# More Realistic Adversarial Settings

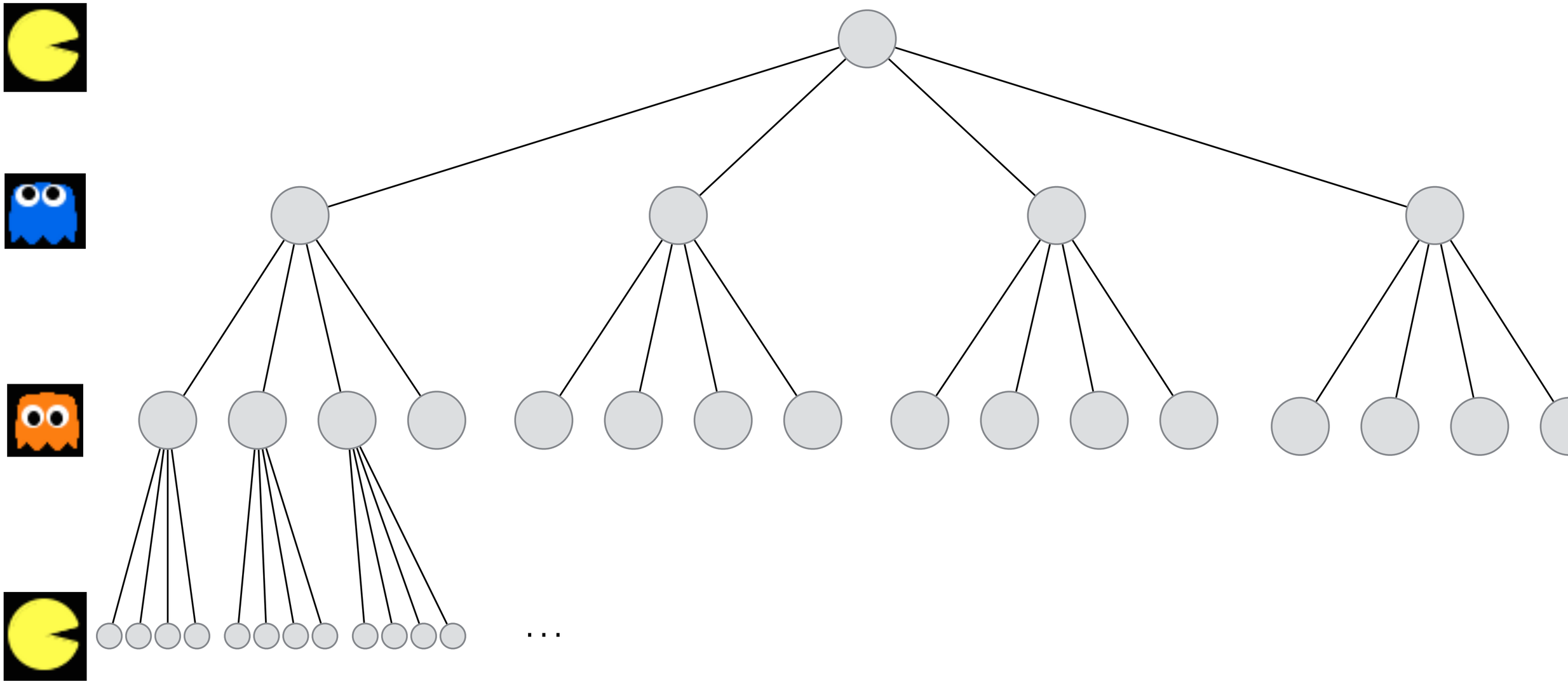
Virginia Tech CS5804
Introduction to Artificial Intelligence
Spring 2015

# Review

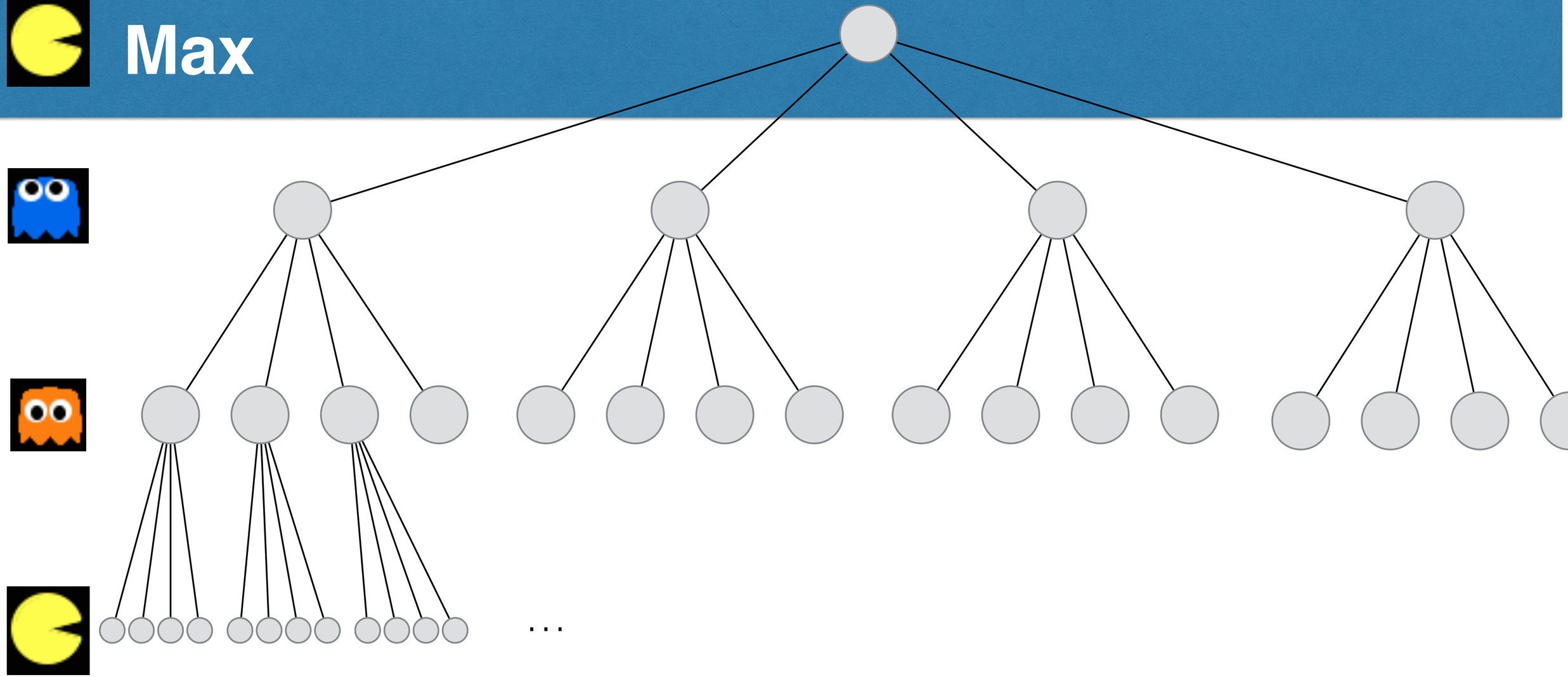
- Minimax search
  - How to adjust for more than two agents, for non-zero-sum
  - Analysis very similar to DFS
- Alpha-beta pruning
  - Analysis?

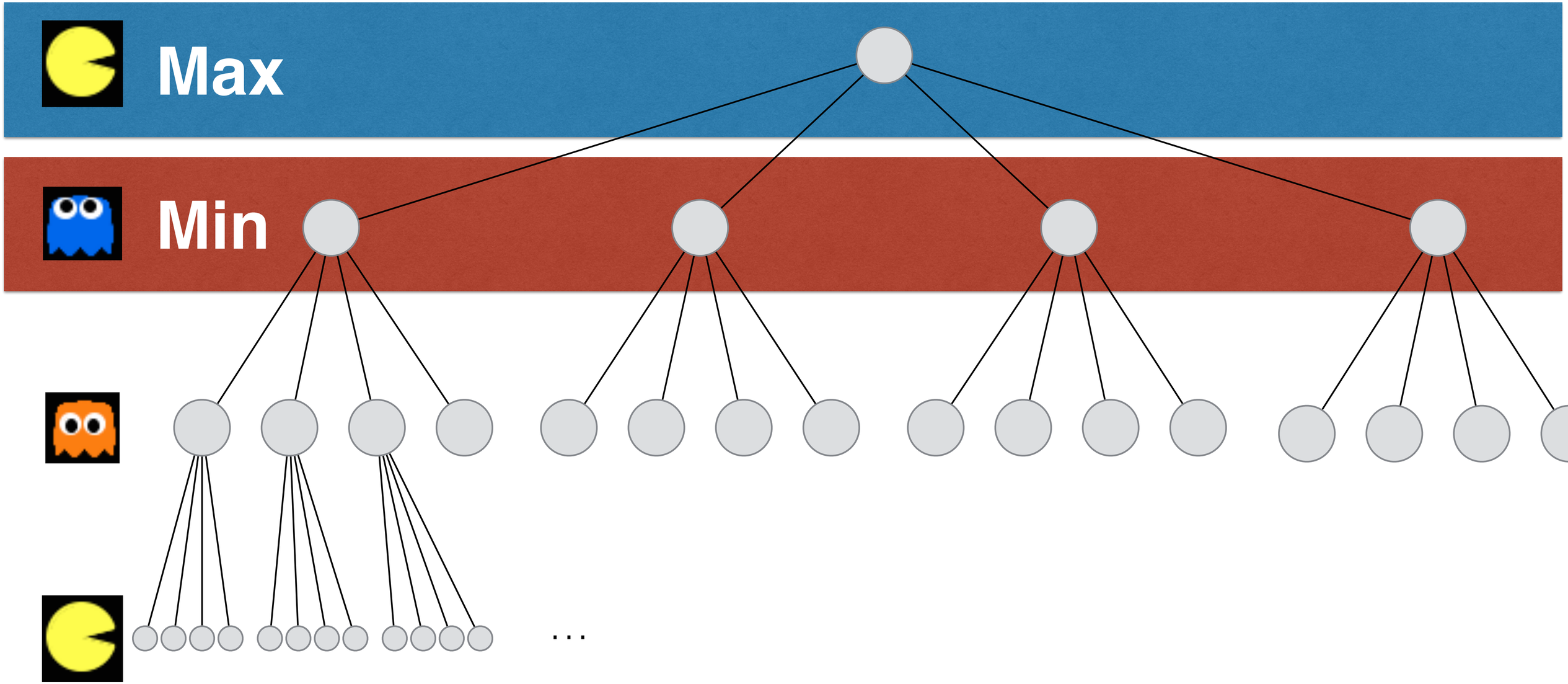
# Outline

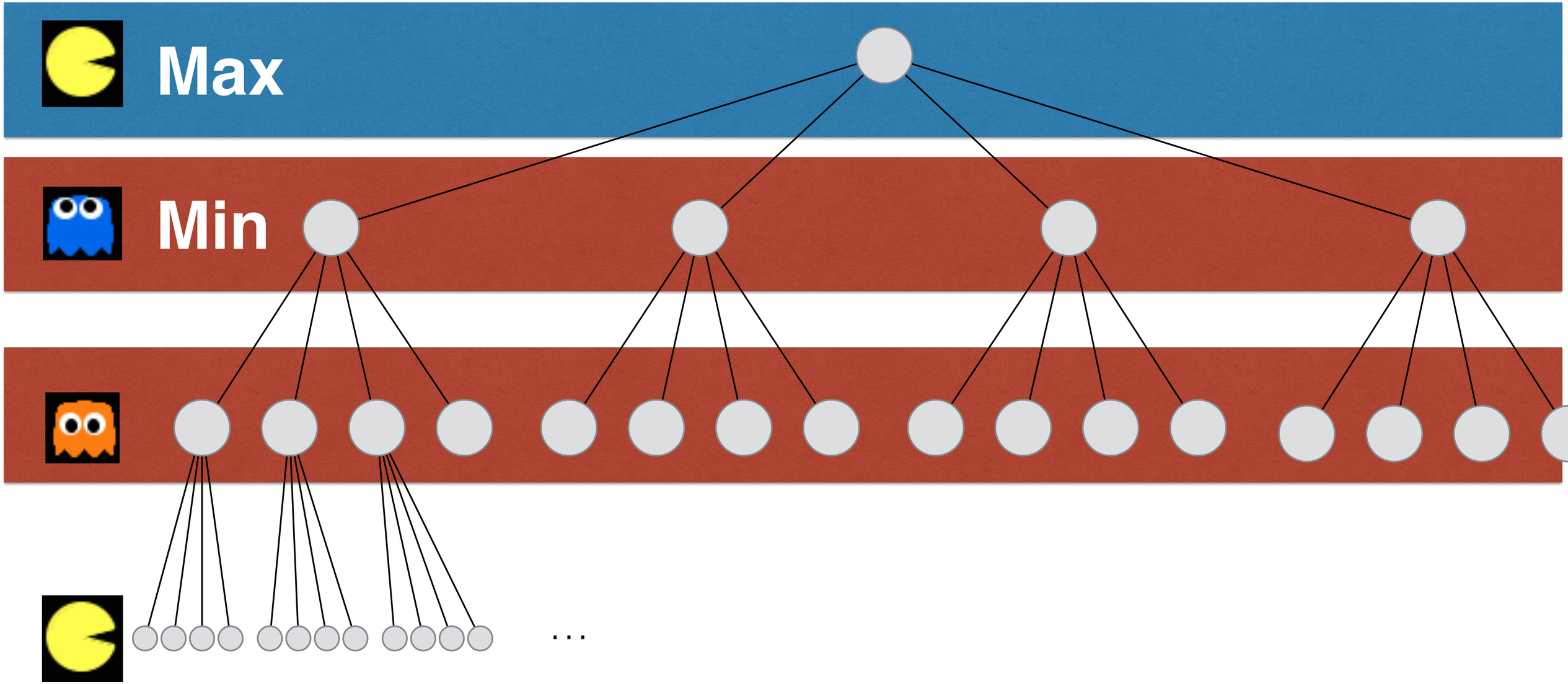
- Move ordering
- Stochastic games
- (Partially-observable games)

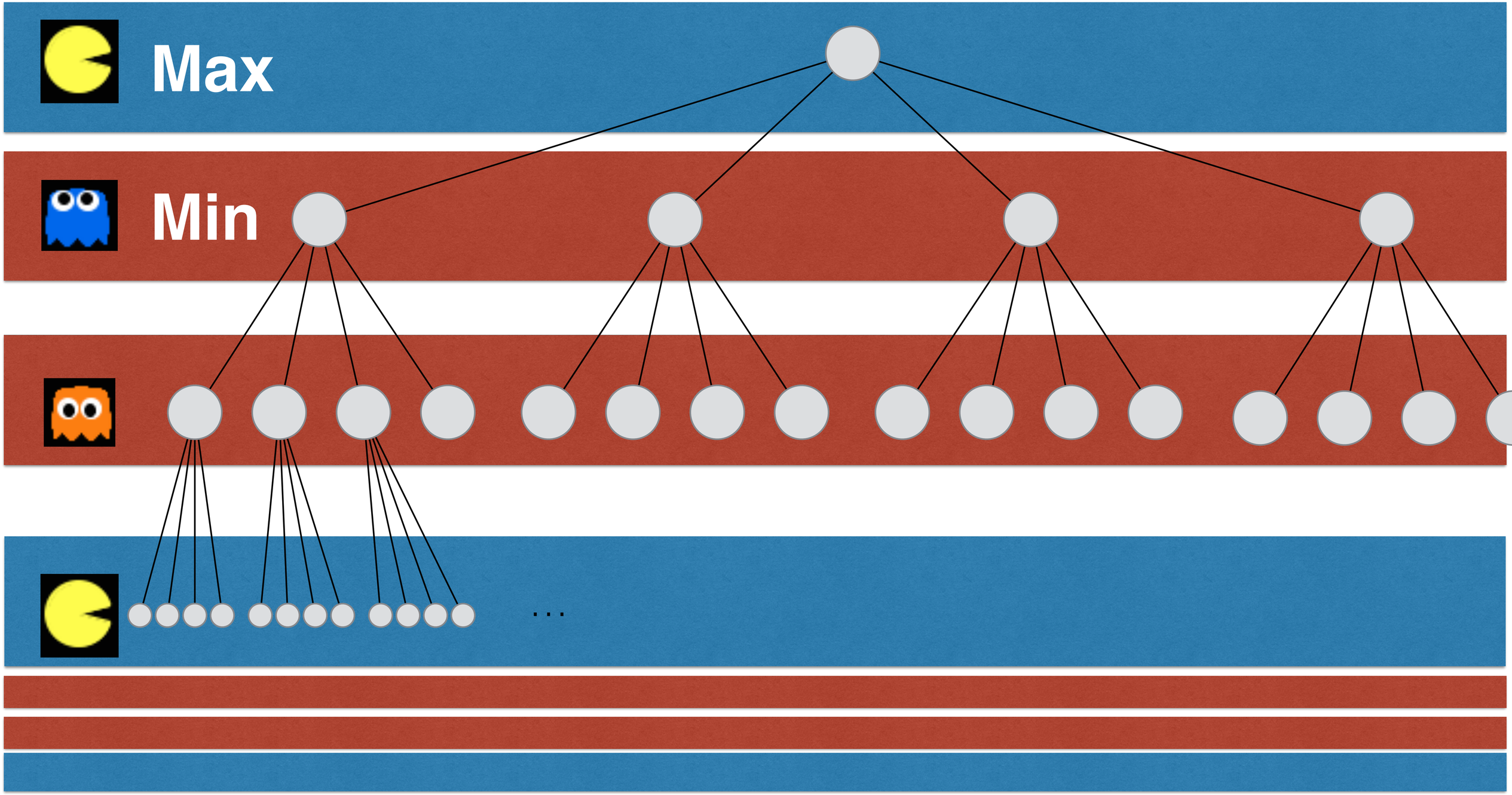


# Max




Max
Min
...


Max
Min
...


Max
Min
...

MINIMAX(*s*) =

if TERMINAL-TEST(*s*) then UTILITY(*s*)

if PLAYER(*s*) = MAX then

max of MINIMAX(RESULT(*s*,*a*)) for *a* in ACTIONS(s)

if PLAYER(s) = MIN then

min of MINIMAX(RESULT(*s*,*a*)) for *a* in ACTIONS(s)

**function** MINIMAX-DECISION($state$) **returns** *an action*
  **return** $\arg\max_{a \in \text{ACTIONS}(s)}$ MIN-VALUE(RESULT($state, a$))

---

**function** MAX-VALUE($state$) **returns** *a utility value*
  **if** TERMINAL-TEST($state$) **then return** UTILITY($state$)
  $v \leftarrow -\infty$
  **for each** $a$ **in** ACTIONS($state$) **do**
    $v \leftarrow$ MAX($v$, MIN-VALUE(RESULT($s$, $a$)))
  **return** $v$

---

**function** MIN-VALUE($state$) **returns** *a utility value*
  **if** TERMINAL-TEST($state$) **then return** UTILITY($state$)
  $v \leftarrow \infty$
  **for each** $a$ **in** ACTIONS($state$) **do**
    $v \leftarrow$ MIN($v$, MAX-VALUE(RESULT($s$, $a$)))
  **return** $v$

```
function ALPHA-BETA-SEARCH(state) returns an action
  v ← MAX-VALUE(state, −∞, +∞)
  return the action in ACTIONS(state) with value v
```

---

```
function MAX-VALUE(state, α, β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← −∞
  for each a in ACTIONS(state) do
    v ← MAX(v, MIN-VALUE(RESULT(s,a), α, β))
    if v ≥ β then return v
    α ← MAX(α, v)
  return v
```

---

```
function MIN-VALUE(state, α, β) returns a utility value
  if TERMINAL-TEST(state) then return UTILITY(state)
  v ← +∞
  for each a in ACTIONS(state) do
    v ← MIN(v, MAX-VALUE(RESULT(s,a) , α, β))
    if v ≤ α then return v
    β ← MIN(β, v)
  return v
```

NATURE | NEWS

# Game theorists crack poker

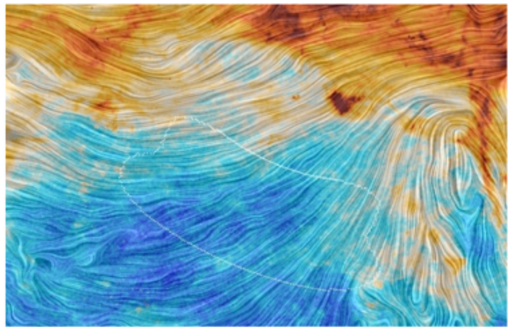
**An 'essentially unbeatable' algorithm for the popular card game points to strategies for solving real-life problems without having complete information.**

Philip Ball

08 January 2015



Top story



Gravitational waves discovery officially dead

Galactic dust confirmed as the source of the signal that researchers thought was evidence for gravitational waves from the early Universe.

Recent | Read | Commented | Emailed

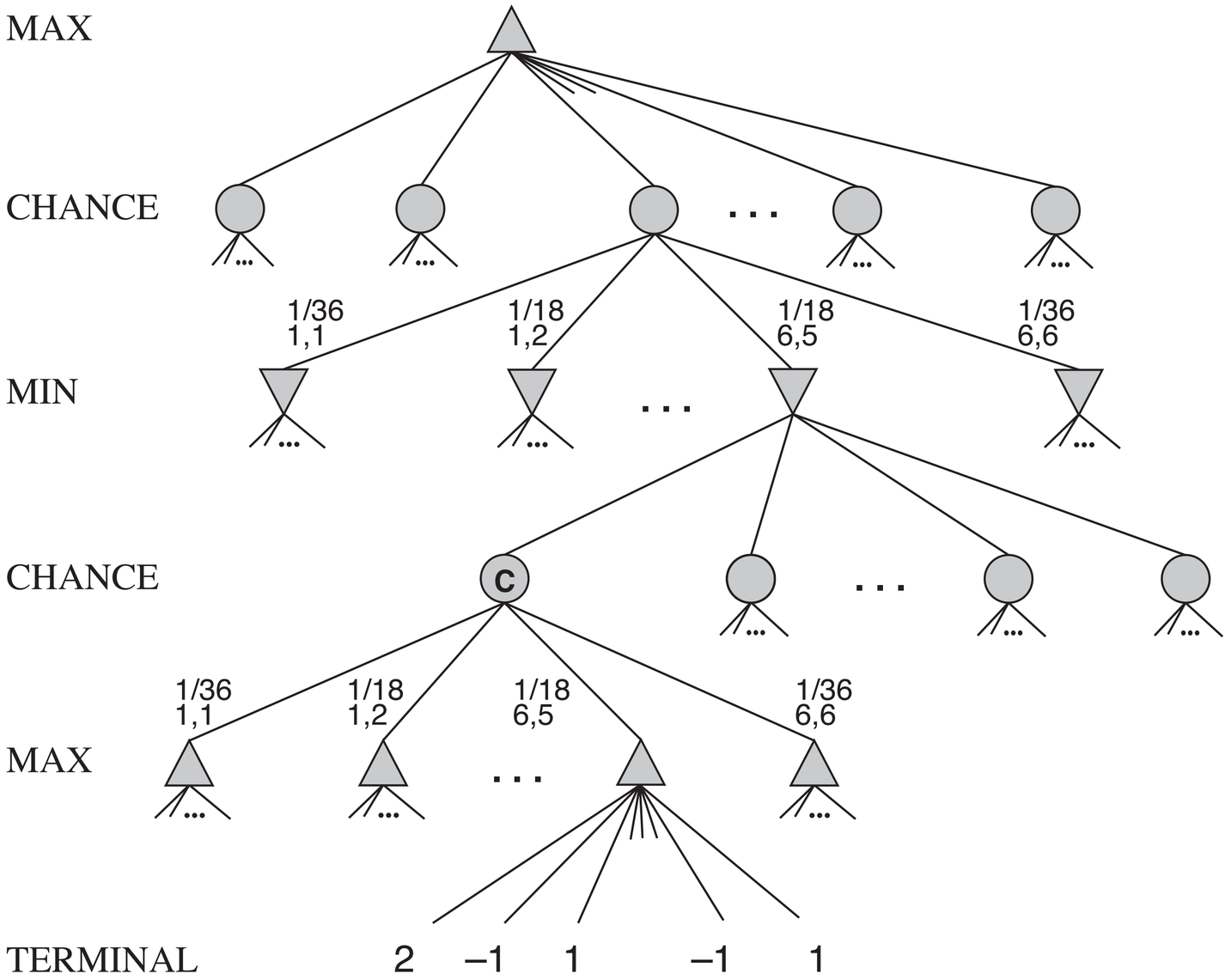
1. Comet scientists abandon Philae flyby
   *Nature* | 03 February 2015
2. Scientists cheer vote to allow three-person embryos
   *Nature* | 03 February 2015
3. Carl Djerassi, 1923–2015
   *Nature* | 03 February 2015
4. Saudi university backs slow road to modernization
   *Nature* | 03 February 2015
5. Major biodiversity initiative needs support
   *Nature* | 03 February 2015

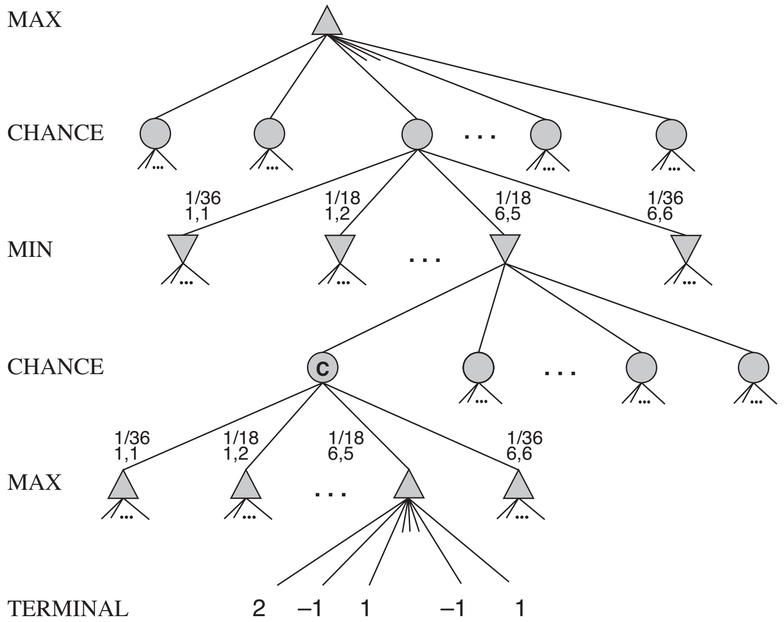
# Expectimax

- New player type: Chance

$$\textsc{Expectinimimax}(s) =$$

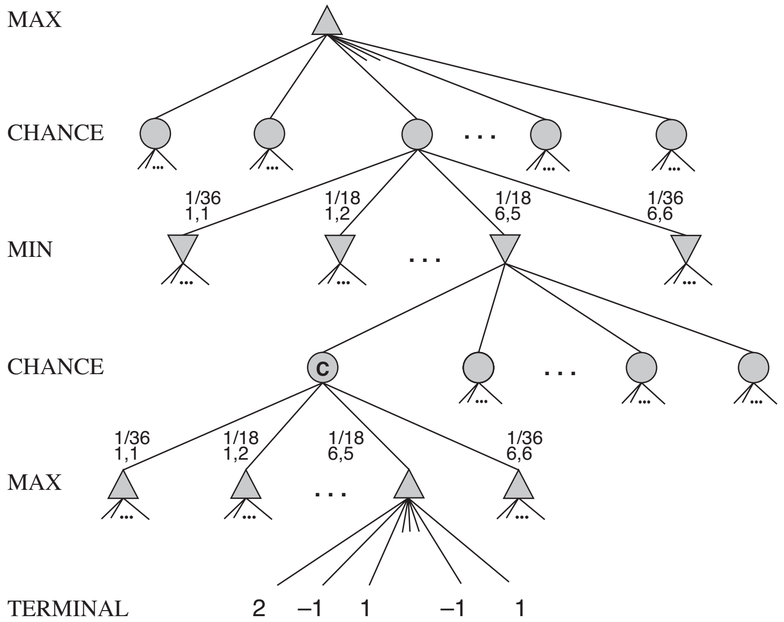
$$\begin{cases} \textsc{Utility}(s) & \text{if } \textsc{Terminal-Test}(s) \\ \max_a \textsc{Expectiminimax}(\textsc{Result}(s,a)) & \text{if } \textsc{Player}(s) = \textsc{Max} \\ \min_a \textsc{Expectiminimax}(\textsc{Result}(s,a)) & \text{if } \textsc{Player}(s) = \textsc{Min} \\ \sum_r \Pr(r)\textsc{Expectiminimax}(\textsc{Result}(s,r)) & \text{if } \textsc{Player}(s) = \textsc{Chance} \end{cases}$$


MAX
CHANCE
1/36
1,1
1/18
1,2
1/18
6,5
1/36
6,6
MIN
C
CHANCE
1/36
1,1
1/18
1,2
1/18
6,5
1/36
6,6
MAX
TERMINAL
2
–1
1
–1
1




$$\textsc{Expectinimimax}(s) =$$

$$\begin{cases} \textsc{Utility}(s) & \text{if } \textsc{Terminal-Test}(s) \\ \max_a \textsc{Expectiminimax}(\textsc{Result}(s,a)) & \text{if } \textsc{Player}(s) = \textsc{Max} \\ \min_a \textsc{Expectiminimax}(\textsc{Result}(s,a)) & \text{if } \textsc{Player}(s) = \textsc{Min} \\ \sum_r \Pr(r)\textsc{Expectiminimax}(\textsc{Result}(s,r)) & \text{if } \textsc{Player}(s) = \textsc{Chance} \end{cases}$$




$$\textsc{Expectinimimax}(s) =$$

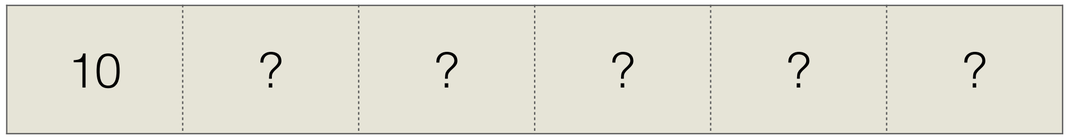
$$\begin{cases} \textsc{Utility}(s) & \text{if } \textsc{Terminal-Test}(s) \\ \max_a \textsc{Expectiminimax}(\textsc{Result}(s,a)) & \text{if } \textsc{Player}(s) = \textsc{Max} \\ \min_a \textsc{Expectiminimax}(\textsc{Result}(s,a)) & \text{if } \textsc{Player}(s) = \textsc{Min} \\ \sum_r \Pr(r)\textsc{Expectiminimax}(\textsc{Result}(s,r)) & \text{if } \textsc{Player}(s) = \textsc{Chance} \end{cases}$$

# Pruning Chance Nodes

- Bounds on true utility function -> bounds on expectation
- -10 <= Utility <= 10
- uniform probability die:

# Pruning Chance Nodes

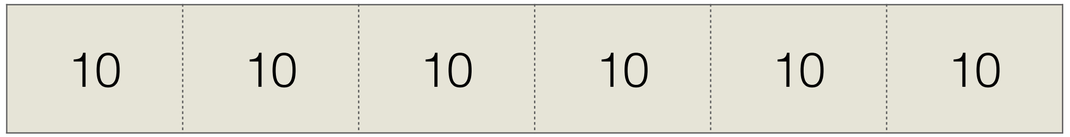
- Bounds on true utility function -> bounds on expectation
- -10 <= Utility <= 10
- uniform probability die:



# Pruning Chance Nodes

- Bounds on true utility function -> bounds on expectation
- -10 <= Utility <= 10
- uniform probability die:


Best case

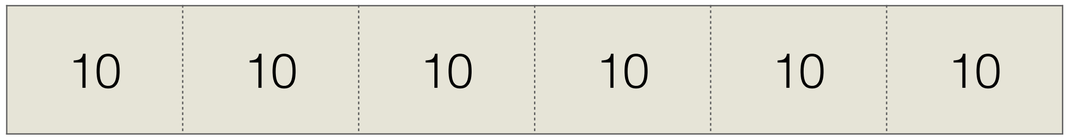



expectation 10

# Pruning Chance Nodes

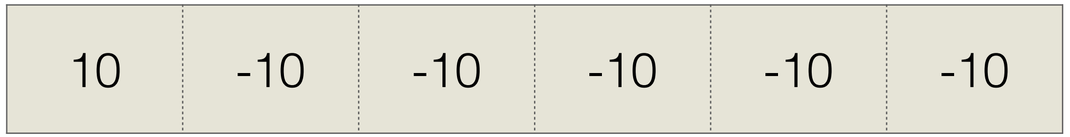
- Bounds on true utility function -> bounds on expectation
- -10 <= Utility <= 10
- uniform probability die:


Best case




expectation 10

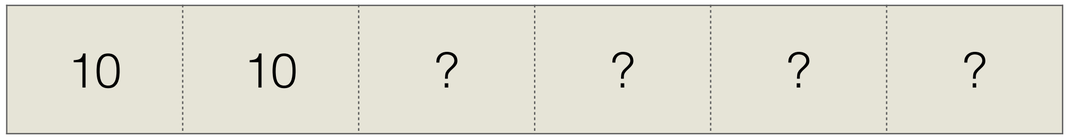
Worst case




expectation -6.67

# Pruning Chance Nodes

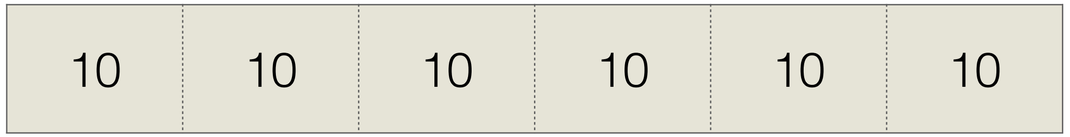
- Bounds on true utility function -> bounds on expectation
- -10 <= Utility <= 10
- uniform probability die:



# Pruning Chance Nodes

- Bounds on true utility function -> bounds on expectation
- -10 <= Utility <= 10
- uniform probability die:


Best case

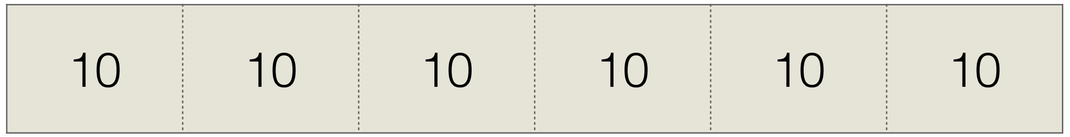



expectation 10

# Pruning Chance Nodes

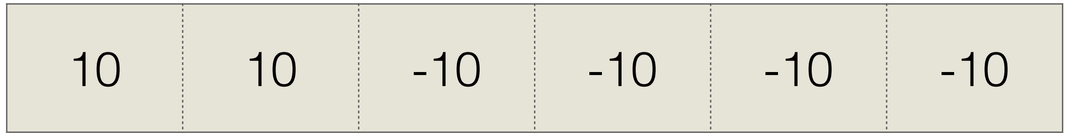
- Bounds on true utility function -> bounds on expectation
- -10 <= Utility <= 10
- uniform probability die:


Best case




expectation 10

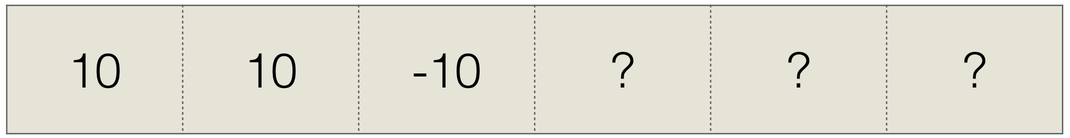
Worst case




expectation -3.33

# Pruning Chance Nodes

- Bounds on true utility function -> bounds on expectation
- -10 <= Utility <= 10
- uniform probability die:



# Pruning Chance Nodes

- Bounds on true utility function -> bounds on expectation
- -10 <= Utility <= 10
- uniform probability die:

| 10 | 10 | -10 | ? | ? | ? |
|---|---|---|---|---|---|

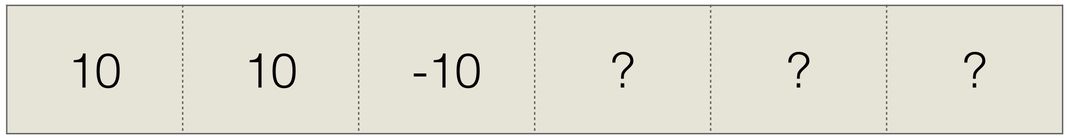
Best case

| 10 | 10 | -10 | 10 | 10 | 10 |
|---|---|---|---|---|---|

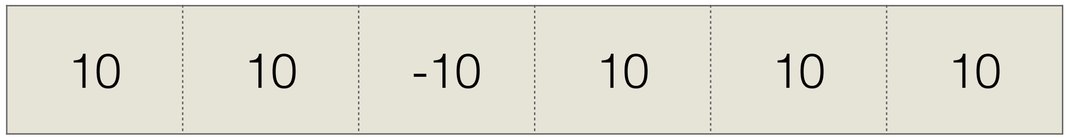
expectation 6.67

# Pruning Chance Nodes

- Bounds on true utility function -> bounds on expectation
- -10 <= Utility <= 10
- uniform probability die:

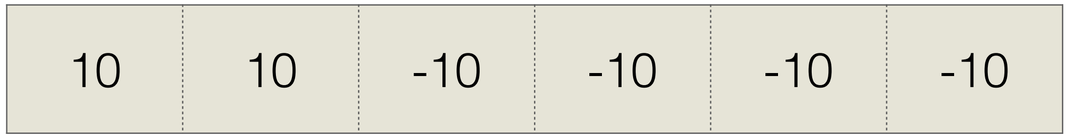



Best case




expectation 6.67

Worst case




expectation -3.33

# Monte Carlo

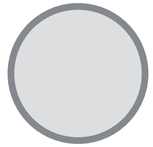
- Often too expensive to consider all Chance outcomes
- Randomly sample result on Chance turns

# Partial Observations

- One approach: simulate perfect information with Chance nodes

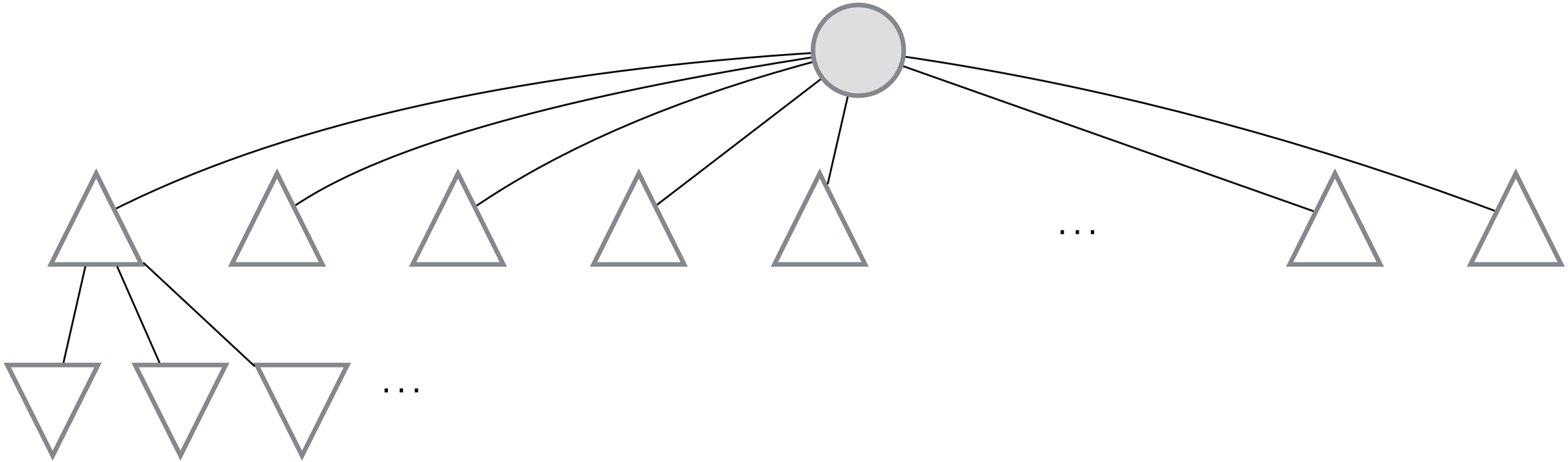
# Partial Observations

- One approach: simulate perfect information with Chance nodes



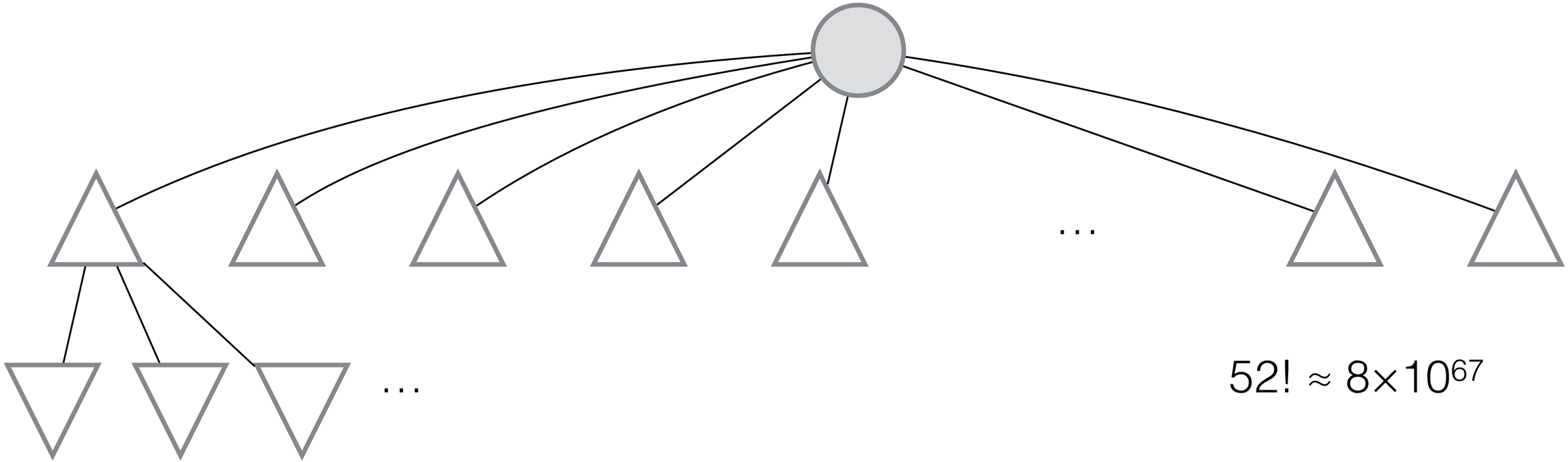
# Partial Observations

- One approach: simulate perfect information with Chance nodes



# Partial Observations

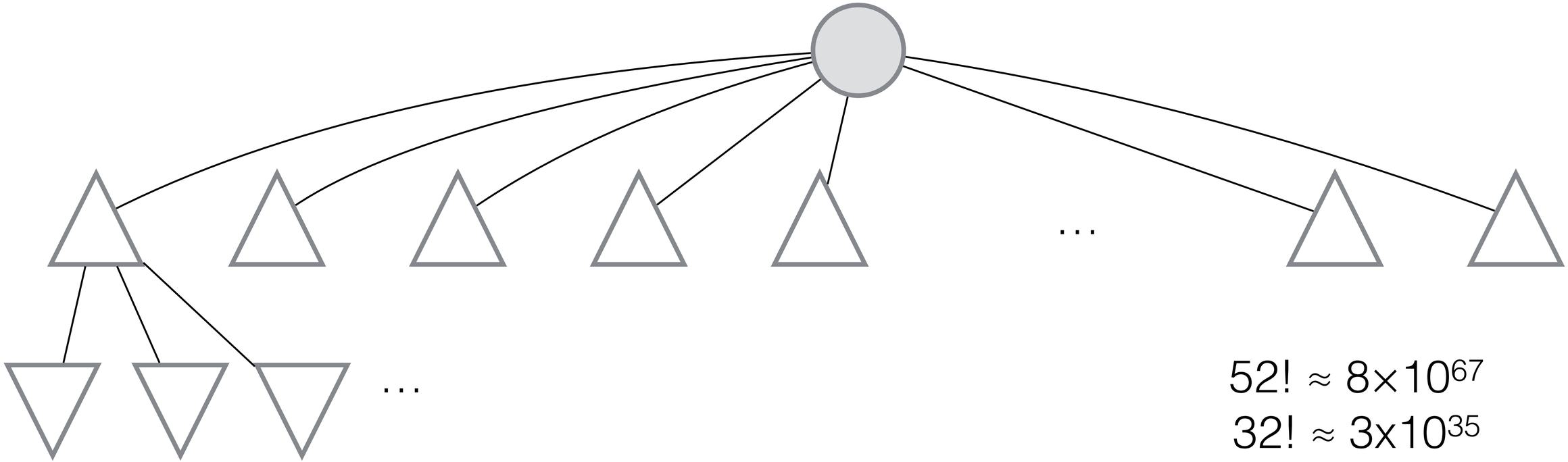
- One approach: simulate perfect information with Chance nodes




$52! \approx 8 \times 10^{67}$

# Partial Observations

- One approach: simulate perfect information with Chance nodes

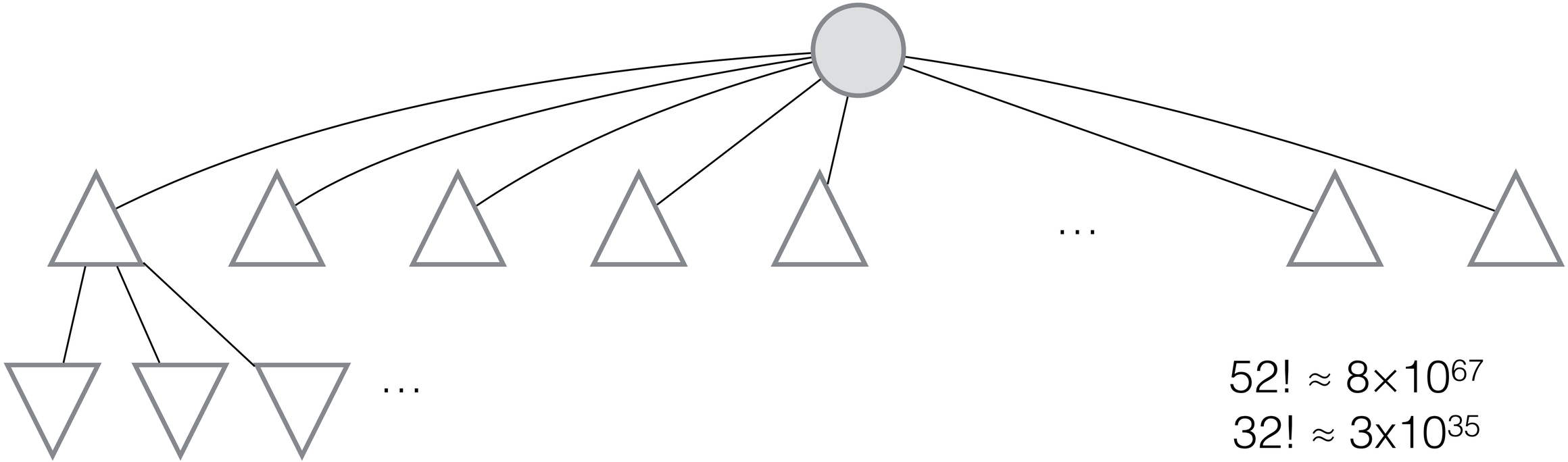



$52! \approx 8 \times 10^{67}$

$32! \approx 3 \times 10^{35}$

# Partial Observations

- One approach: simulate perfect information with Chance nodes




$52! \approx 8 \times 10^{67}$

$32! \approx 3 \times 10^{35}$

$22! \approx 4 \times 10^{26}$

# Summary

- Minimax logic works for any move ordering
- Expectiminimax adds Chance “player” and uses expected value
- Monte Carlo simulates chance nodes
  - Strategy for handling partial information