CS 4700: Foundations of Artificial Intelligence

Spring 2020

Prof. Haym Hirsh

Lecture 8

February 10, 2020

Reminder: Technology Policy

No technology except for first four rows of left and right sides

Jupyter Notebooks Postponed

"Primer": ~~Friday 5pm in Gates G01~~

Postponed – TBA

Covers:

Jupyter Notebooks and .ipynb files

Google Colab (online Jupyter Notebook environment)

Useful Python features

Bring your laptop

This Friday's Lecture Preponed

To be given

Wednesday, February 12 3:30-4:30

Gates G01

Will be recorded

and available on course website

(must watch between Wed and following Mon)

This Friday's Lecture Preponed

To be given

Wednesday, February 12 2:30-3:30

Gates G01

Will be recorded

and available on course website

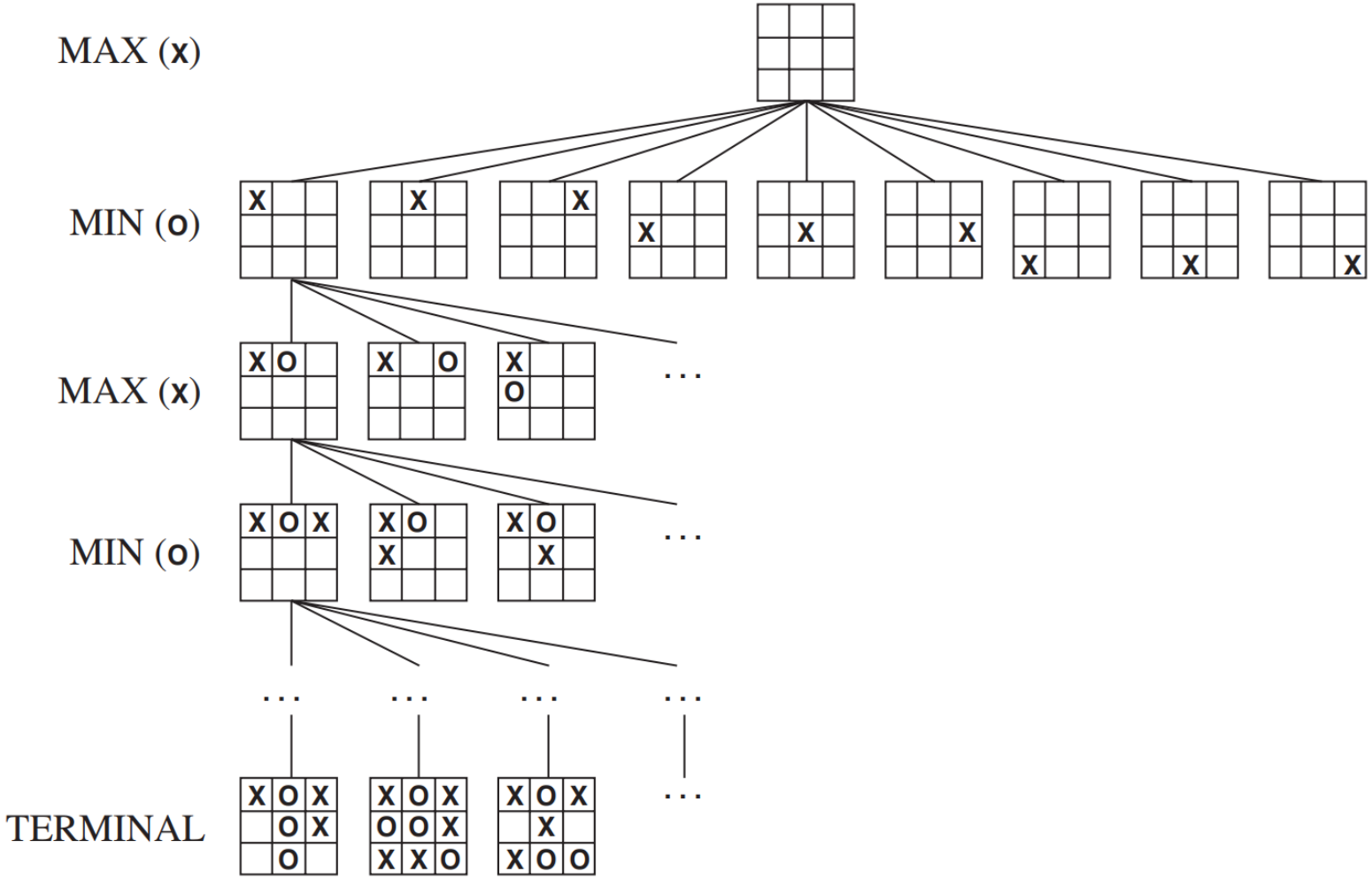
(must watch between Wed and following Mon)

(Please come to the live version!)

Recap

Game tree concept

Tic Tac Toe



Recap

Game tree concept

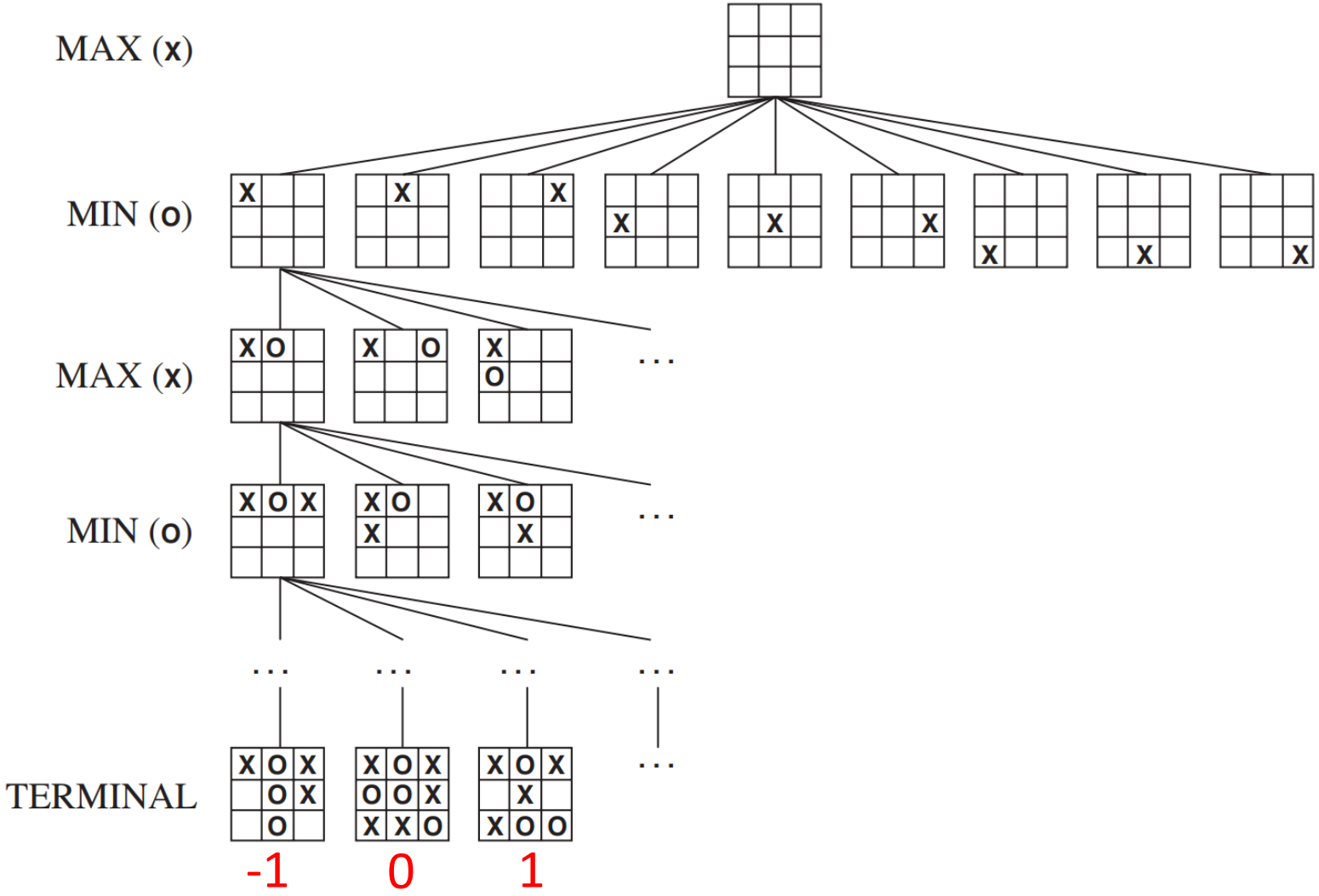
Recap

Game tree concept

Terminal nodes have “ground truth” values

(examples: $\{+1, 0, -1\}$, $\{+\infty, -\infty\}$)

Tic Tac Toe



Recap

Game tree concept

Terminal nodes have “ground truth” values

(examples: $\{+1, 0, -1\}$, $\{+\infty, -\infty\}$)

Recap

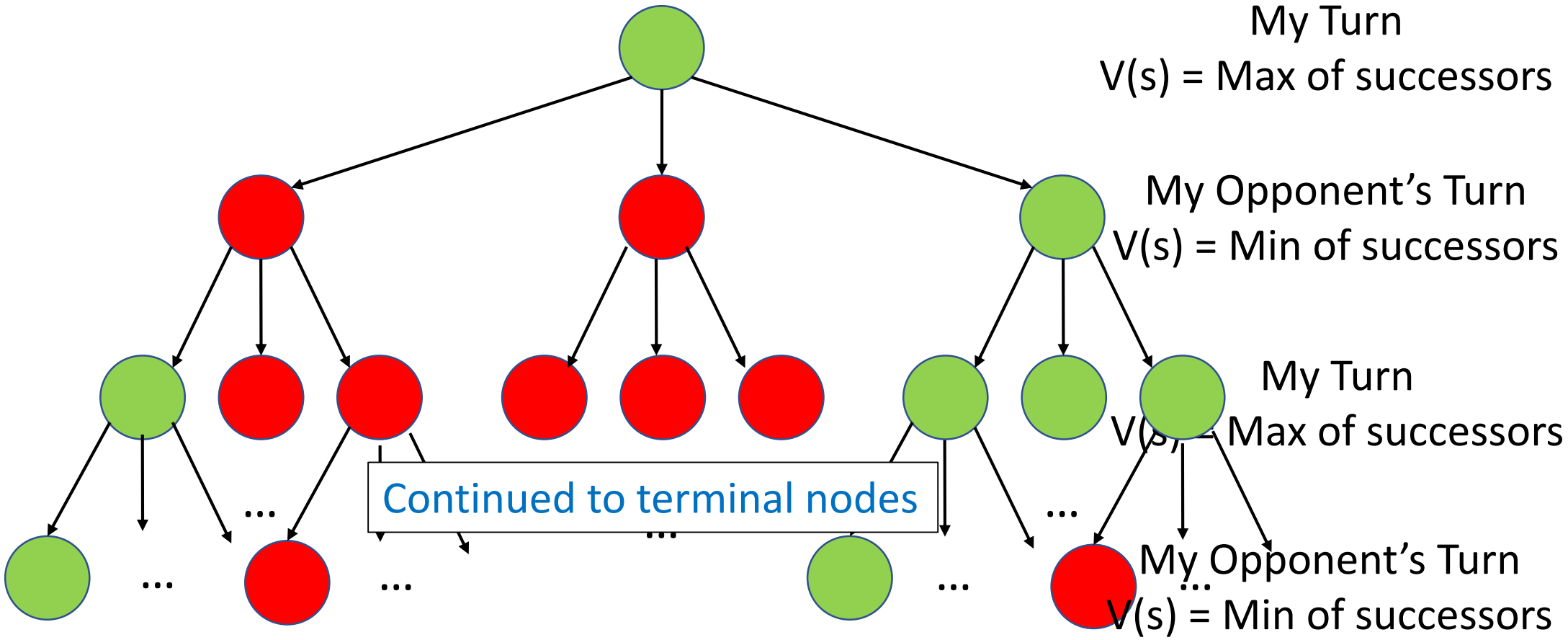
Game tree concept

Terminal nodes have “ground truth” values

(examples: $\{+1, 0, -1\}$, $\{+\infty, -\infty\}$)

Internal nodes have values based on values of descendants

Minimax Value of a Game



Recap

Game tree concept

Terminal nodes have “ground truth” values

(examples: $\{+1, 0, -1\}$, $\{+\infty, -\infty\}$)

Internal nodes have values based on values of descendants

Recap

Game tree concept

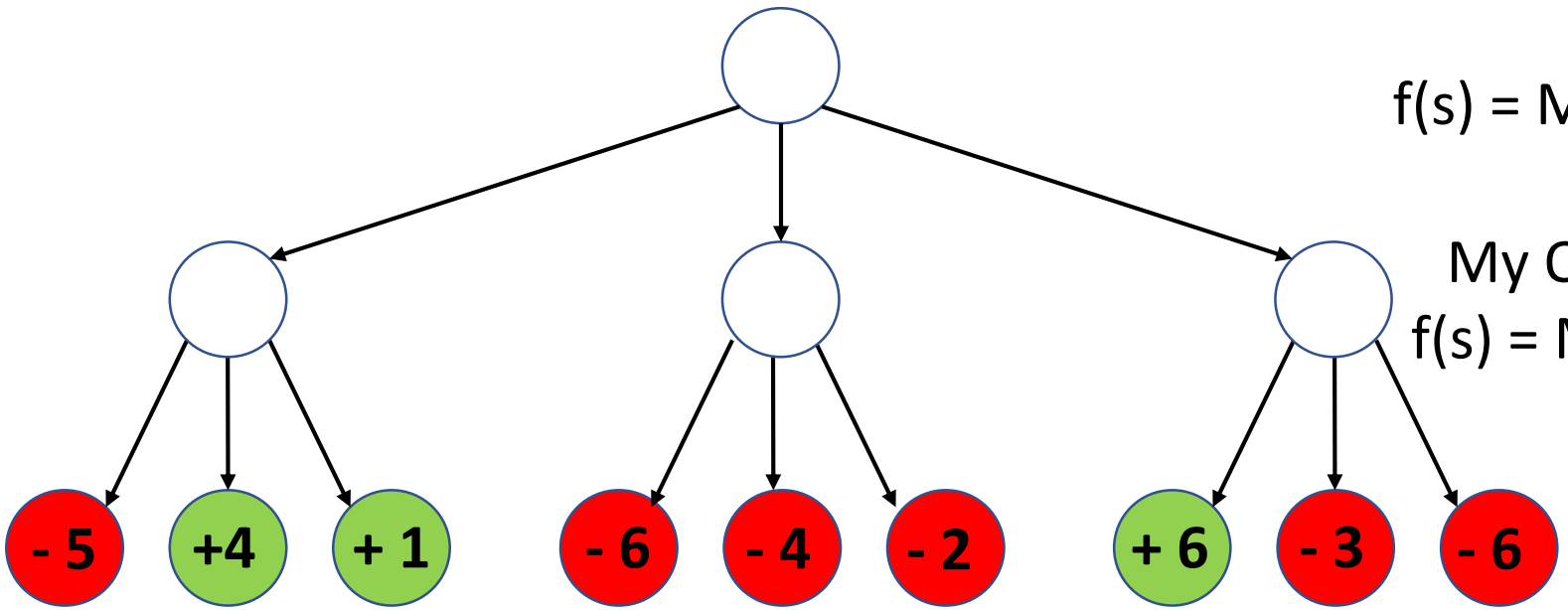
Terminal nodes have “ground truth” values

(examples: $\{+1, 0, -1\}$, $\{+\infty, -\infty\}$)

Internal nodes have values based on values of descendants

Can stop early and use heuristic evaluation functions

Minimax Value of a Game

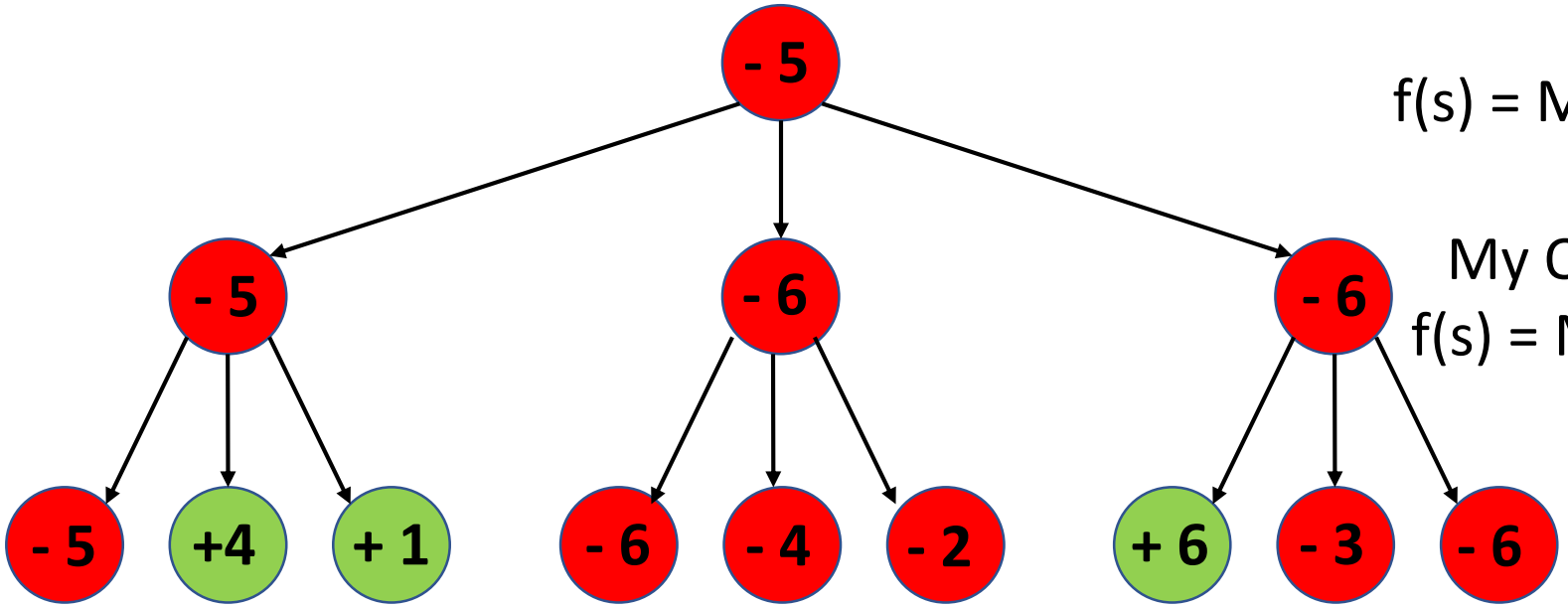


My Turn
f(s) = Max of successors

My Opponent's Turn
f(s) = Min of successors

Use heuristic evaluation function

Minimax Value of a Game



My Turn
f(s) = Max of successors

My Opponent's Turn
f(s) = Min of successors

Use heuristic evaluation function

Recap

Game tree concept

Terminal nodes have “ground truth” values

(examples: $\{+1, 0, -1\}$, $\{+\infty, -\infty\}$)

Internal nodes have values based on values of descendants

Can stop early and use heuristic evaluation functions

Recap

Game tree concept

Terminal nodes have “ground truth” values

(examples: {+1, 0, -1}, {+∞, -∞})

Internal nodes have values based on values of descendants

Can stop early and use heuristic evaluation functions

“Minimax Search”

Minimax Algorithm

```
maxturn(s,ops):
    if cutoff(s) then return f(s)
    else
        val ← -∞;
        foreach o ∈ ops
            val' ← minturn(apply(s,o),ops);
            if val' > val then
                val ← val';
                bestop ← o;
        return val
```

```
minturn(s,ops):
    if cutoff(s) then return f(s)
    else
        val ← +∞;
        foreach o ∈ ops
            val' ← maxturn(apply(s,o),ops);
            if val' < val then
                val ← val';
                bestop ← o;
        return val
```

Initial call:

- If I go first: maxturn(initial-state,ops)
- If opponent goes first: minturn(initial-state,ops)

cutoff(s) could be a depth-bound or something more sophisticated

Recap

Game tree concept

Terminal nodes have “ground truth” values

(examples: {+1, 0, -1}, {+∞, -∞})

Internal nodes have values based on values of descendants

Can stop early and use heuristic evaluation functions

“Minimax Search”

Recap

Game tree concept

Terminal nodes have “ground truth” values

(examples: {+1, 0, -1}, {+∞, -∞})

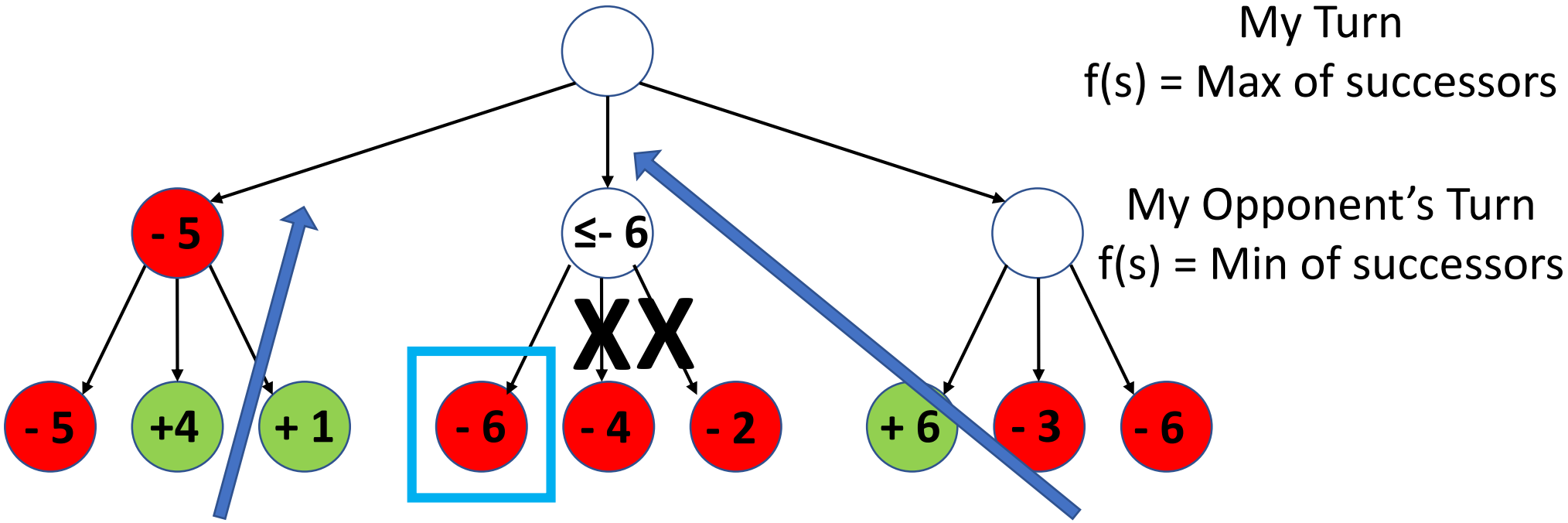
Internal nodes have values based on values of descendants

Can stop early and use heuristic evaluation functions

“Minimax Search”

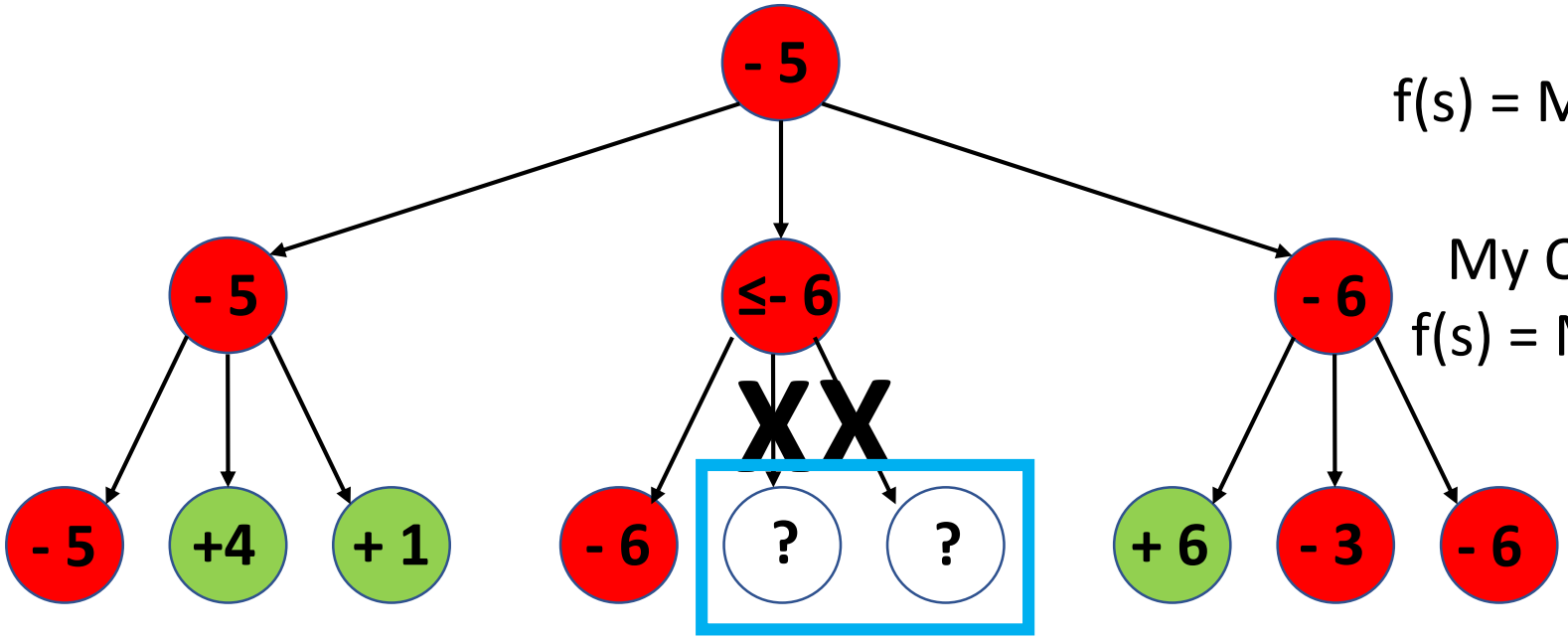
Can avoid needless effort: Alpha-Beta Pruning

Alpha-Beta Pruning



This is <u>guaranteed</u> to be a better move than this
regardless of what results from the other actions

Alpha-Beta Pruning

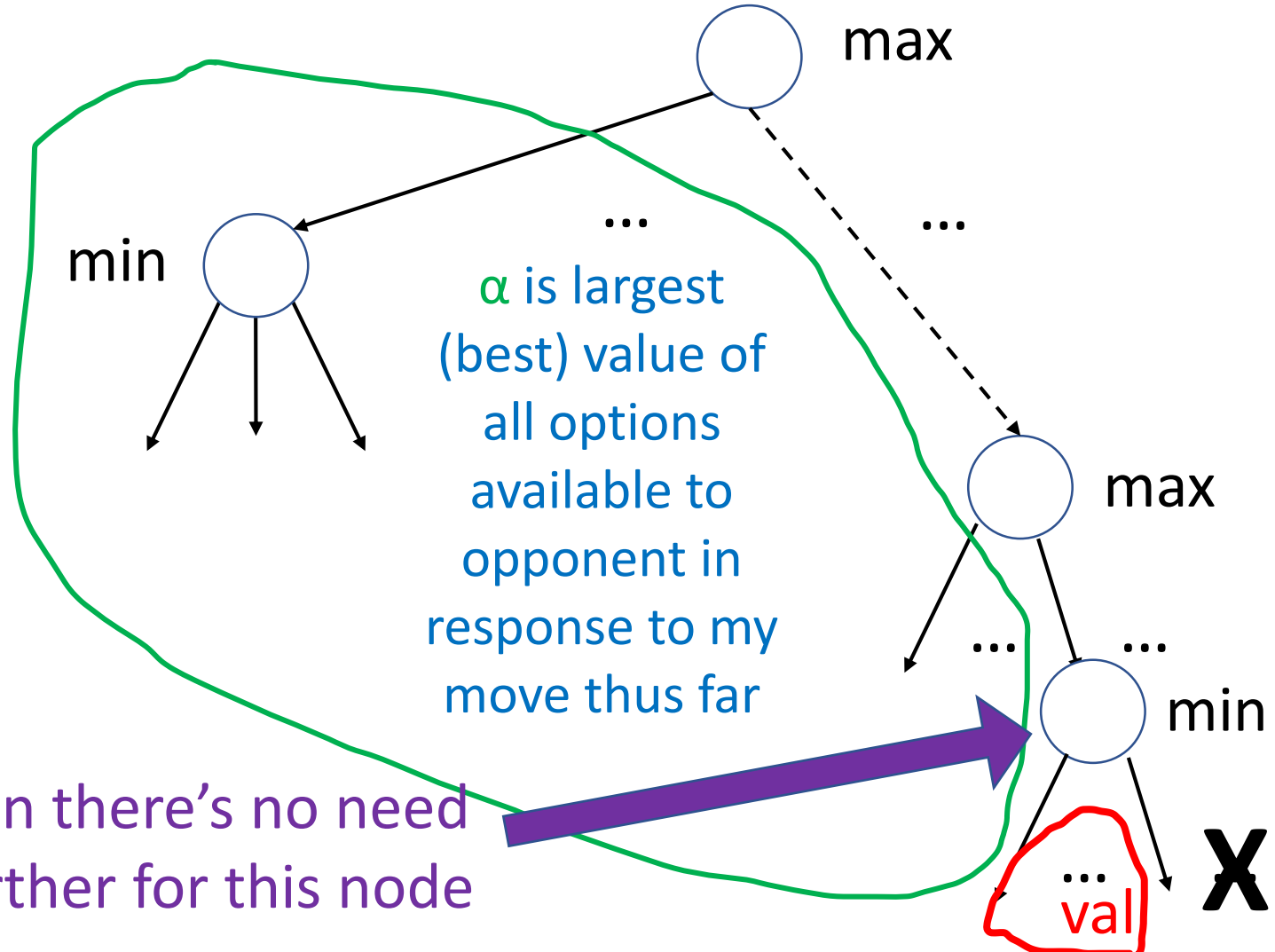


My Turn
f(s) = Max of successors

My Opponent's Turn
f(s) = Min of successors

There is no way to set a value to either of these two states that would change the outcome

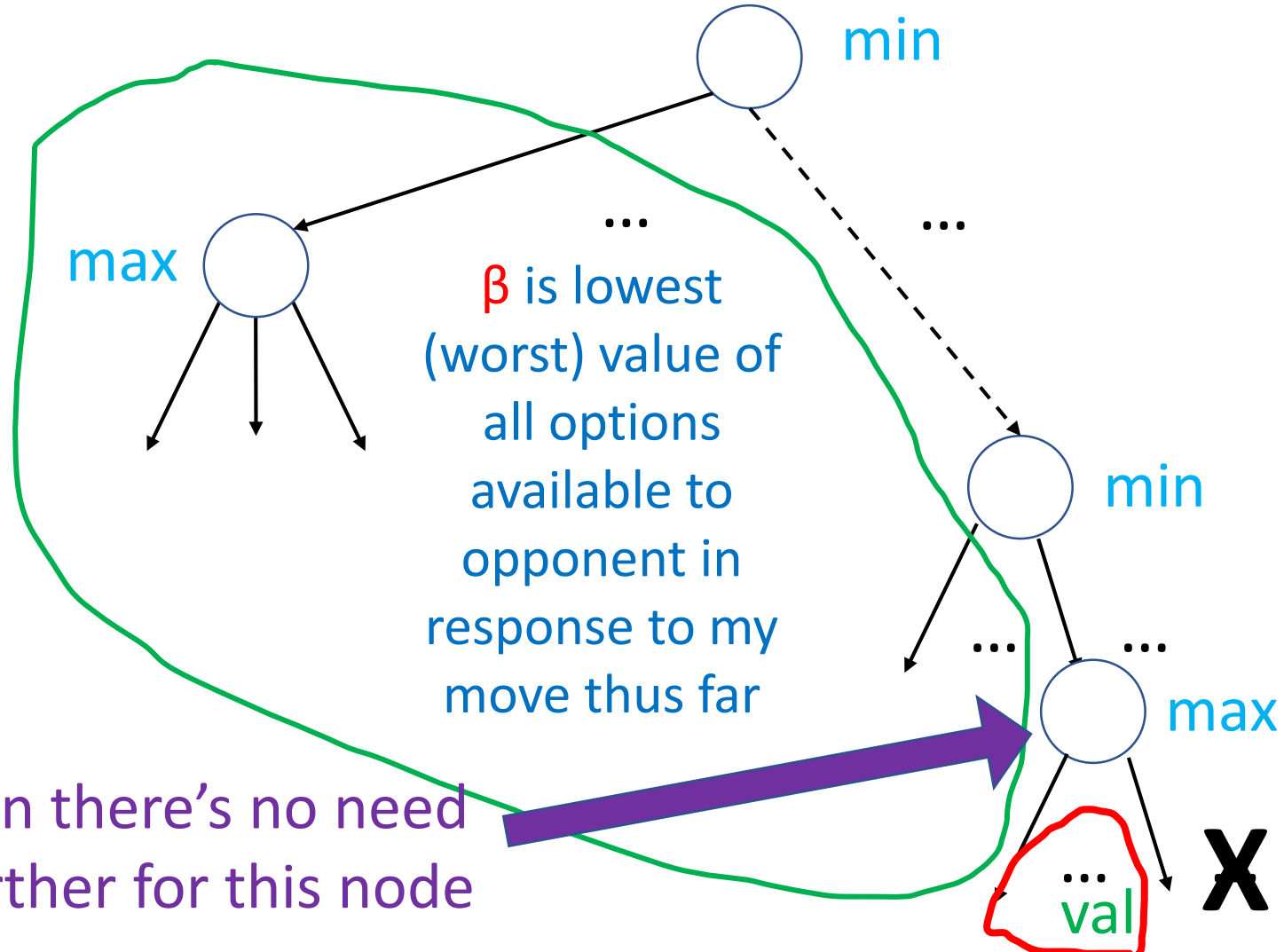
Alpha-Beta Pruning: General Case



If α ≥ val then there's no need to search further for this node

val is lowest (worst) value of all options available to opponent in response to my move thus far

Alpha-Beta Pruning: General Case



If β ≤ val then there's no need to search further for this node

val is biggest (best) value of all options available to opponent in response to my move thus far

Minimax Algorithm with Alpha-Beta Pruning

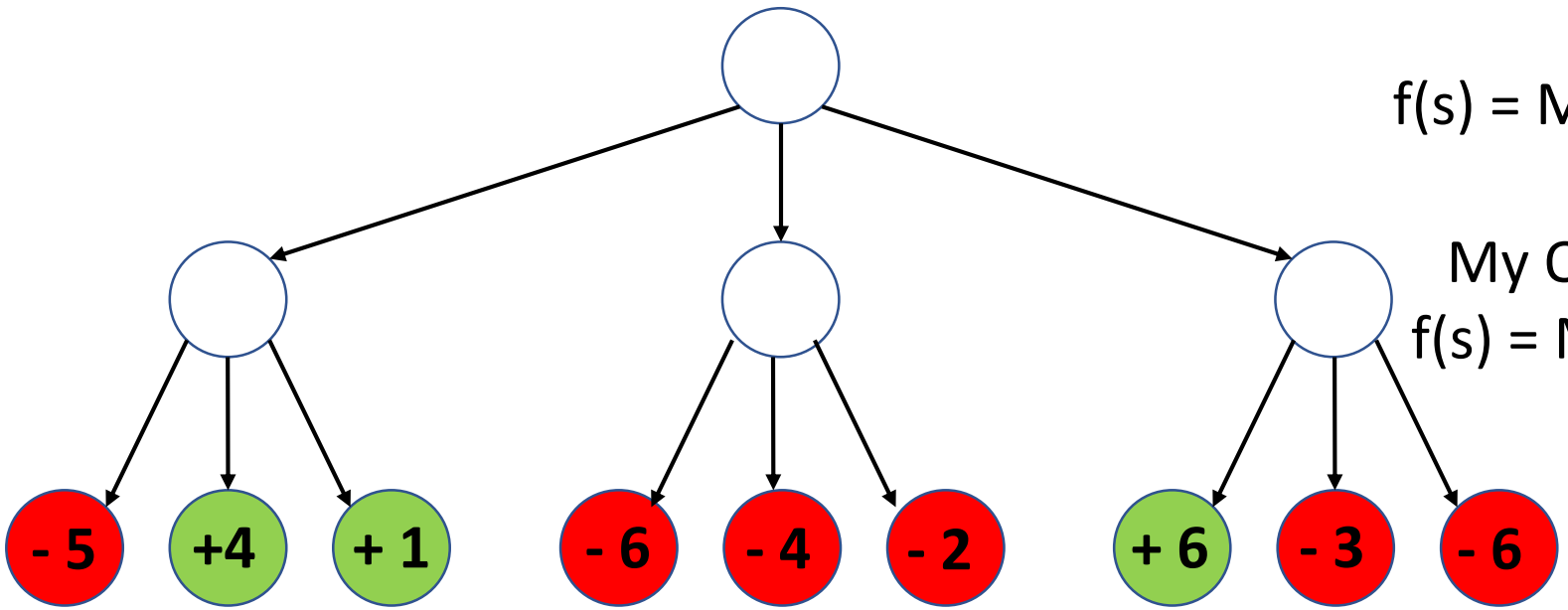
```
maxturn(s,ops,a,b):     {my turn}
    if cutoff(s,depth) then return f(s)
    else
        val ← -∞;
        foreach o ∈ ops
            val' ← minturn(apply(s,o),ops,a,b);
            if val' > val then
                val ← val';
                bestop ← o;
            if val ≥ b then return val;
            a ← max(a,val)
        return val
```

```
minturn(s,ops,a,b):     {opponent's turn}
    if cutoff(s,depth) then return f(s)
    else
        val ← +∞;
        foreach o ∈ ops
            val' ← maxturn(apply(s,o),ops,a,b);
            if val' < val then
                val ← val';
                bestop ← o;
            if val ≤ a then return val;
            b ← min(b,val)
        return val
```

Initial call:

- If I go first: maxturn(initial-state,ops,-∞,+∞)
- If opponent goes first: minturn(initial-state,ops,-∞,+∞)

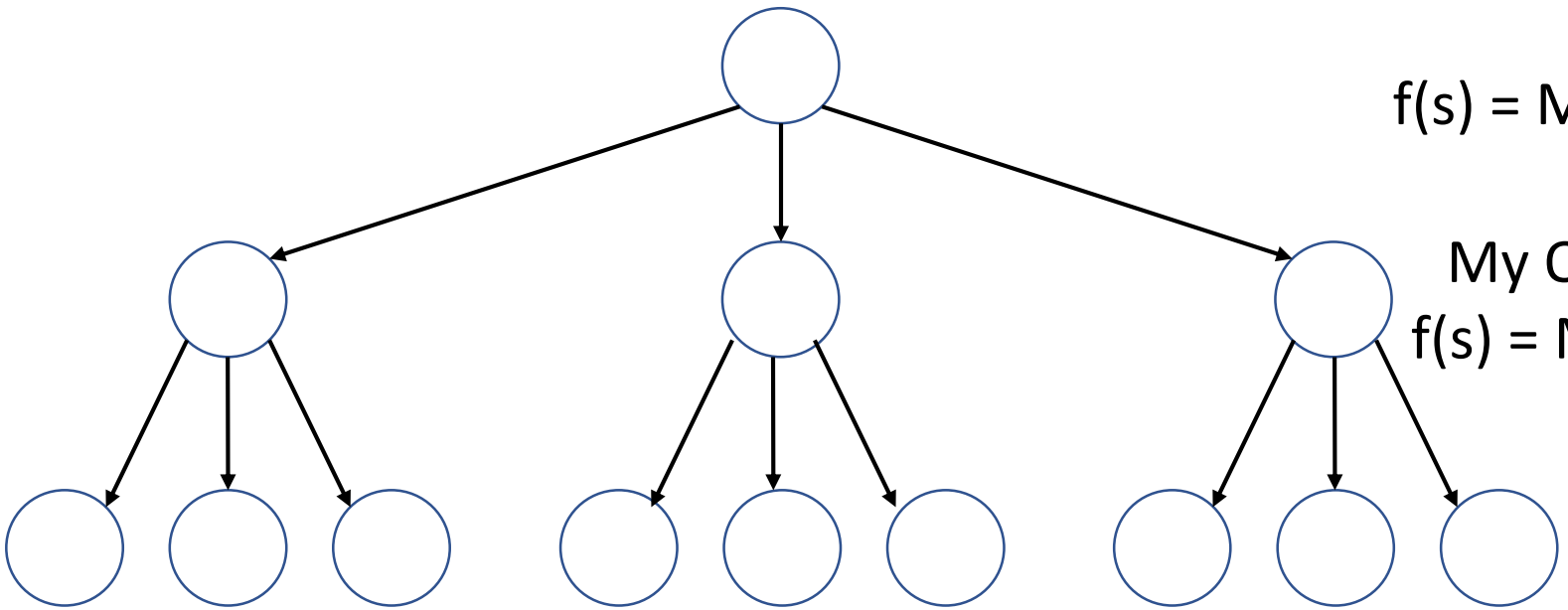
Alpha-Beta Pruning: Implementation



My Turn
f(s) = Max of successors

My Opponent's Turn
f(s) = Min of successors

Alpha-Beta Pruning: Implementation



My Turn
f(s) = Max of successors

My Opponent's Turn
f(s) = Min of successors