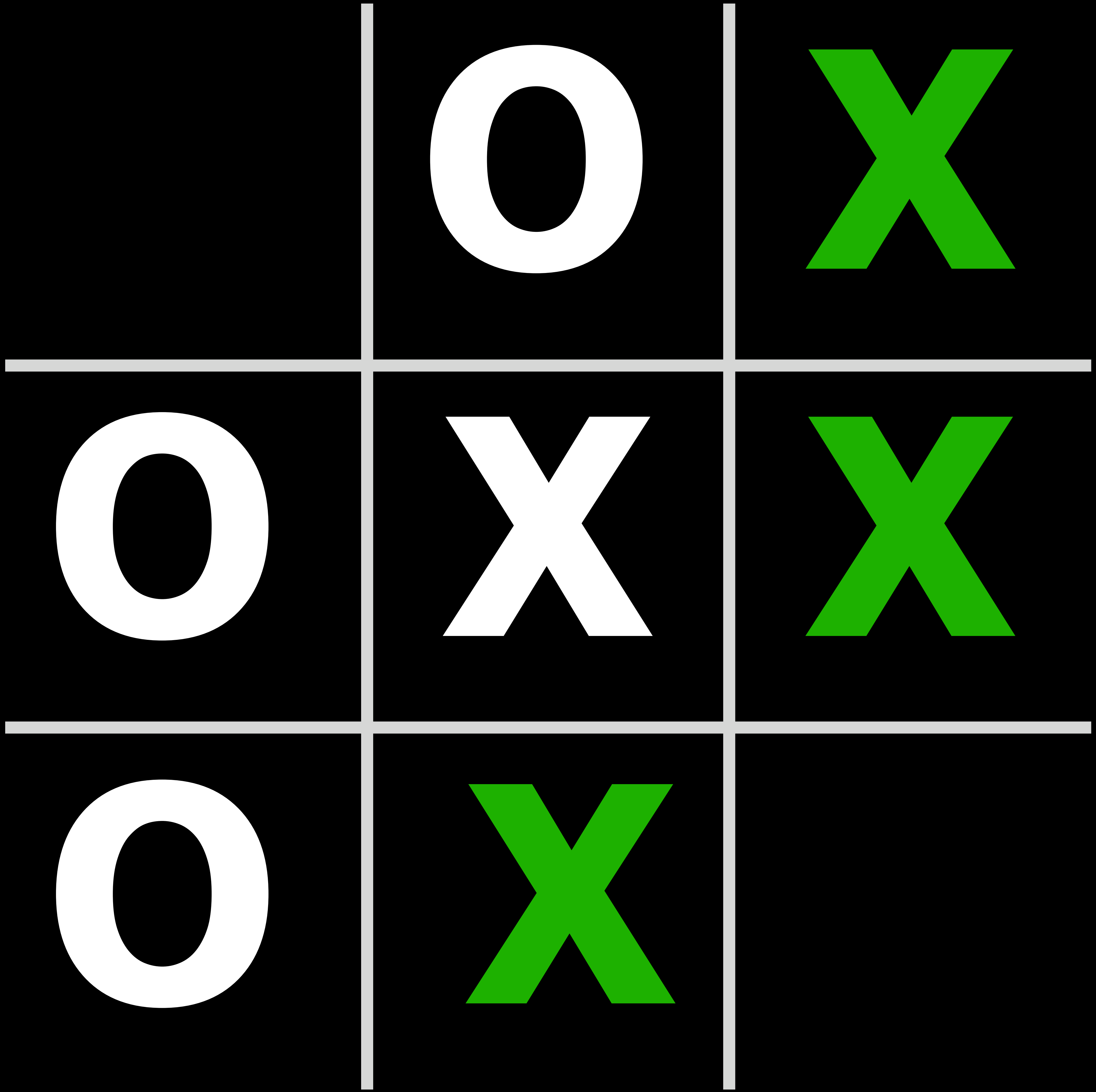
Introduction to
Artificial Intelligence
with Python

Adversarial Search

	O	X
O	X	X
O	X	


O X
O X X
O X

Minimax

O	X	X
O	O	
O	X	X

-1

X	O	X
O	O	X
X	X	O

0

O		X
	X	O
X	O	X

1

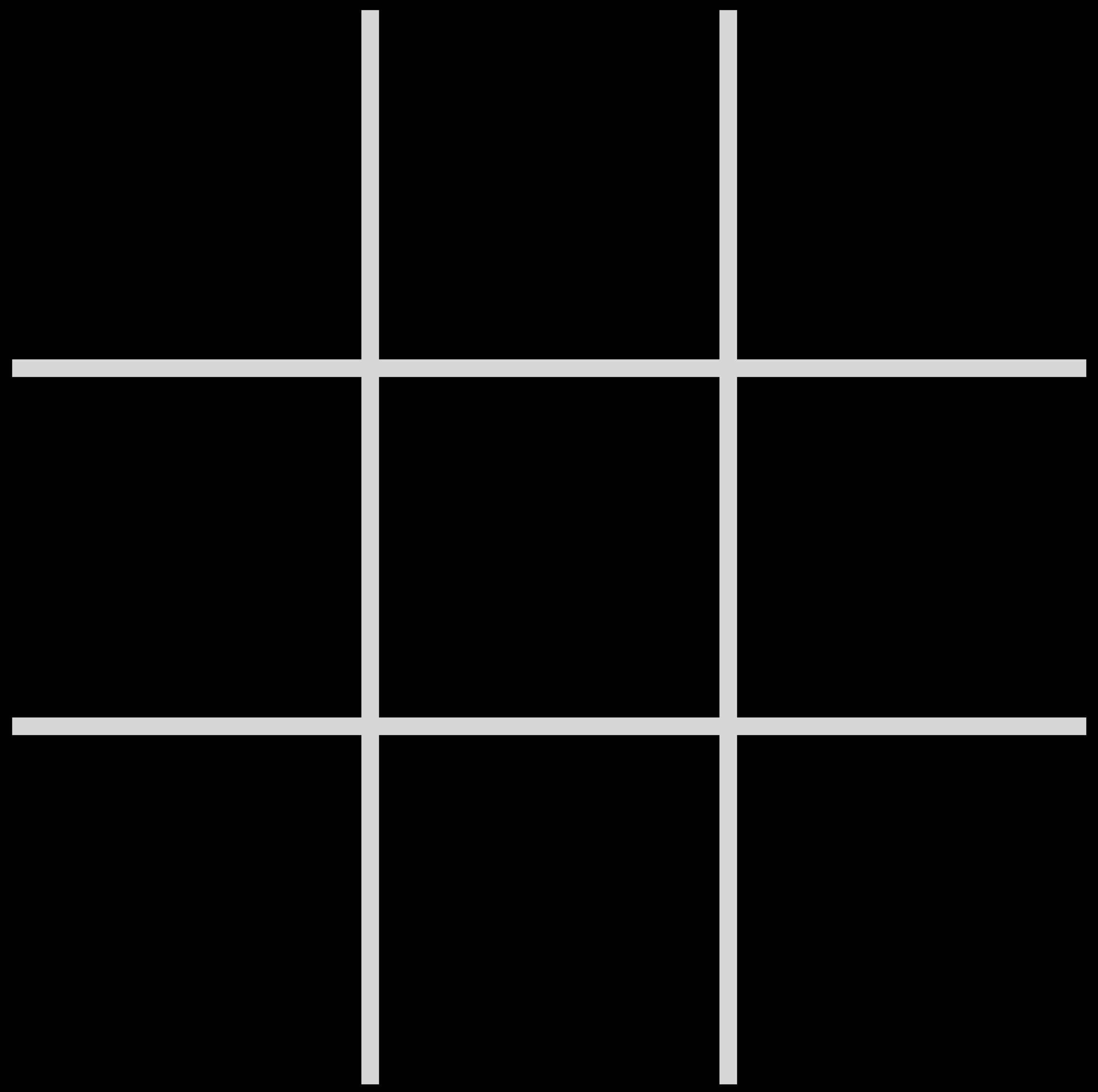
Minimax

- MAX (X) aims to maximize score.
- MIN (O) aims to minimize score.

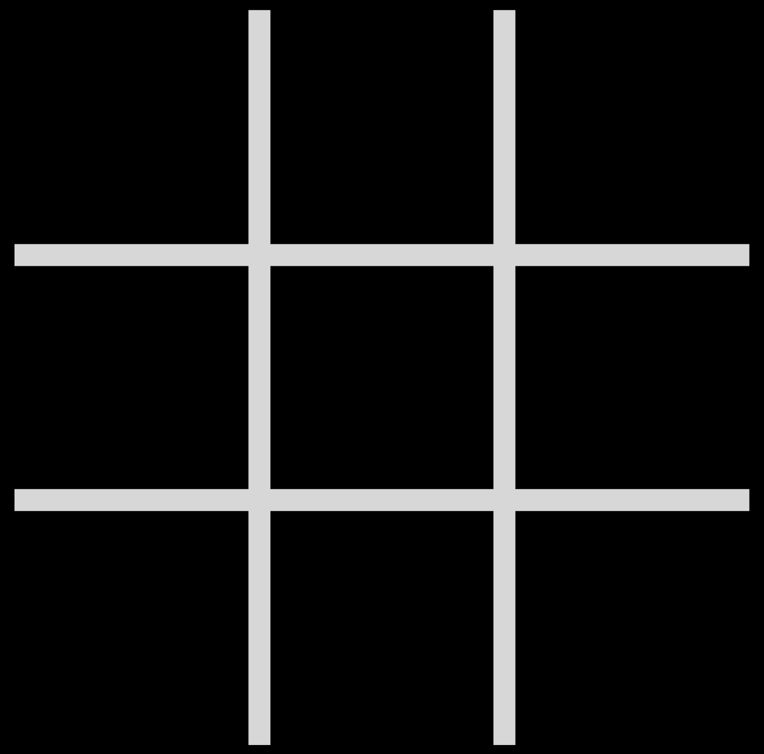
Game

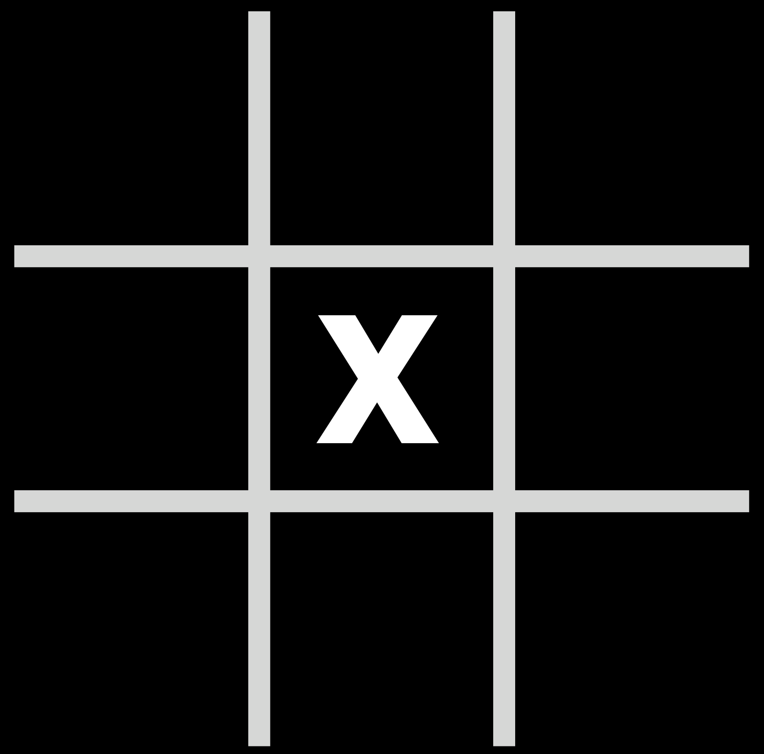
- S_0 : initial state
- $\text{PLAYER}(s)$: returns which player to move in state s
- $\text{ACTIONS}(s)$: returns legal moves in state s
- $\text{RESULT}(s, a)$: returns state after action a taken in state s
- $\text{TERMINAL}(s)$: checks if state s is a terminal state
- $\text{UTILITY}(s)$: final numerical value for terminal state s

Initial State



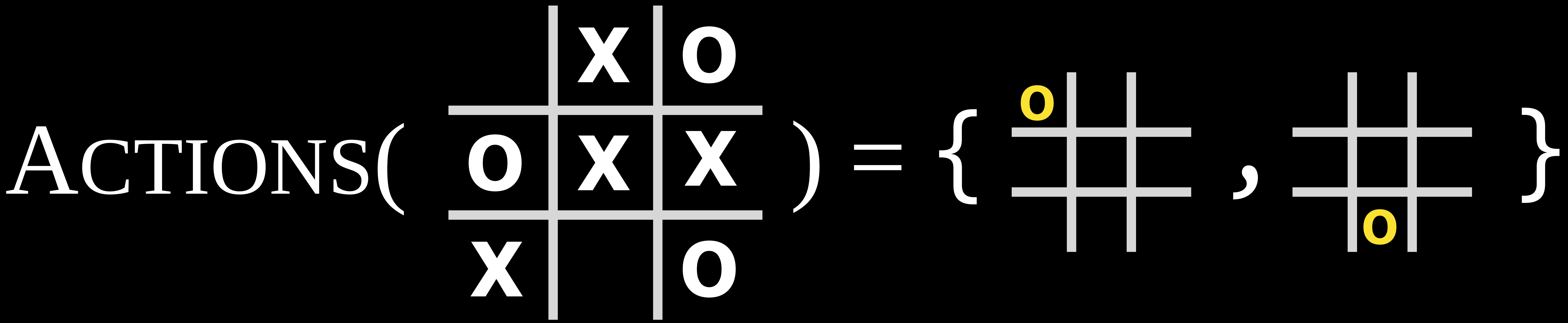
Player(s)

Player() = **X**

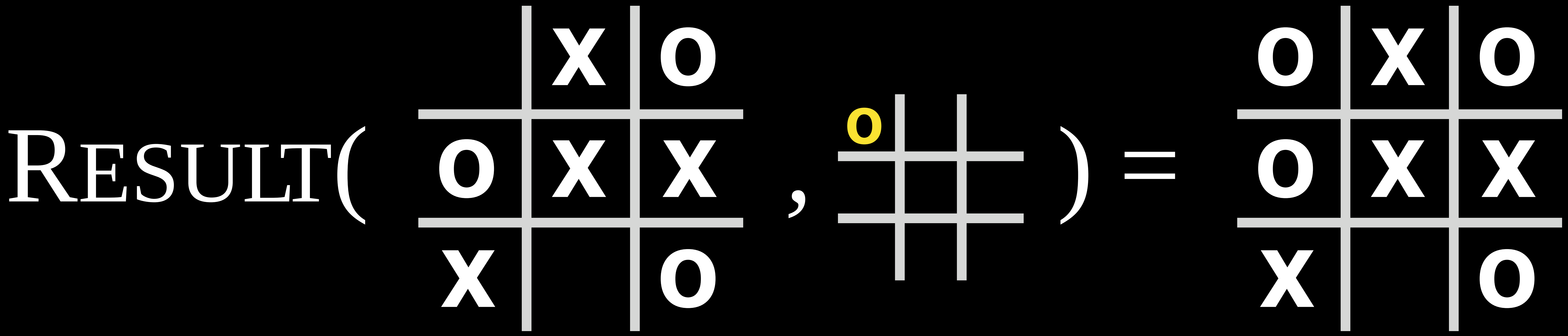
Player(

) = **O**

ACTIONS(s)



$\textsc{Result}(s, a)$



Terminal(s)

Terminal(

O		
O	X	
X	O	X

) = false

Terminal(

O		X
O	X	
X	O	X

) = true

UTILITY(s)

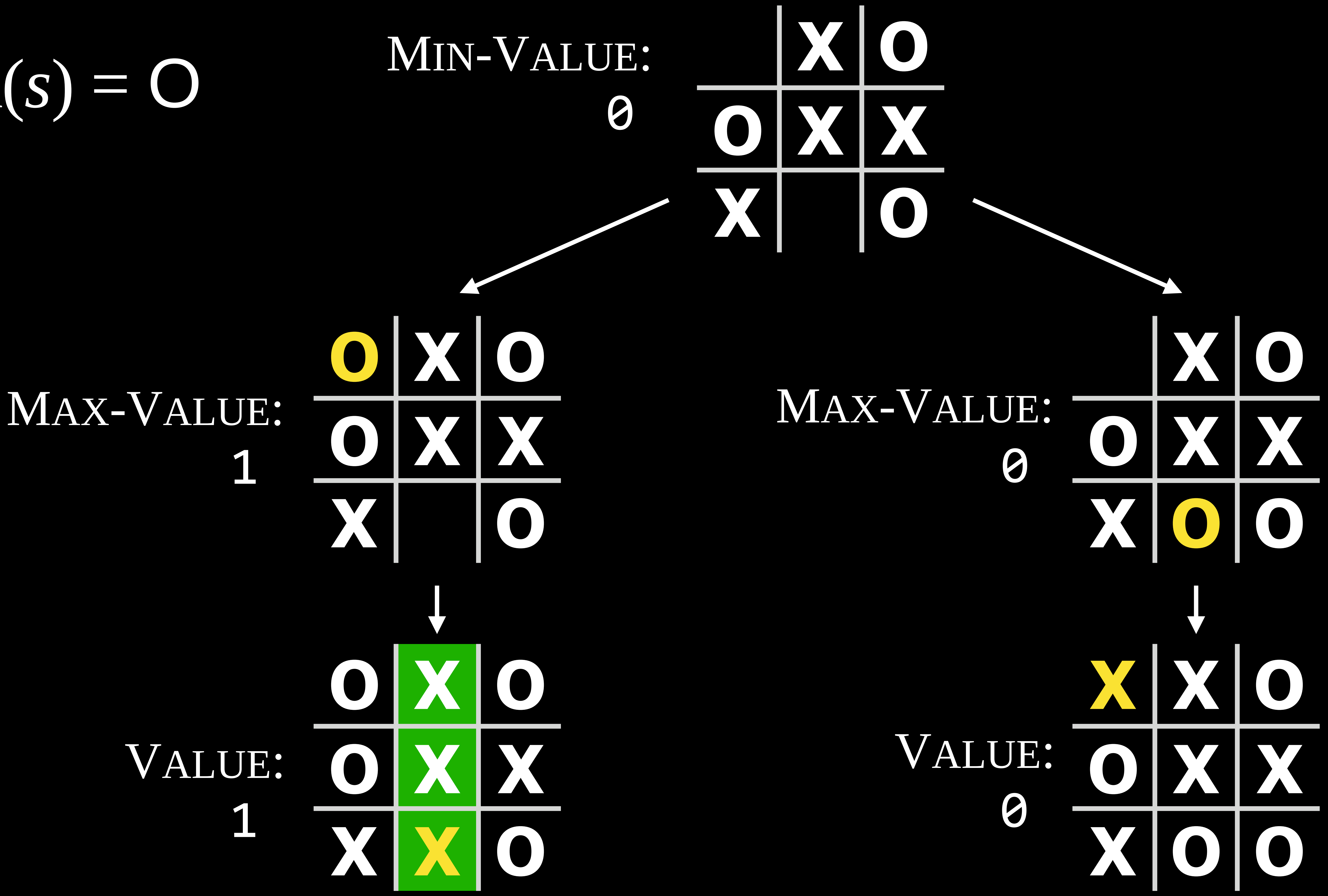
$$\textsc{Utility}\left(\begin{array}{c|c|c} \text{O} & & \text{X} \\ \hline \text{O} & \text{X} & \\ \hline \text{X} & \text{O} & \text{X} \end{array}\right) = 1$$

$$\textsc{Utility}\left(\begin{array}{c|c|c} \text{O} & \text{X} & \text{X} \\ \hline \text{X} & \text{O} & \\ \hline \text{O} & \text{X} & \text{O} \end{array}\right) = -1$$

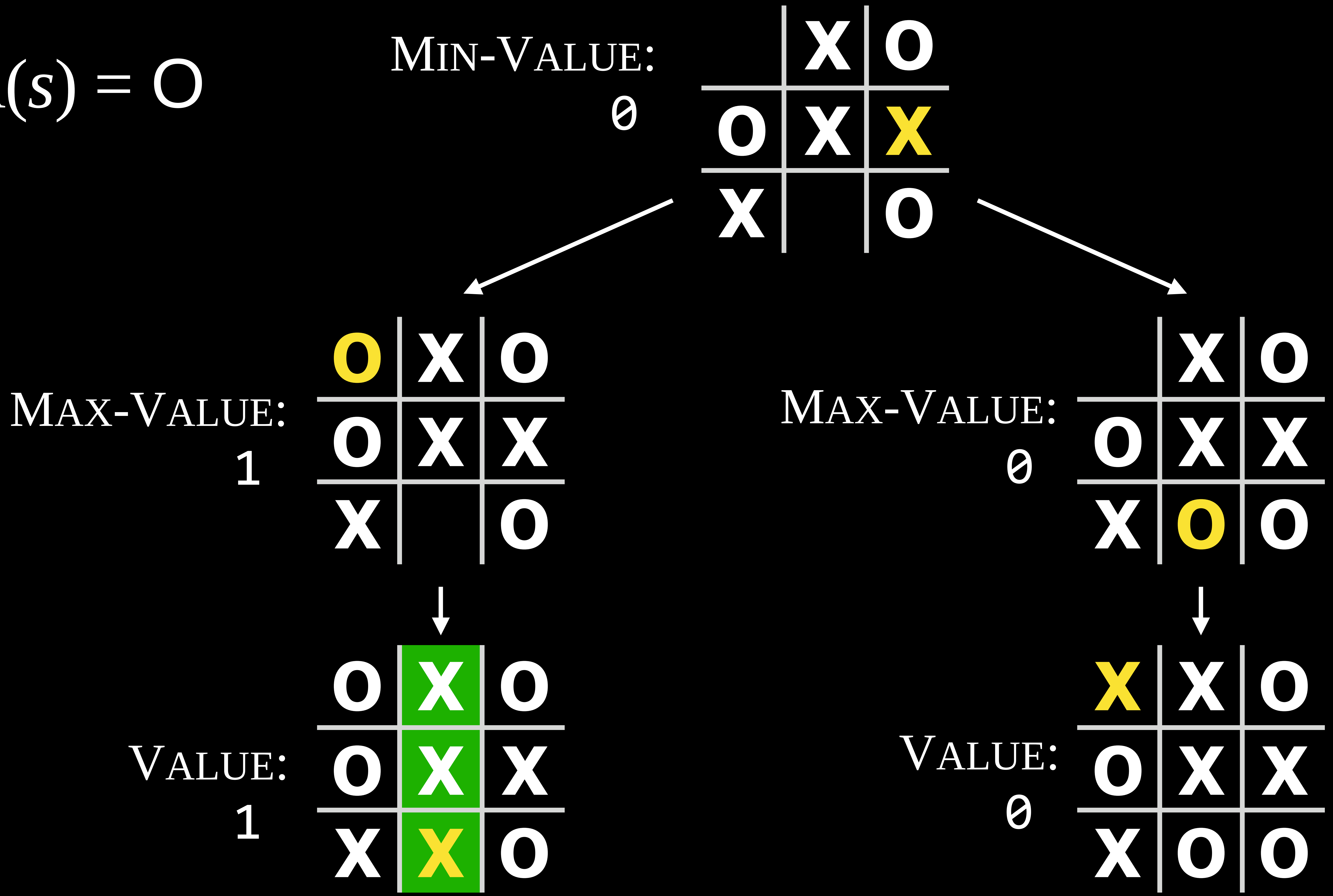
O	X	O
O	X	X
X	X	O

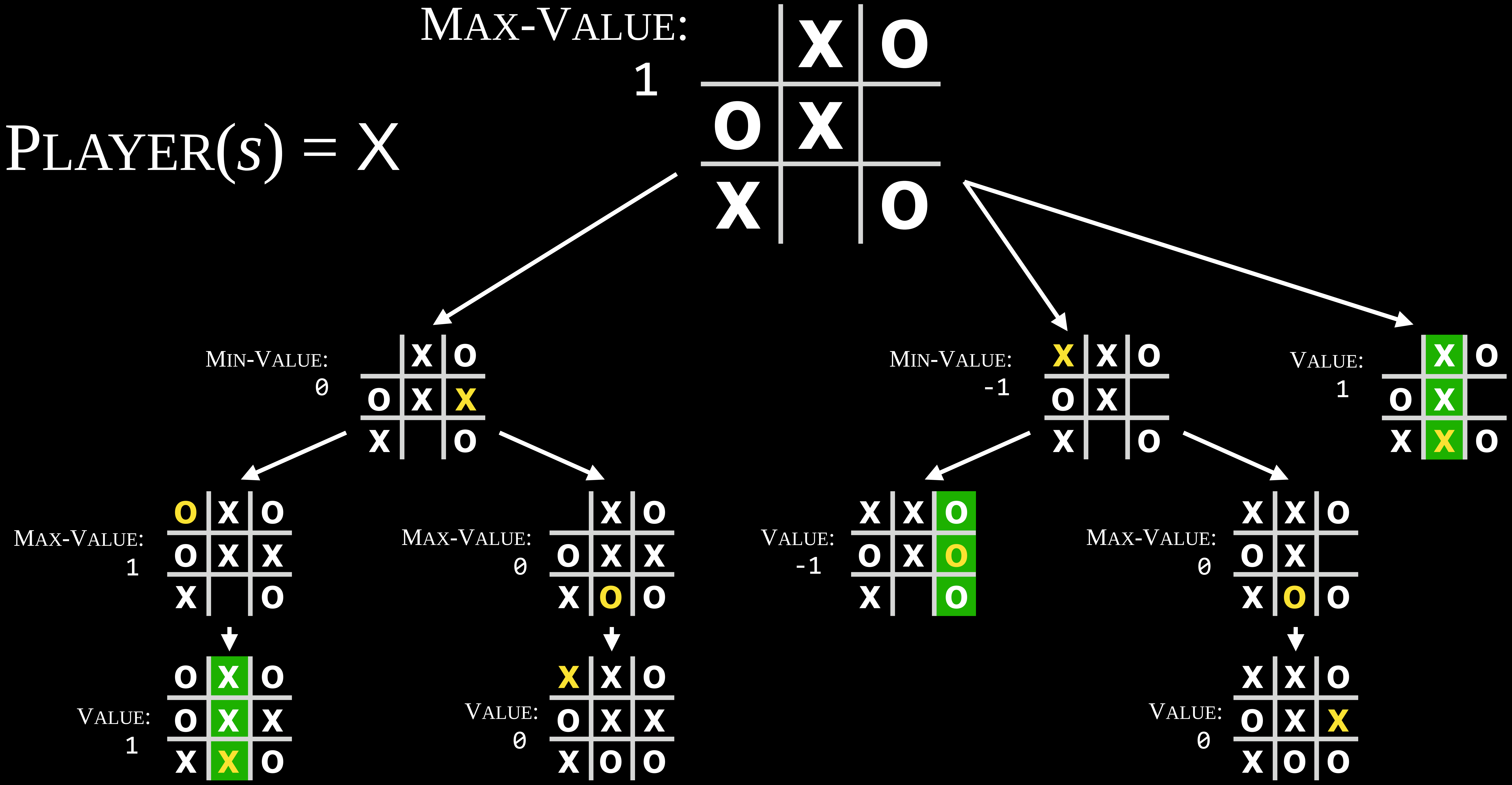
VALUE: 1

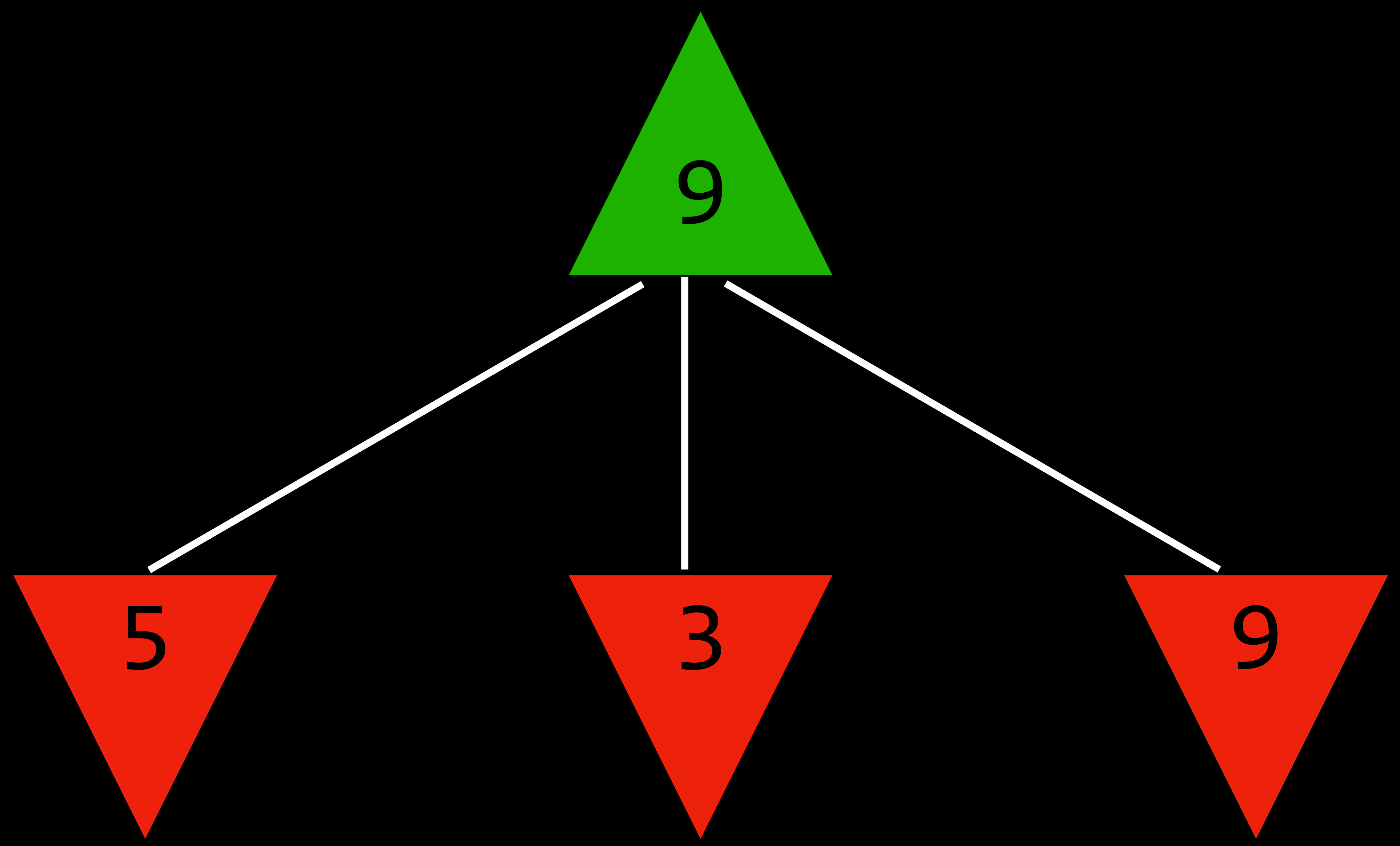
PLAYER(s) = O

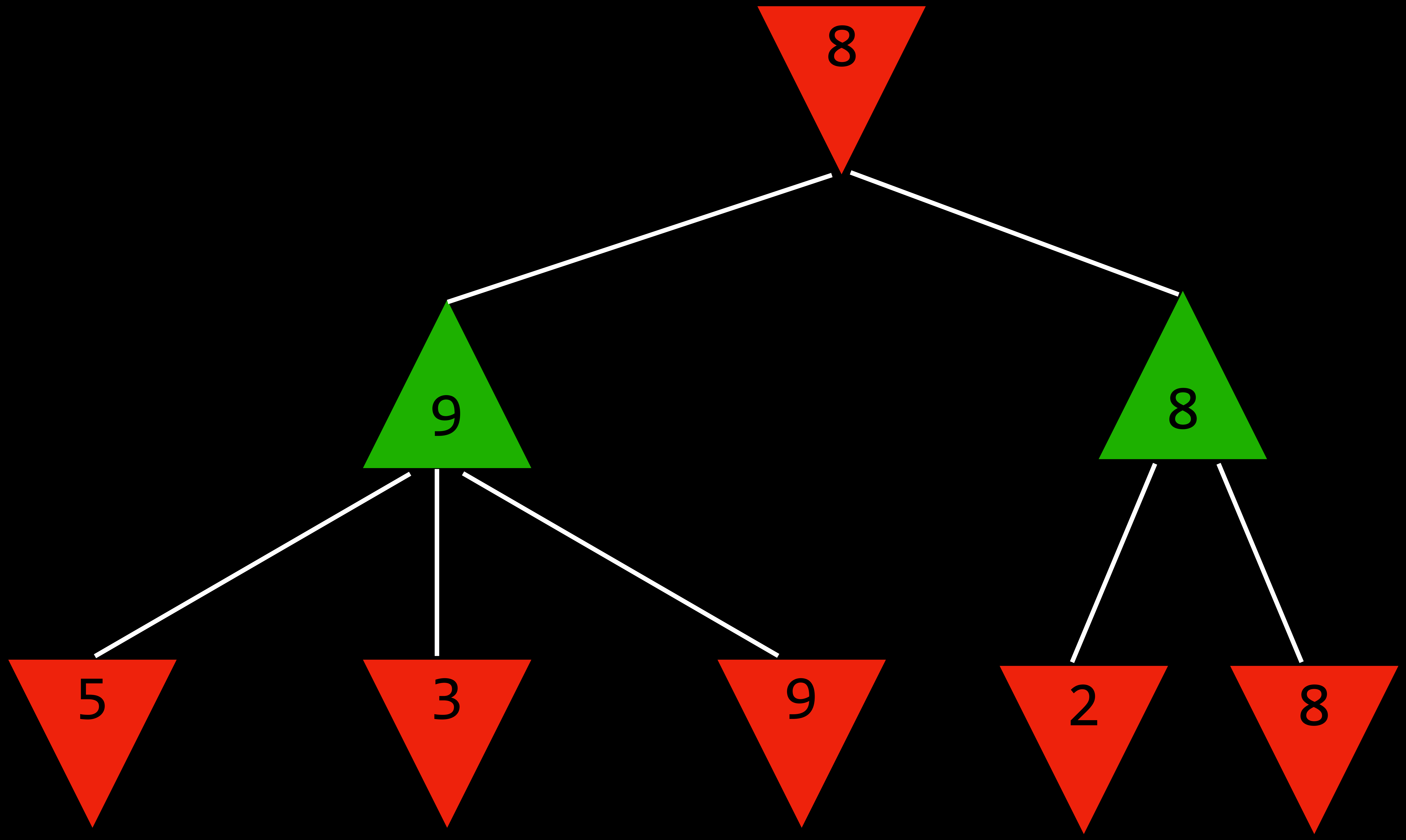


PLAYER(s) = O




MAX-VALUE:
1
PLAYER(s) = X
MIN-VALUE:
0
MIN-VALUE:
-1
VALUE:
1
MAX-VALUE:
1
MAX-VALUE:
0
VALUE:
-1
MAX-VALUE:
0
VALUE:
1
VALUE:
0
VALUE:
0


9
5
3
9


8
9
8
5
3
9
2
8

Minimax

- Given a state s:
  - MAX picks action a in $\textsc{Actions}(s)$ that produces highest value of $\textsc{Min-Value}(\textsc{Result}(s, a))$
  - MIN picks action a in $\textsc{Actions}(s)$ that produces smallest value of $\textsc{Max-Value}(\textsc{Result}(s, a))$

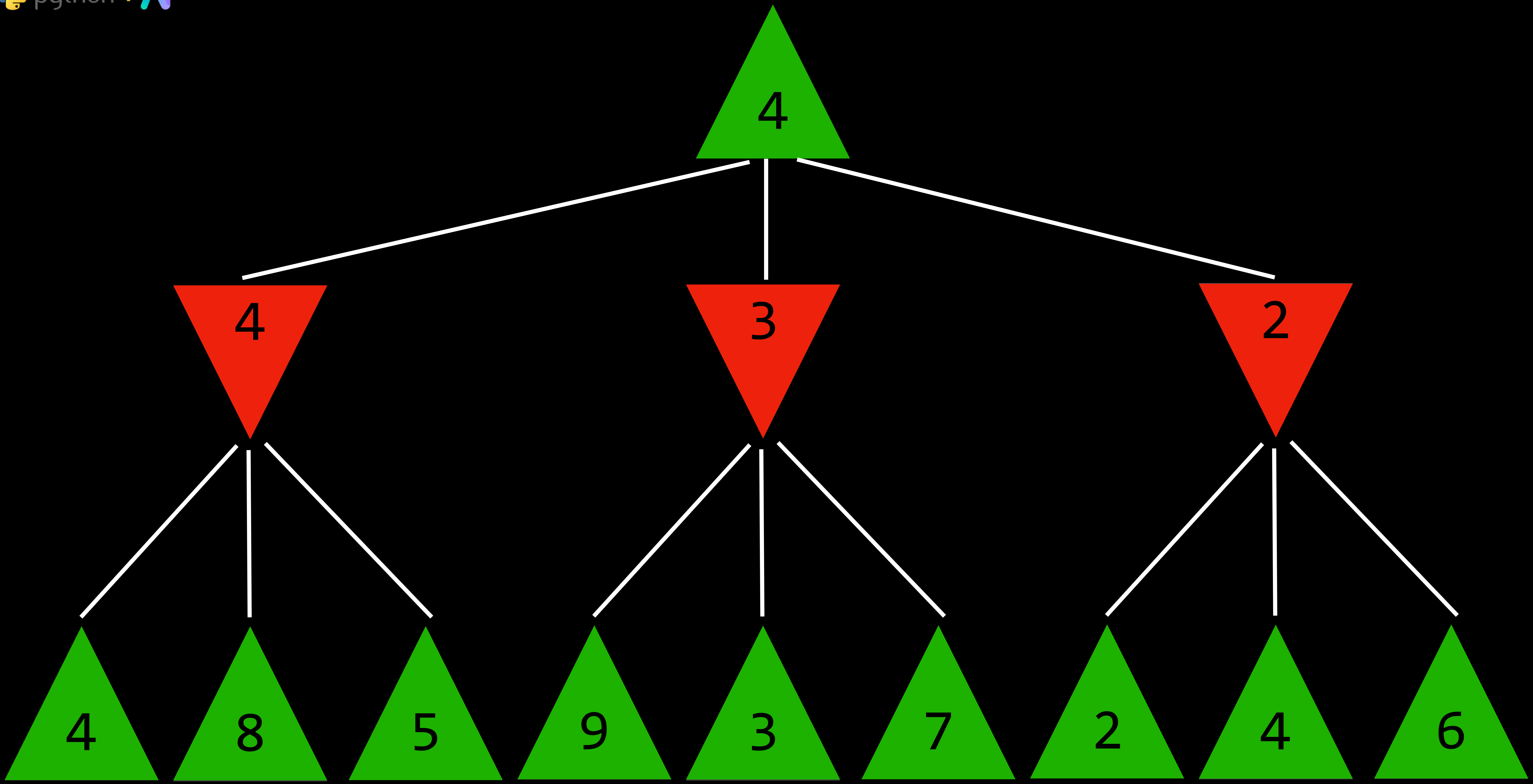
Minimax

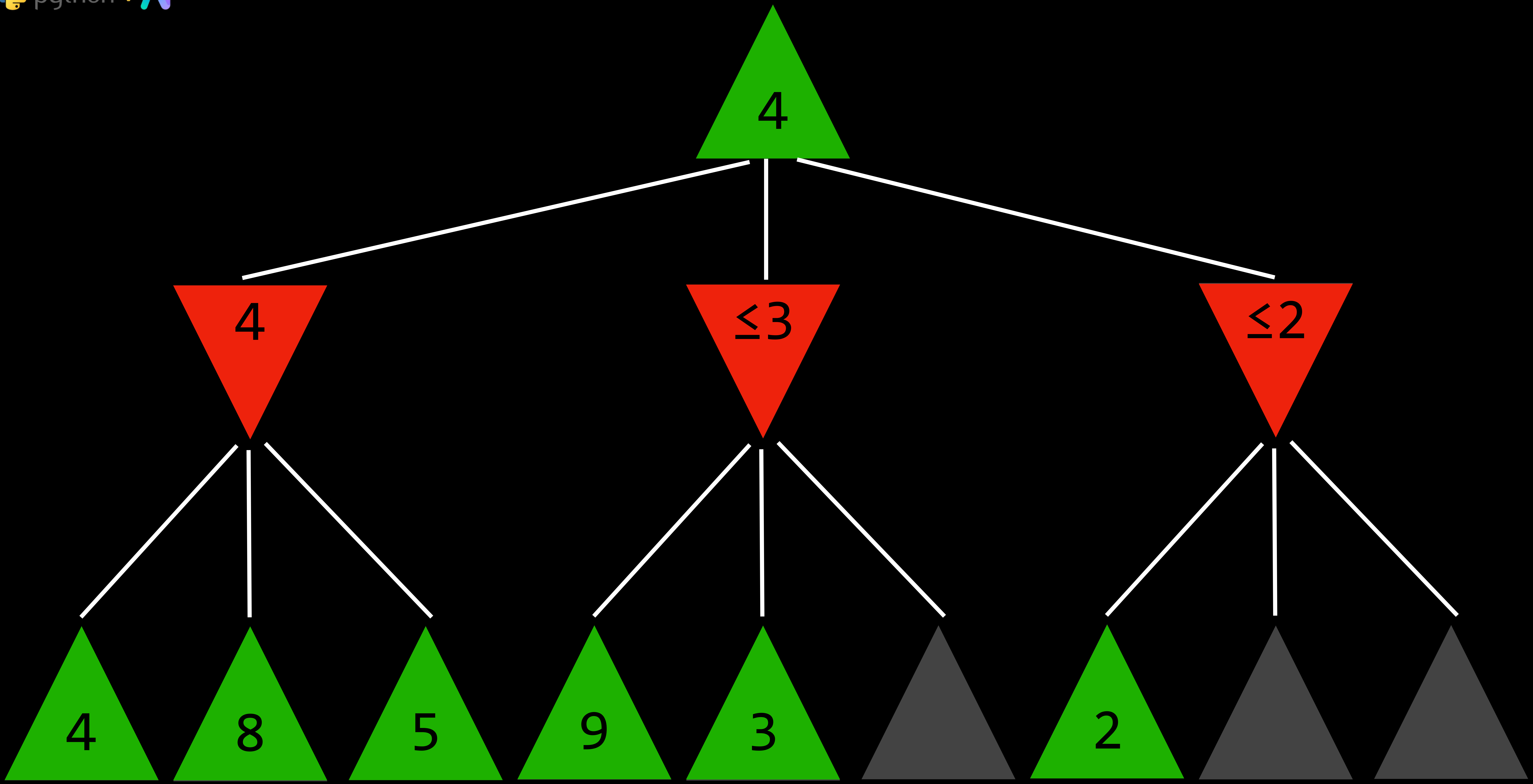
```
function MAX-VALUE(state):
    if TERMINAL(state):
        return UTILITY(state)
    v = -∞
    for action in ACTIONS(state):
        v = MAX(v, MIN-VALUE(RESULT(state, action)))
    return v
```

Minimax

```
function MIN-VALUE(state):
    if TERMINAL(state):
        return UTILITY(state)
    v = ∞
    for action in ACTIONS(state):
        v = MIN(v, MAX-VALUE(RESULT(state, action)))
    return v
```

Optimizations


4
4
3
2
4
8
5
9
3
7
2
4
6



4
4
≤3
≤2
4
8
5
9
3
2

Alpha-Beta Pruning

255,168

total possible Tic-Tac-Toe games

288,000,000,000

total possible chess games
after four moves each

10^{29000}

total possible chess games
(lower bound)

10^{17}

Age of universe
in seconds

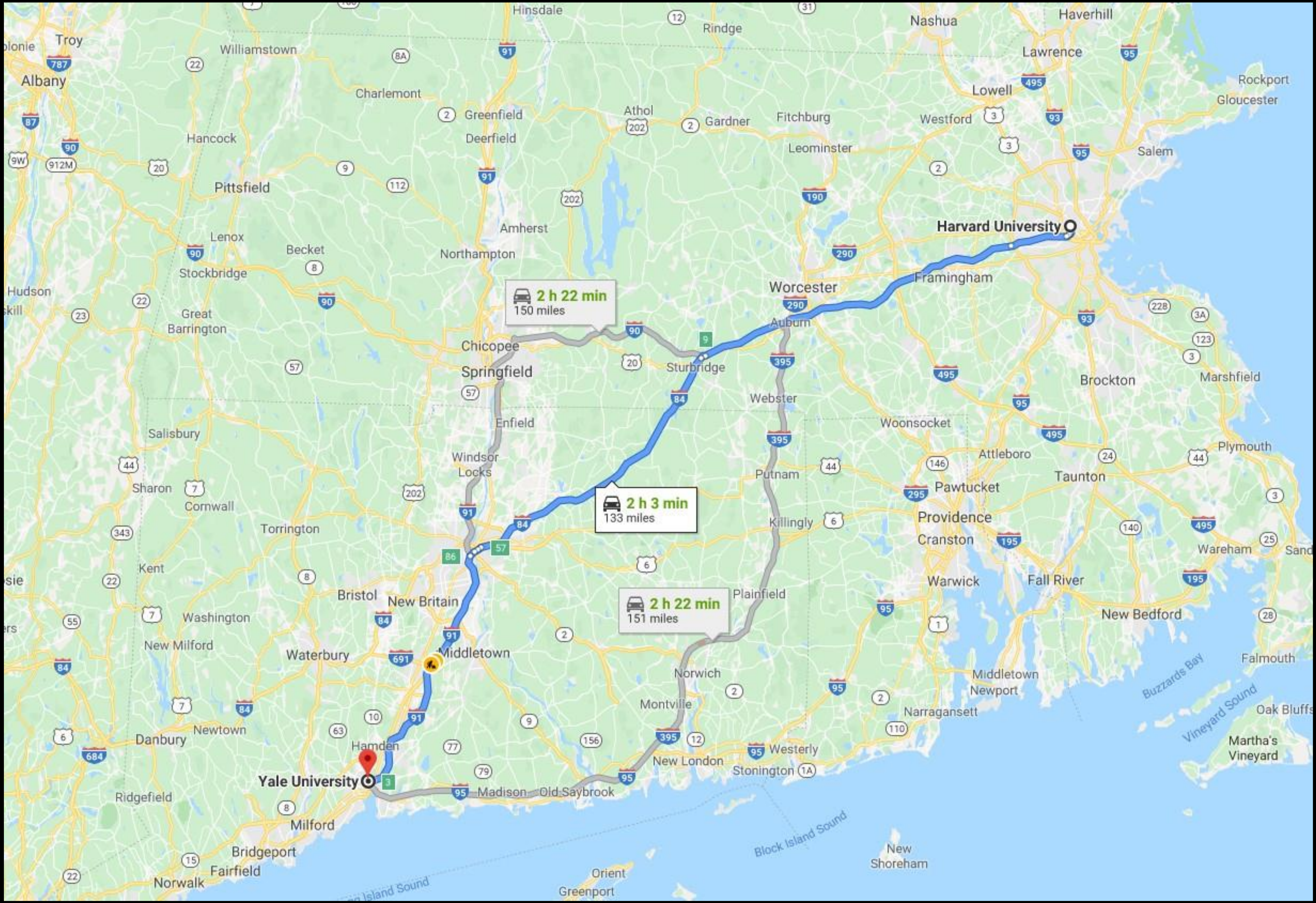
$$10^{10^{48}}$$

total possible go games
(lower bound)

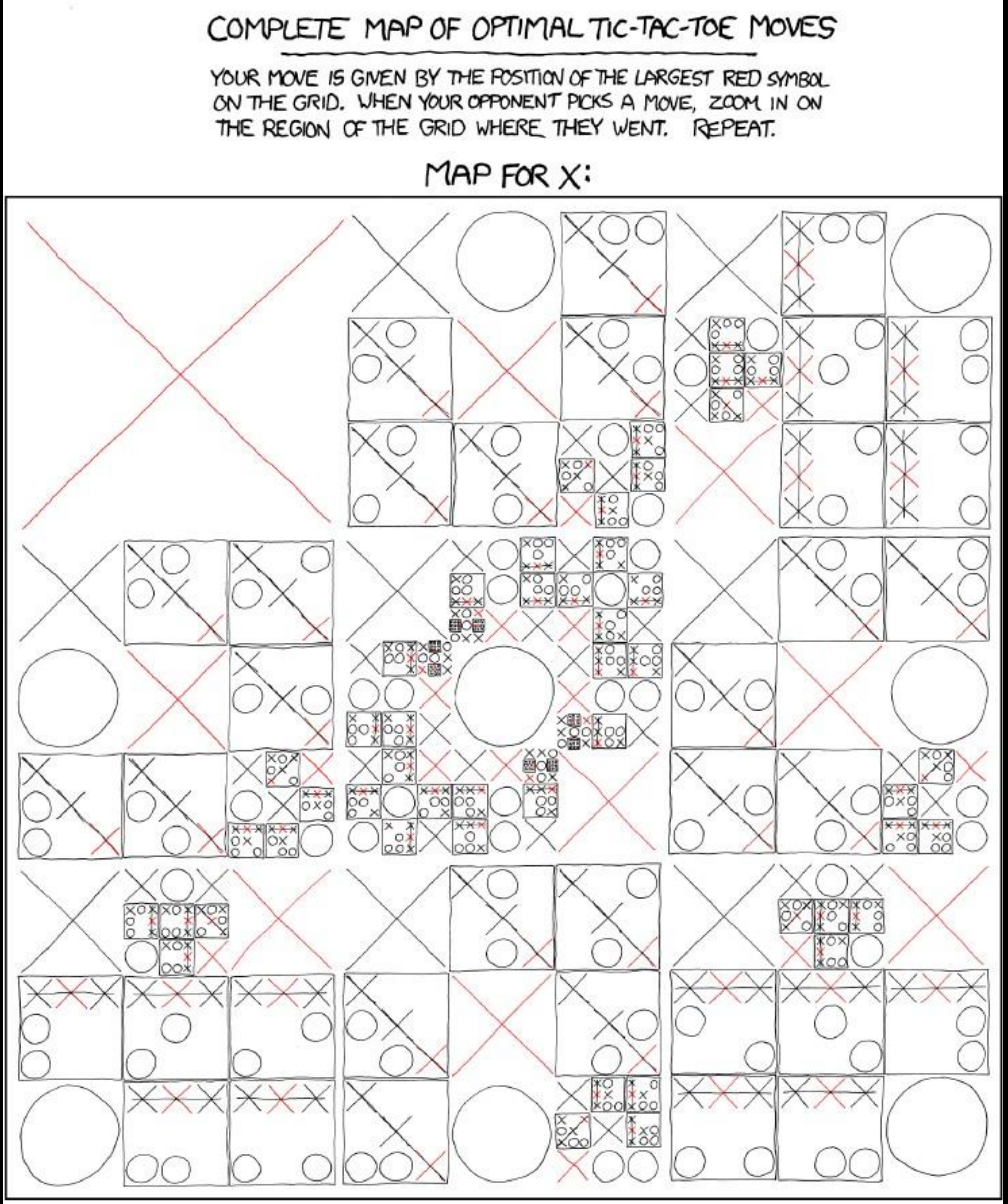
Depth-Limited Minimax

evaluation function

function that estimates the expected utility of the game from a given state



Harvard University
Yale University
2 h 22 min
150 miles
2 h 3 min
133 miles
2 h 22 min
151 miles
Worcester
Framingham
Auburn
Sturbridge
Springfield
Chicopee
Enfield
Windsor Locks
New Britain
Middletown
Hamden
Webster
Putnam
Killingly
Plainfield
Norwich
Montville
New London
Madison
Old Saybrook
Providence
Pawtucket
Cranston
Warwick
Fall River
New Bedford
Brockton
Taunton
Attleboro
Woonsocket
Lowell
Lawrence
Haverhill
Nashua
Westford
Leominster
Fitchburg
Gardner
Athol
Amherst
Northampton
Greenfield
Deerfield
Pittsfield
Albany
Troy
Waterbury
Bristol
Torrington
Danbury
Milford
Bridgeport
Fairfield
Norwalk
Newport
Narragansett
Westerly
Stonington
Block Island Sound
Martha's Vineyard



https://xkcd.com/832/

Adversarial Search

Introduction to

Artificial Intelligence

with Python