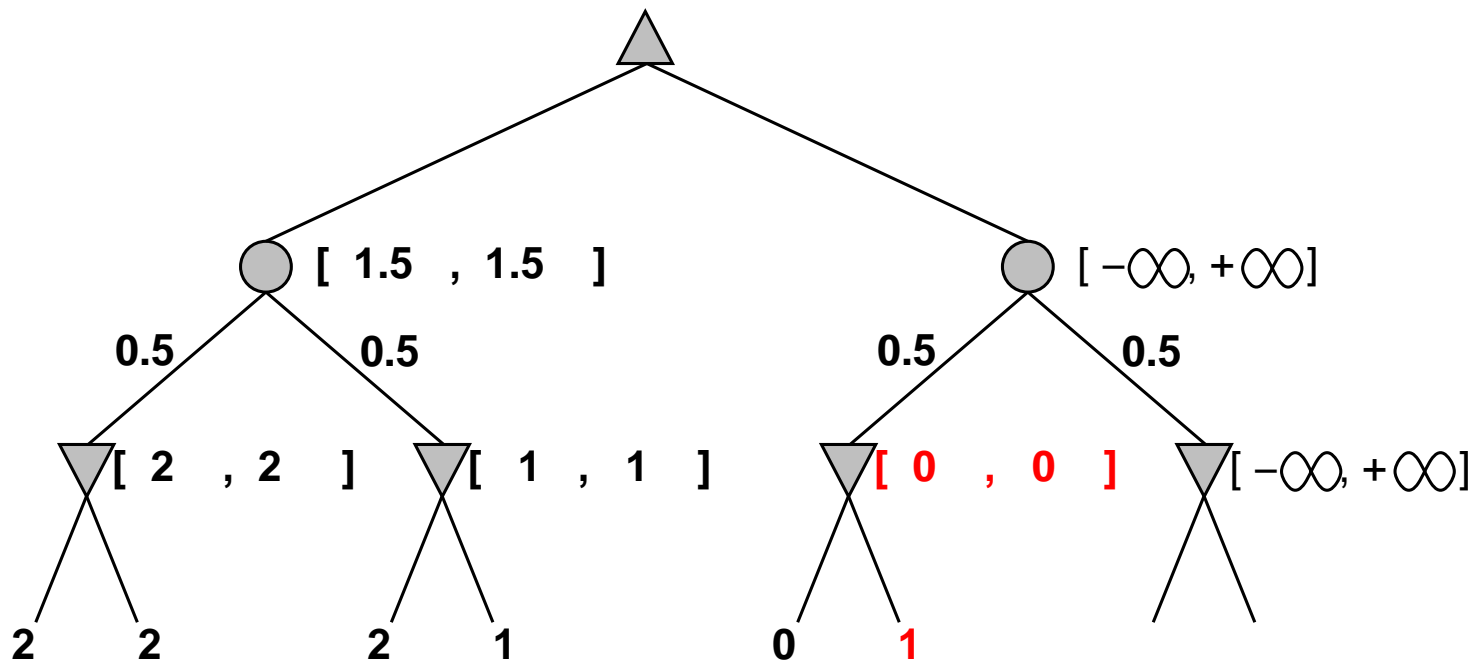
Pruning in nondeterministic game trees

A version of α - β pruning is possible:



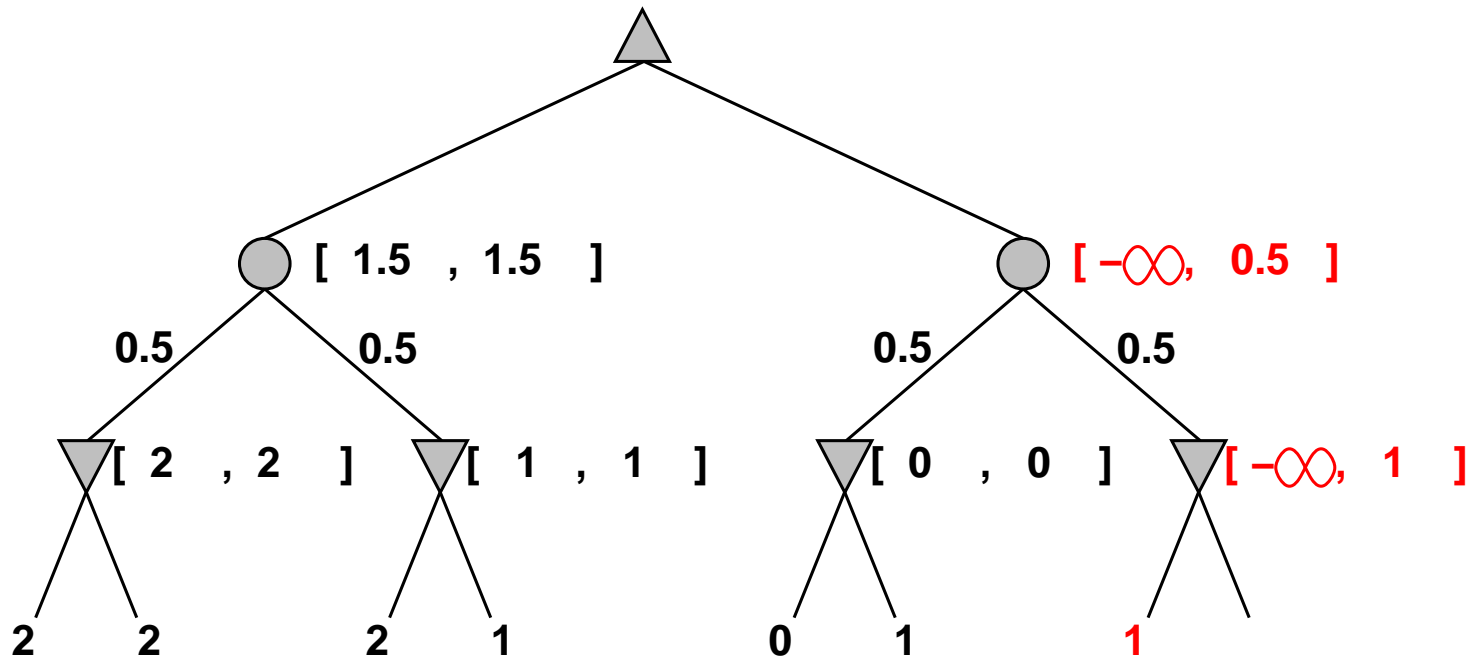
Pruning in nondeterministic game trees

A version of α - β pruning is possible:



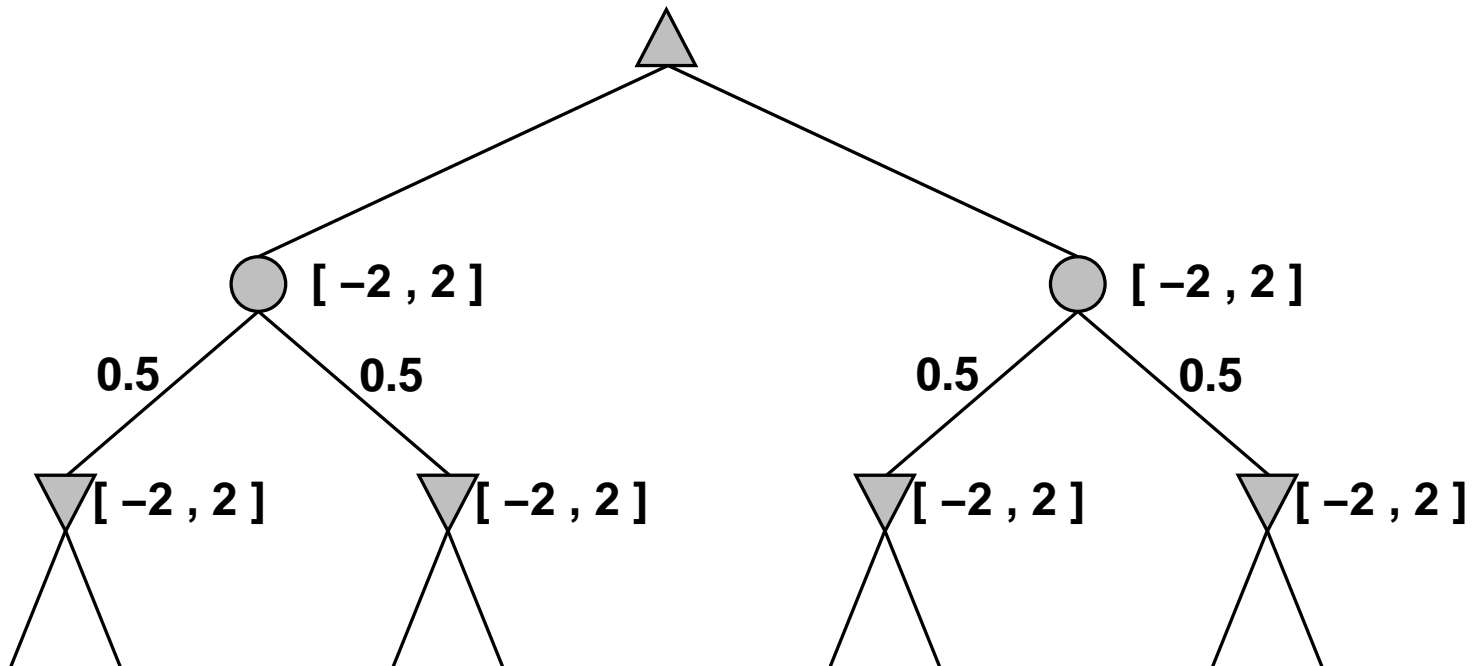
Pruning in nondeterministic game trees

A version of α - β pruning is possible:



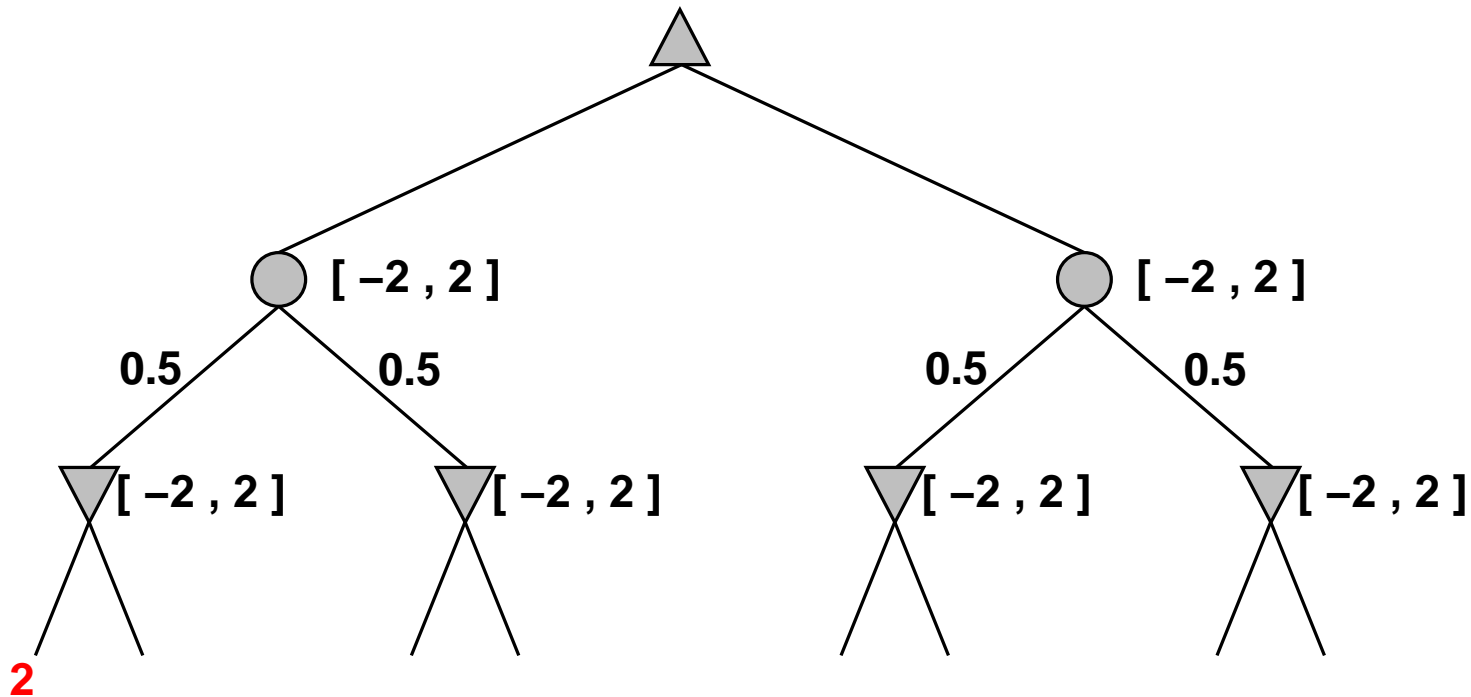
Pruning contd.

More pruning occurs if we can bound the leaf values



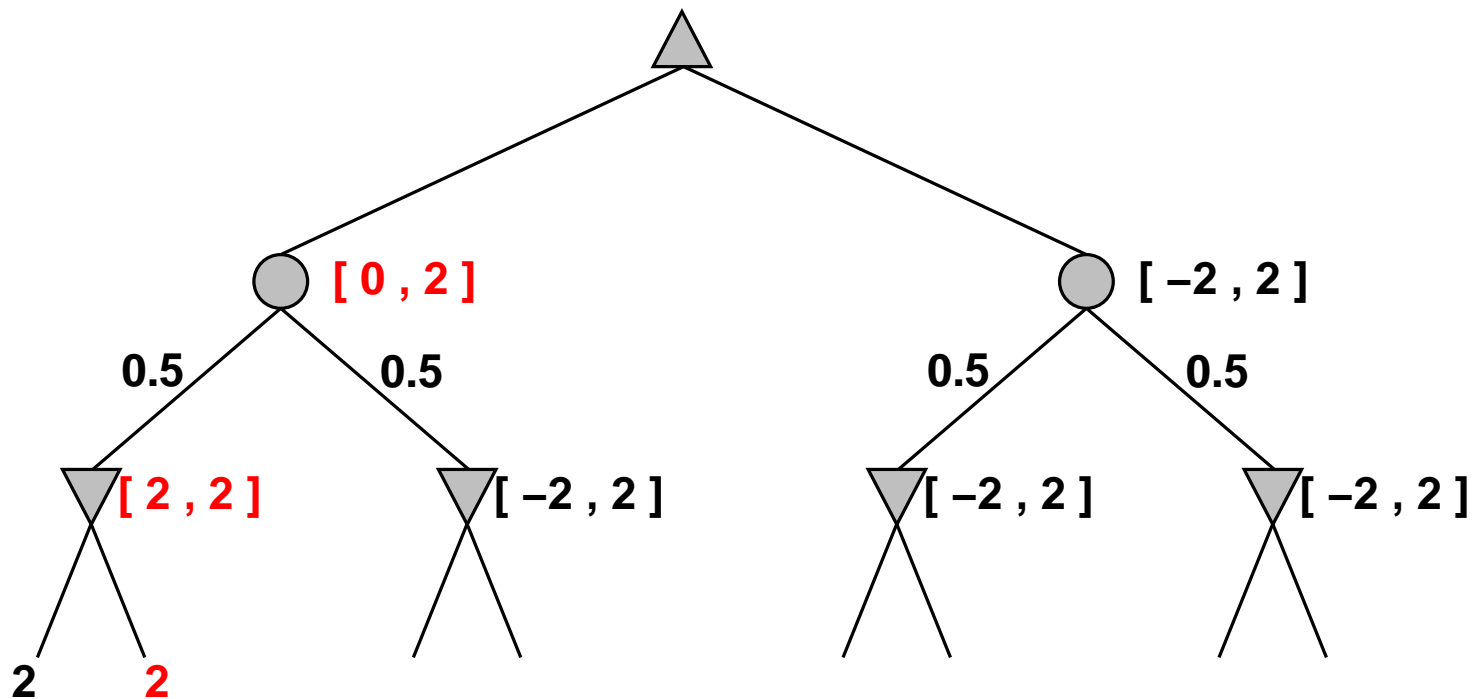
Pruning contd.

More pruning occurs if we can bound the leaf values



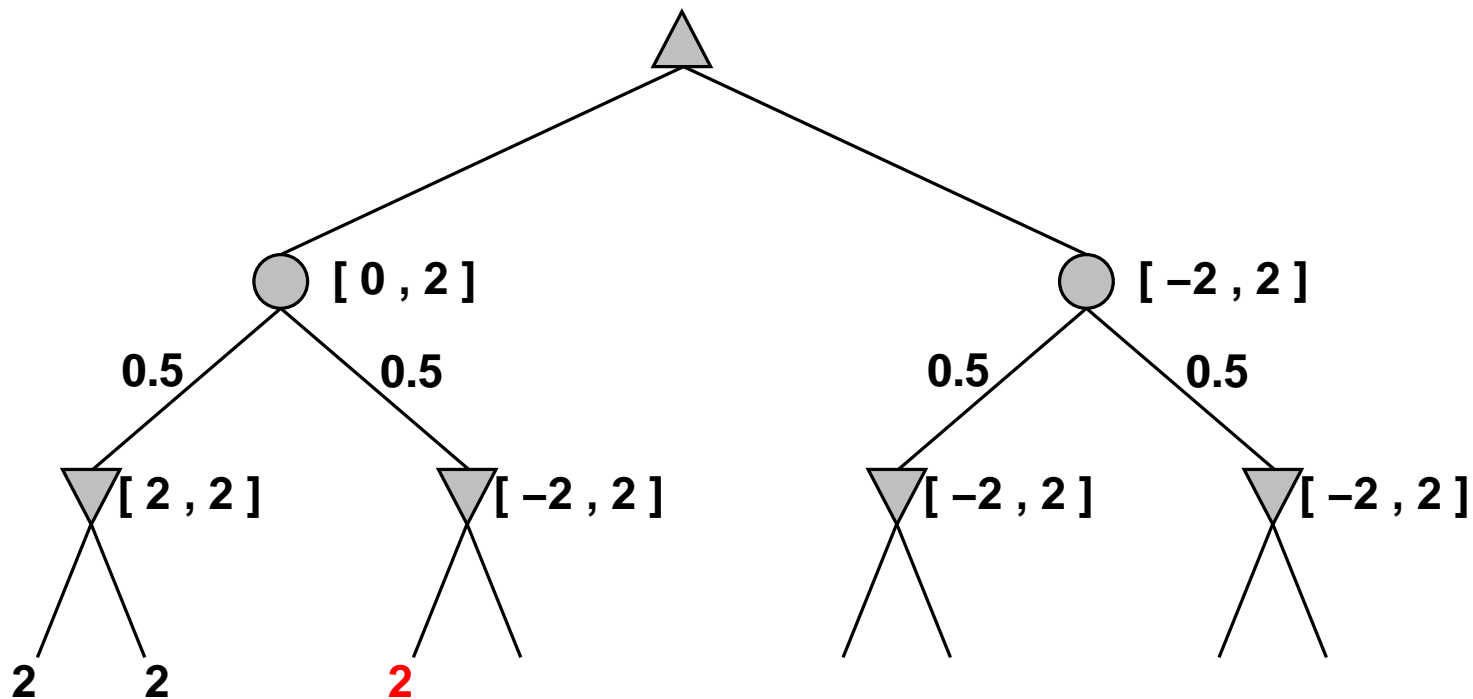
Pruning contd.

More pruning occurs if we can bound the leaf values



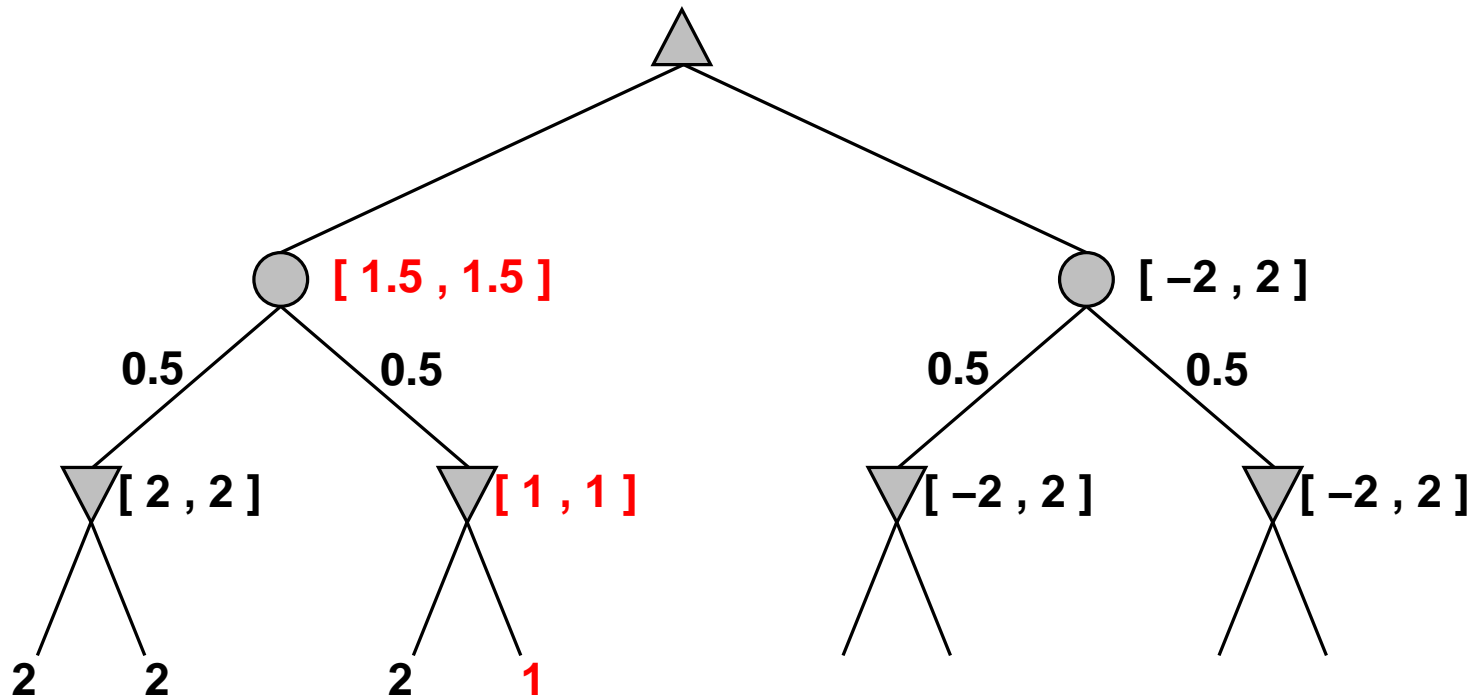
Pruning contd.

More pruning occurs if we can bound the leaf values



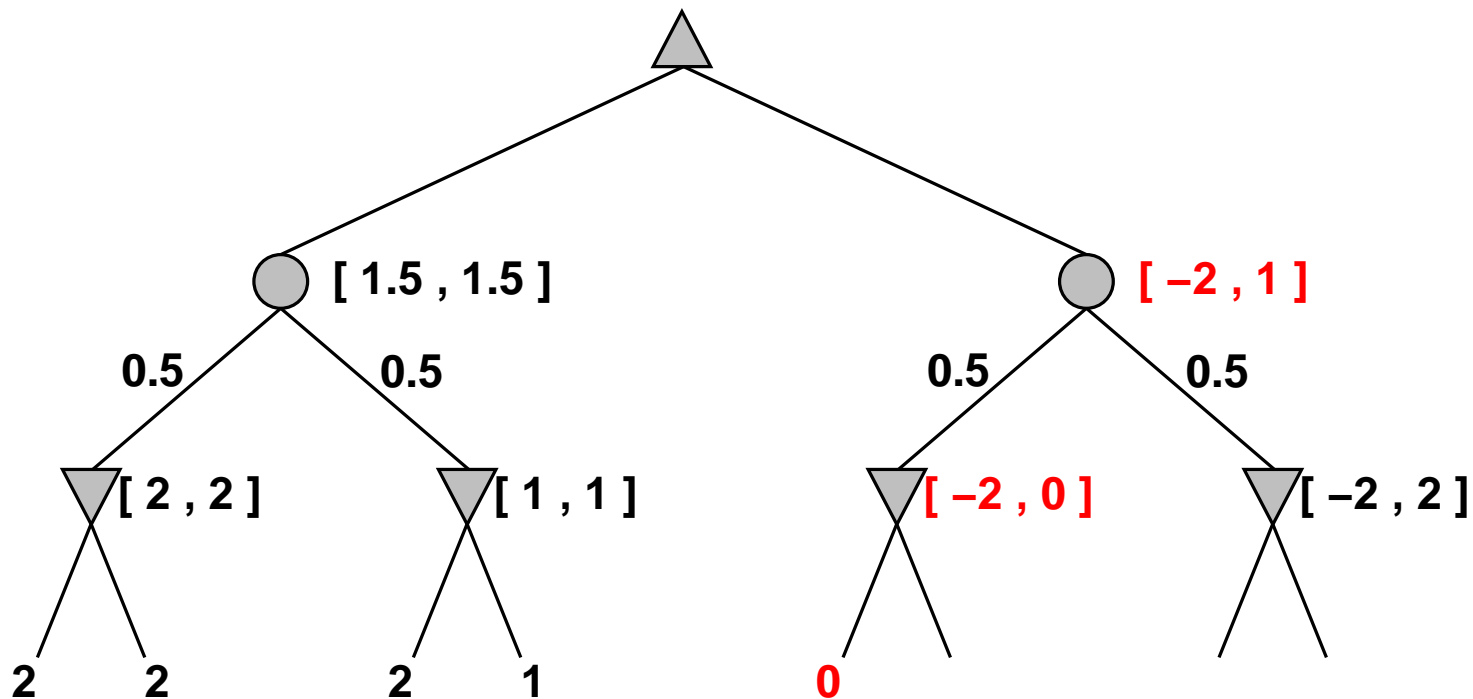
Pruning contd.

More pruning occurs if we can bound the leaf values



Pruning contd.

More pruning occurs if we can bound the leaf values



Nondeterministic games in practice

Dice rolls increase b : 21 possible rolls with 2 dice

Backgammon \approx 20 legal moves (can be 6,000 with 1-1 roll)

$$\text{depth } 4 = 20 \times (21 \times 20)^3 \approx 1.2 \times 10^9$$

As depth increases, probability of reaching a given node shrinks

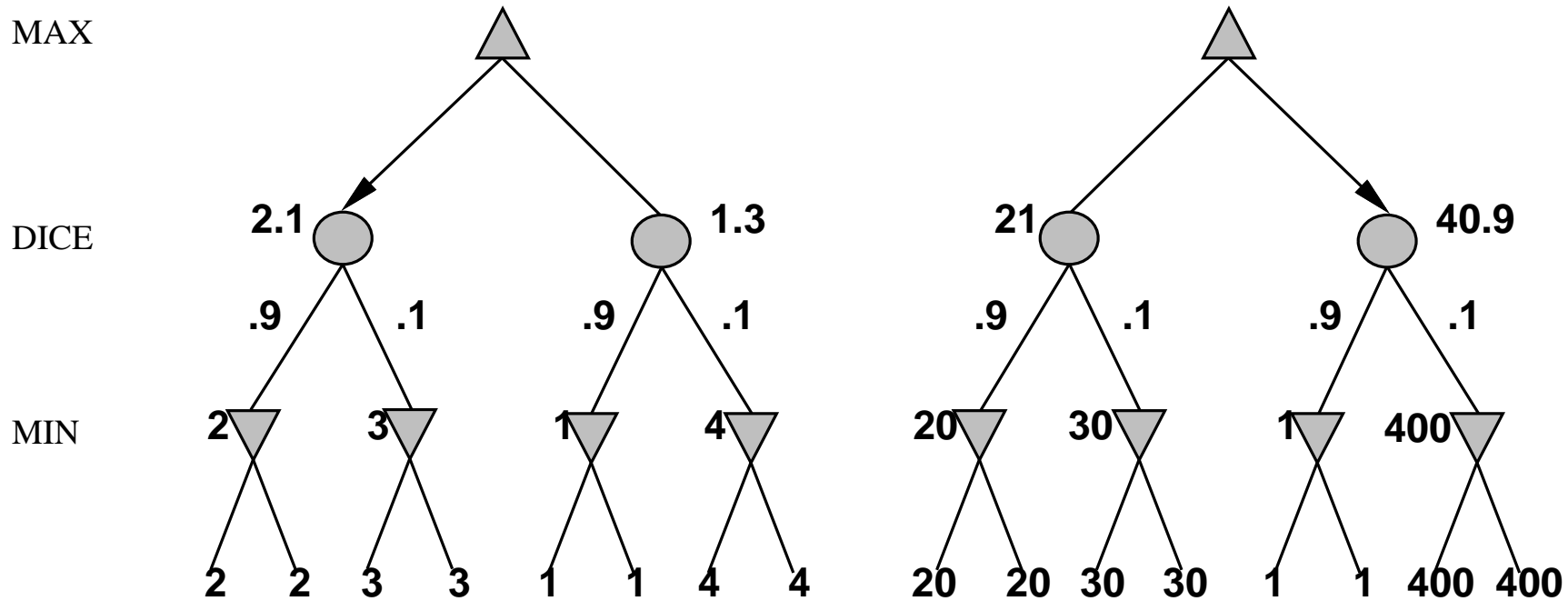
\Rightarrow value of lookahead is diminished

α - β pruning is much less effective

TDGAMMON uses depth-2 search + very good EVAL

\approx world-champion level

Digression: Exact values DO matter



Behaviour is preserved only by *positive linear* transformation of $EVAL$

Hence $EVAL$ should be proportional to the expected payoff

Games of imperfect information

E.g., card games, where opponent's initial cards are unknown

Typically we can calculate a probability for each possible deal

Seems just like having one big dice roll at the beginning of the game*

Idea: compute the minimax value of each action in each deal,
then choose the action with highest expected value over all deals*

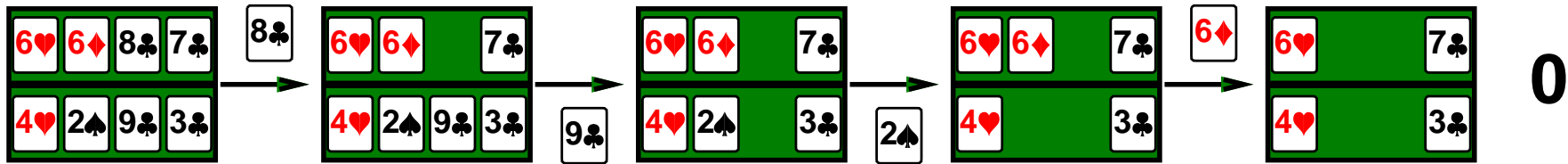
Special case: if an action is optimal for all deals, it's optimal.*

GIB, current best bridge program, approximates this idea by

- 1) generating 100 deals consistent with bidding information
- 2) picking the action that wins most tricks on average

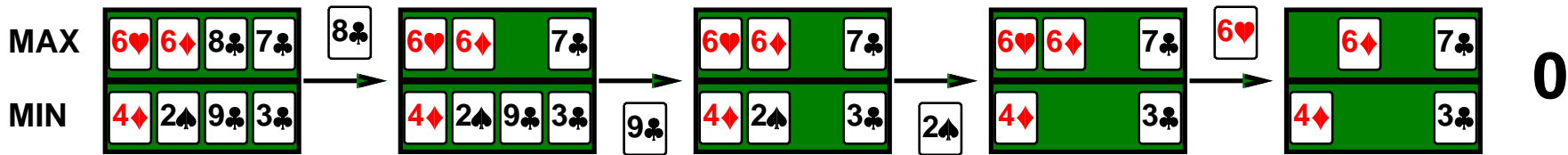
Example

Four-card bridge/whist/hearts hand, MAX to play first



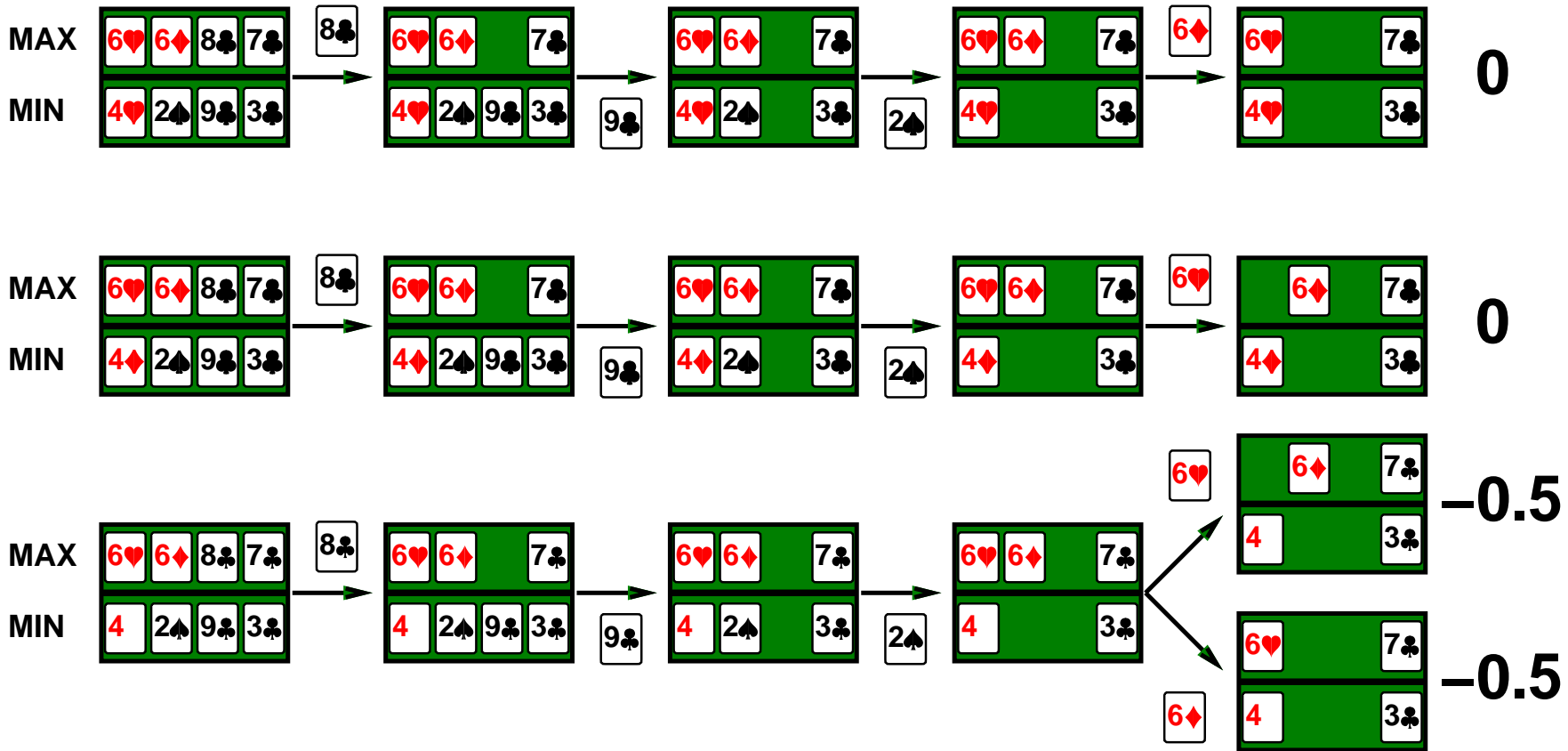
Example

Four-card bridge/whist/hearts hand, MAX to play first



Example

Four-card bridge/whist/hearts hand, MAX to play first



Commonsense example

Road A leads to a small heap of gold pieces

Road B leads to a fork:

take the left fork and you'll find a mound of jewels;

take the right fork and you'll be run over by a bus.

Commonsense example

Road A leads to a small heap of gold pieces

Road B leads to a fork:

take the left fork and you'll find a mound of jewels;

take the right fork and you'll be run over by a bus.

Road A leads to a small heap of gold pieces

Road B leads to a fork:

take the left fork and you'll be run over by a bus;

take the right fork and you'll find a mound of jewels.

Commonsense example

Road A leads to a small heap of gold pieces

Road B leads to a fork:

take the left fork and you'll find a mound of jewels;

take the right fork and you'll be run over by a bus.

Road A leads to a small heap of gold pieces

Road B leads to a fork:

take the left fork and you'll be run over by a bus;

take the right fork and you'll find a mound of jewels.

Road A leads to a small heap of gold pieces

Road B leads to a fork:

guess correctly and you'll find a mound of jewels;

guess incorrectly and you'll be run over by a bus.

Proper analysis

* Intuition that the value of an action is the average of its values in all actual states is WRONG

With partial observability, value of an action depends on the **information state** or **belief state** the agent is in

Can generate and search a tree of information states

Leads to rational behaviors such as

- ◇ Acting to obtain information
- ◇ Signalling to one's partner
- ◇ Acting randomly to minimize information disclosure

Summary

Games are fun to work on! (and dangerous)

They illustrate several important points about AI

- ◇ perfection is unattainable \Rightarrow must approximate
- ◇ good idea to think about what to think about
- ◇ uncertainty constrains the assignment of values to states

Games are to AI as grand prix racing is to automobile design