MiniMax Games

Frank Pfenning

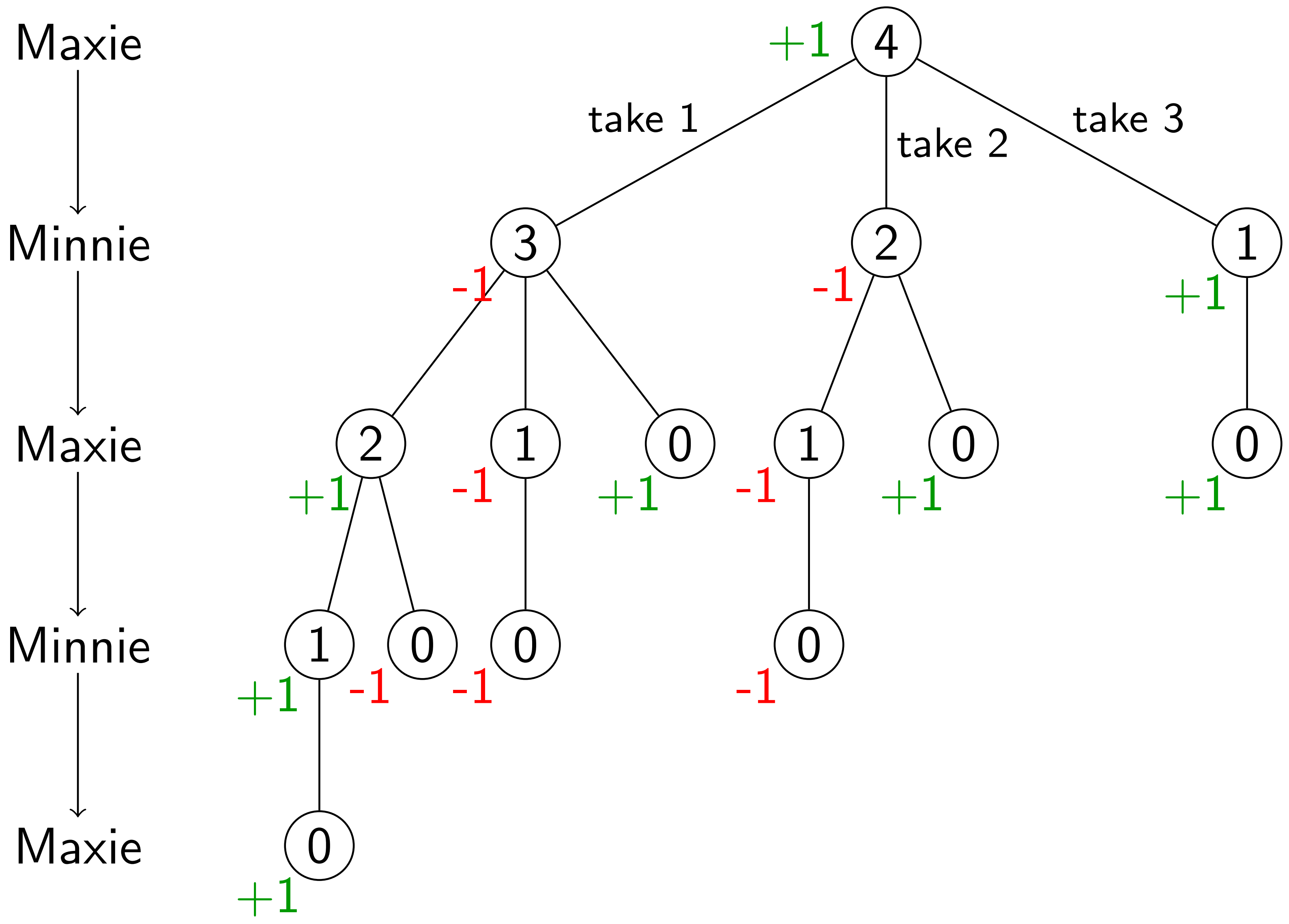
15-150, April 8, 2020

Learning Objectives

- Game trees
- Practicing sequences and parallelism
- Refactoring
    - Generalizing an instance
    - Creating module structure

Outline

- The Nim game
    - Intuitively
    - Live code
    - Sequences, revisited
- Classes of games
- Refactoring
    - A `GAMES` signature
    - A `PLAYER` signature


Maxie
Minnie
Maxie
Minnie
Maxie
take 1
take 2
take 3
4
3
2
1
0
+1
-1

Classes of Games

- 2-player, alternating turns
- Deterministic (no dice)
- Perfect information (no hidden state)
- Zero-sum (A wins iff B loses, or tie)
- Finitely branching
- Examples: tic-tac-toe, connect4, checkers, chess, go, . . .

Estimators

- In practice, we cannot explore the full tree for interesting games
- We cut off exploration (based on various criteria) and estimate the value of the position
- Using minimax (or smarter alternatives, see next lecture) to propagate value up the tree
- Better estimators (generally) result in better players

Summary

- Game trees
- Practicing sequences and parallelism
- Refactoring
  - Generalizing an instance
  - Creating module structure