# Extensive form games

# Five pirates' game

# Five pirates' game

- 5 pirates found a chest containing 1000 gold coins
- Instead of dividing it uniformly, they decided to play the following game
  - Each pirate (after a raffle) will propose a way to divide the coins
  - If the proposal is accepted by the majority, it will occur
  - If not, the pirate who made the proposal will be thrown at the sea
  - Each pirate wants to maximize the amount of coins he will receive
  - All pirates are blood thirsty, i.e., they prefer a dead pirate over an alive pirate

# Extensive form games

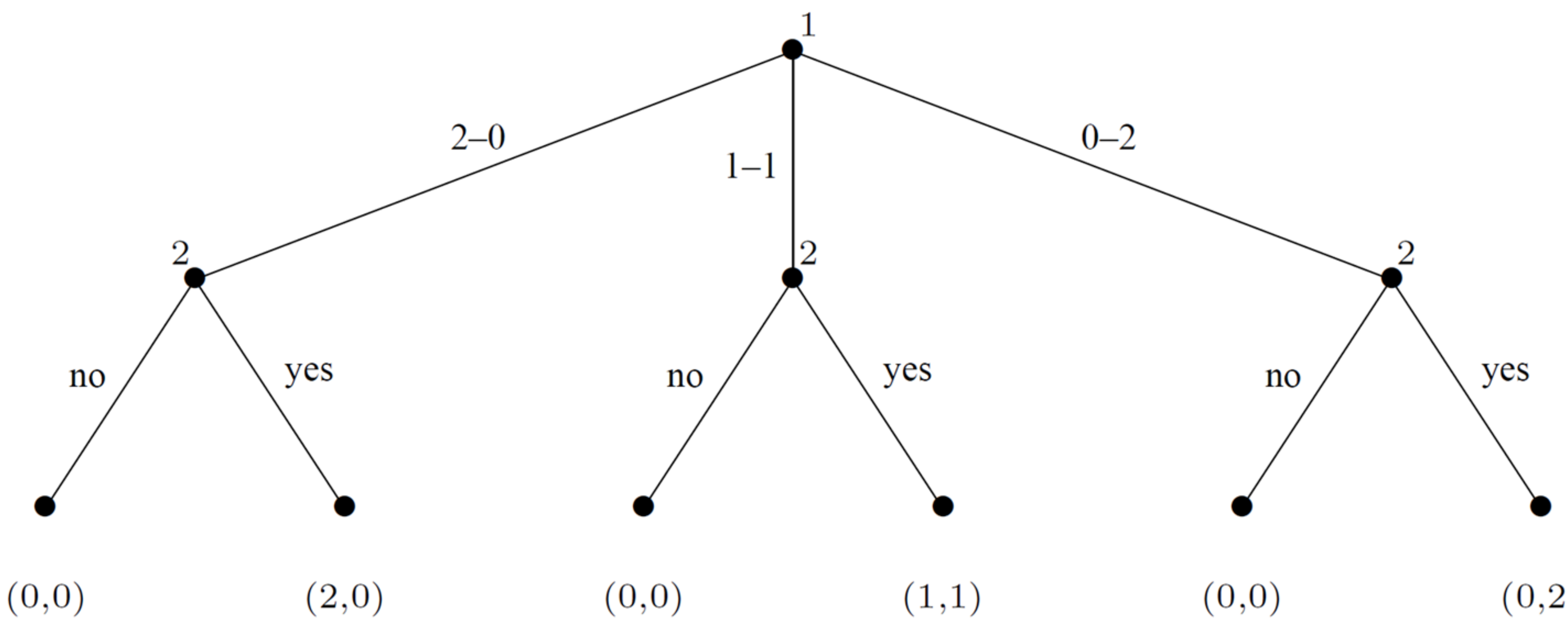
- The normal form game representation does not incorporate any notion of sequence, or time, of the actions of the players
- The extensive form is an alternative representation that makes the temporal structure explicit
- Two variants:
  - perfect information extensive-form games
  - imperfect-information extensive-form games

# Perfect-information games in extesive form

- A tree in the sense of graph theory
  - each node represents the choice of one of the players
  - each edge represents a possible action
  - the leaves represent final outcomes over which each player has a utility function
- In certain circles (in particular, in artificial intelligence), these are known simply as game trees

# Perfect-information games in extesive form

- The sharing game



# Perfect-information games in extesive form

- A (finite) perfect-information game (in extensive form) is a tuple ***G = (N,A,H,Z, χ, ρ, σ, u)***, where:
  - ***N*** is a set of n players
  - ***A*** is a (single) set of actions
  - ***H*** is a set of nonterminal choice nodes
  - ***Z*** is a set of terminal nodes, disjoint from ***H***
  - $\chi : H \rightarrow 2^A$ is the action function, which assigns to each choice node a set of possible actions

# Perfect-information games in extesive form

- A (finite) perfect-information game (in extensive form) is a tuple $G = (N, A, H, Z, \chi, \rho, \sigma, u)$, where:
  - $N$ is a set of n players
  - $A$ is a (single) set of actions
  - $H$ is a set of nonterminal choice nodes
  - $Z$ is a set of terminal nodes, disjoint from $H$
  - $\chi : H \to 2^A$ is the action function
  - $\rho : H \to N$ is the player function, which assigns to each nonterminal node a player $i \in N$ who chooses an action at that node

# Perfect-information games in extesive form

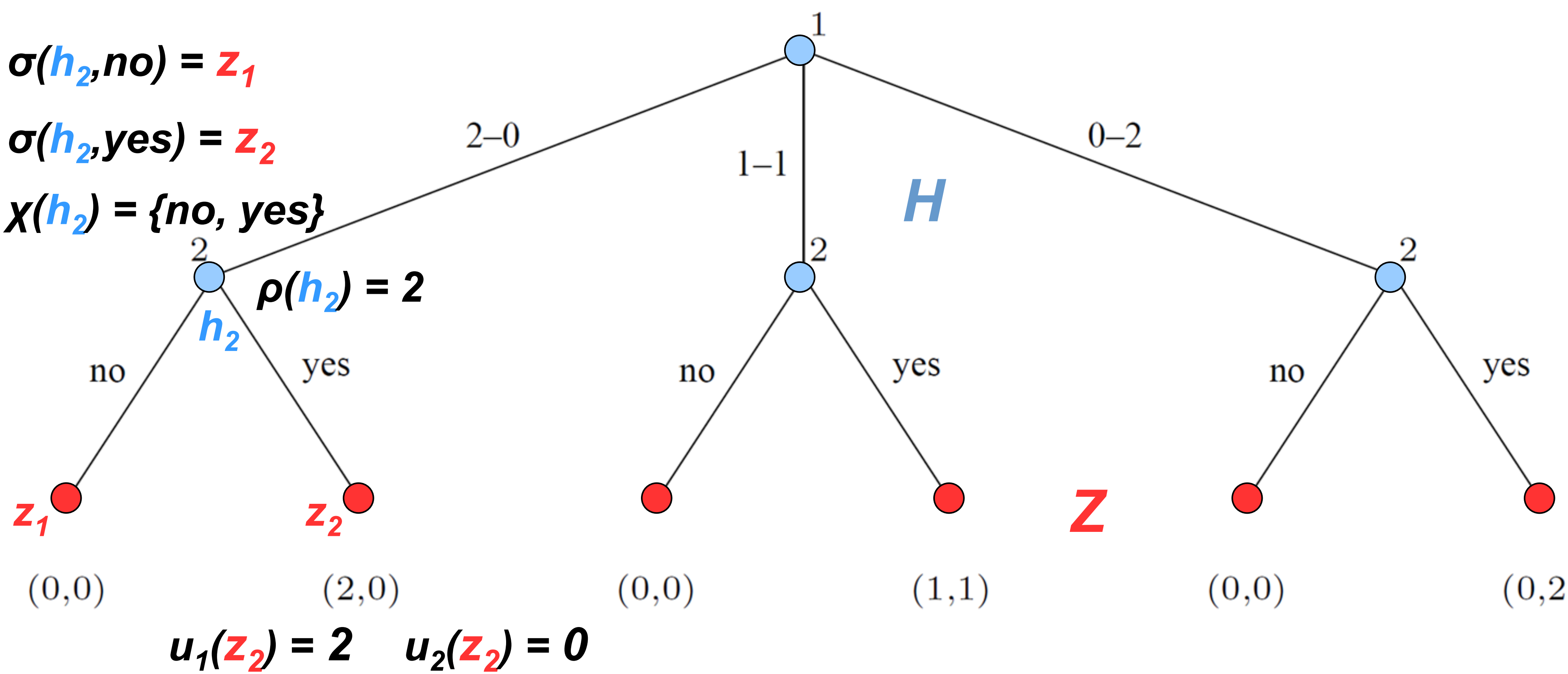
- A (finite) perfect-information game (in extensive form) is a tuple $G = (N, A, H, Z, \chi, \rho, \sigma, u)$, where:
  - $N$ is a set of n players
  - $A$ is a (single) set of actions
  - $H$ is a set of nonterminal choice nodes
  - $Z$ is a set of terminal nodes, disjoint from $H$
  - $\chi : H \rightarrow 2^A$ is the action function
  - $\rho : H \rightarrow N$ is the player function
  - $\sigma : H \times A \rightarrow H \cup Z$ is the successor function, which maps a choice node and an action to a new choice node or terminal node such that for all $h_1, h_2 \in H$ and $a_1, a_2 \in A$, if $\sigma(h_1, a_1) = \sigma(h_2, a_2)$ then $h_1 = h_2$ and $a_1 = a_2$

# Perfect-information games in extesive form

- A (finite) perfect-information game (in extensive form) is a tuple $G = (N, A, H, Z, \chi, \rho, \sigma, u)$, where:
  - $N$ is a set of n players
  - $A$ is a (single) set of actions
  - $H$ is a set of nonterminal choice nodes
  - $Z$ is a set of terminal nodes, disjoint from $H$
  - $\chi : H \rightarrow 2^A$ is the action function
  - $\rho : H \rightarrow N$ is the player function
  - $\sigma : H \times A \rightarrow H \cup Z$ is the successor function
  - $u = (u_1, \ldots, u_n)$, where $u_i : Z \rightarrow \mathbb{R}$ is a real-valued utility function for player $i$ on the terminal nodes $Z$

# Perfect-information games in extesive form

- The sharing game
  - ***N = {1,2}, $A_1$ = {2-0, 1-1, 0-2}, $A_2$ = {no, yes}***



# Perfect-information games in extesive form

- Since the choice nodes form a tree, we can unambiguously identify a node with its history
- We can also define the descendants of a node ***h***, namely all the choice and terminal nodes in the subtree rooted at ***h***

# Perfect-information games in extesive form

- What is the set of pure strategies?
  - A complete specification of which deterministic action to take at every node belonging to that player

# Perfect-information games in extesive form

- Pure strategies
  - Let $\boldsymbol{G = (N,A,H,Z, \chi, \rho, \sigma, u)}$ be a perfect-information extensive-form game
  - Then the pure strategies of player $\boldsymbol{i}$ consist of the Cartesian product:

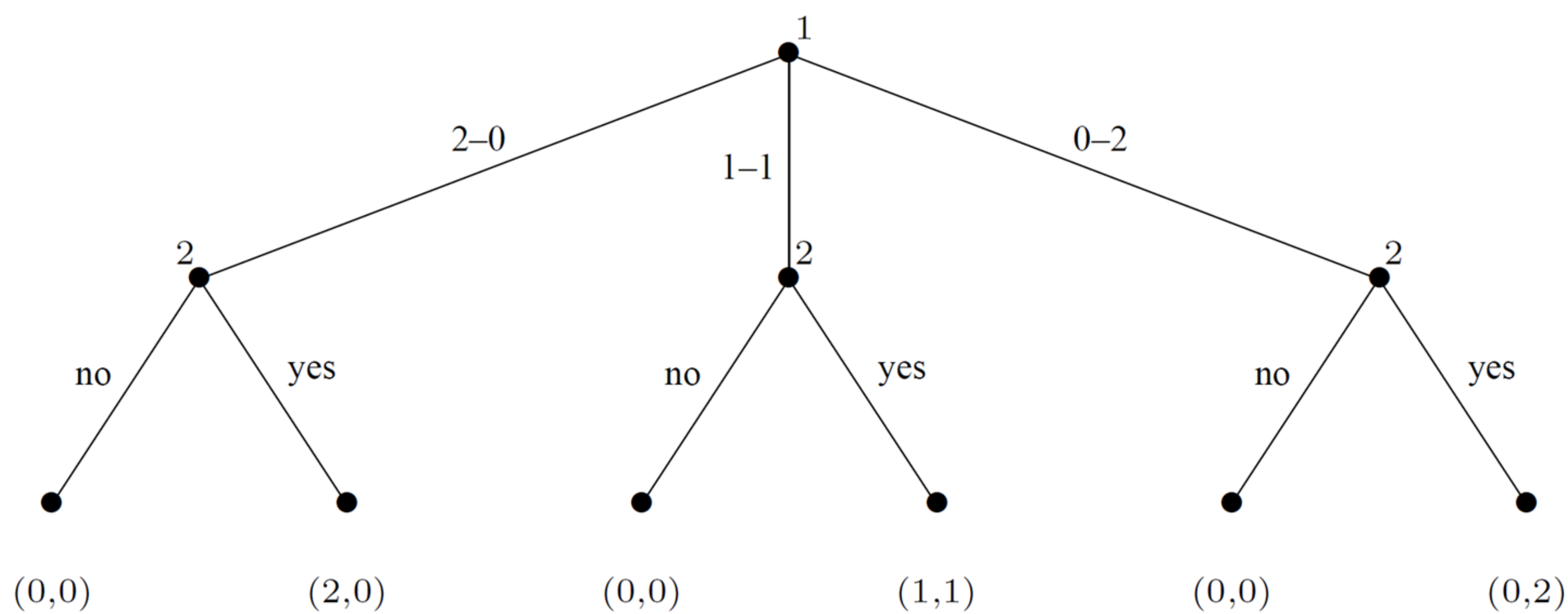
$$\prod_{h \in H,\ \rho(h)=i} \chi(h)$$

# Perfect-information games in extesive form

- Notice that the definition contains a subtlety
  - An agent's strategy requires a decision at each choice node, regardless of whether or not it is reacheable

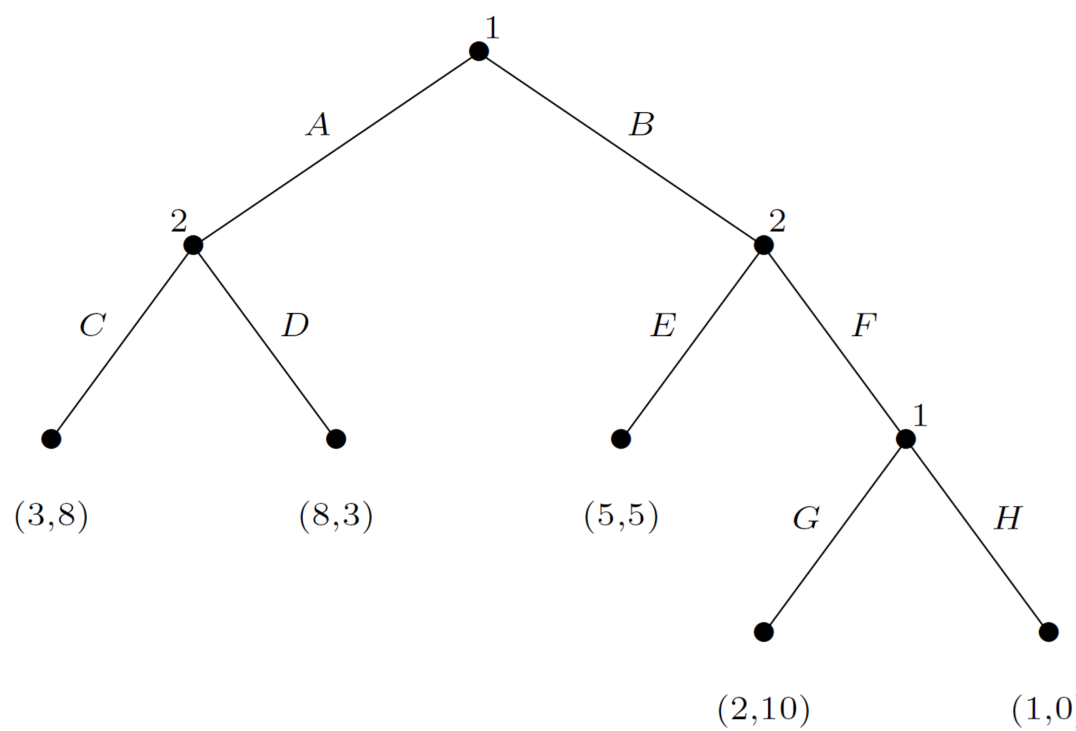
# Perfect-information games in extesive form

- ***$S_1$ = {2–0, 1–1, 0–2}***
- ***$S_2$ = {(yes, yes, yes), (yes, yes, no), (yes, no, yes), (yes, no, no), (no, yes, yes), (no, yes, no), (no, no, yes), (no, no, no)}***



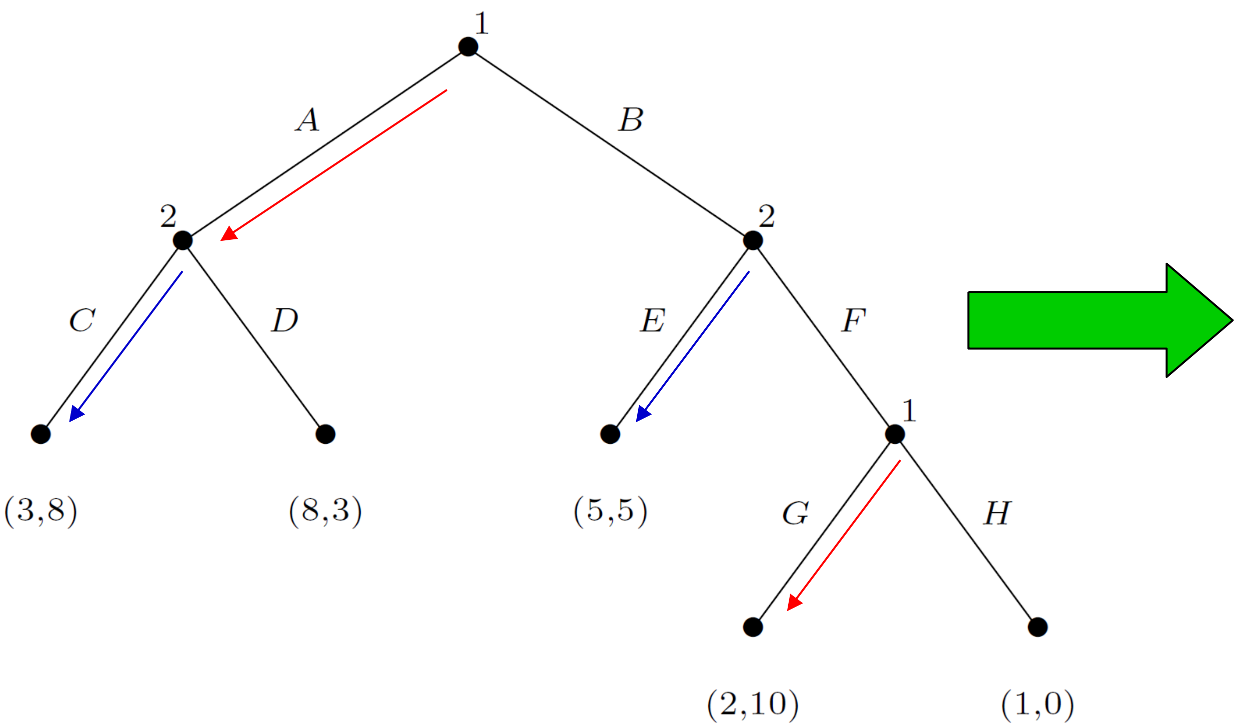
# Perfect-information games in extesive form

- In order to define a complete strategy for this game, each of the players must choose an action at each of his two choice nodes
  - $S_1$ = {(A,G), (A,H), (B,G), (B,H)}
  - $S_2$ = {(C,E), (C, F), (D,E), (D, F)}



# Perfect-information games in extesive form

- For every perfect-information game there exists a corresponding normal-form game



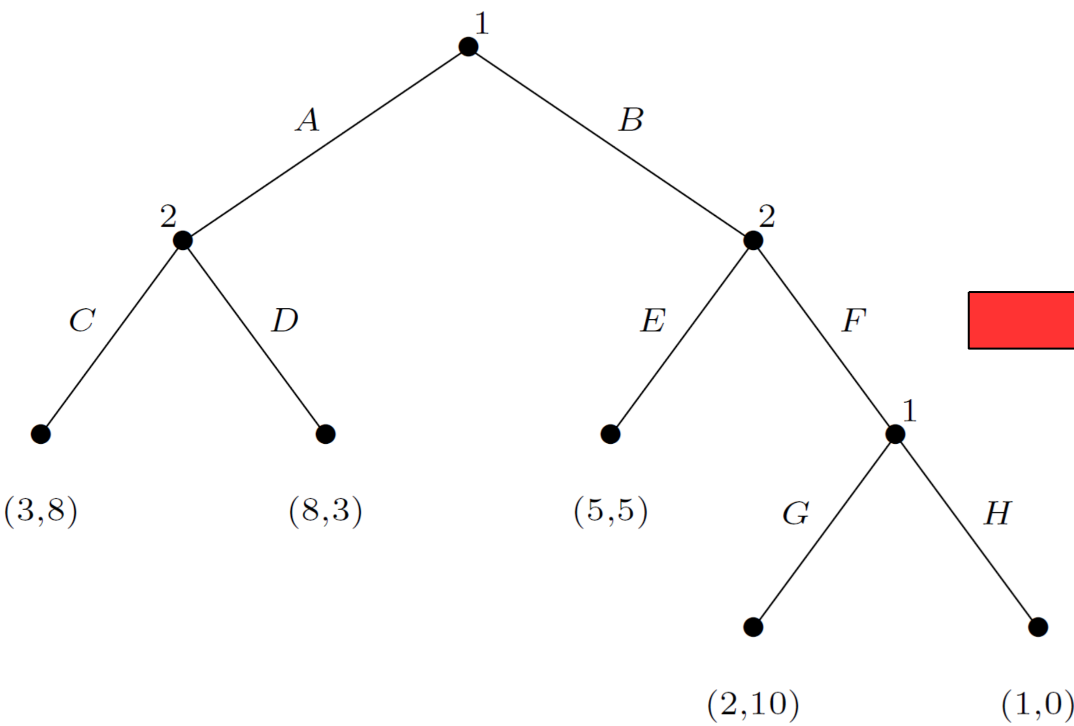

| | (C,E) | (C,F) | (D,E) | (D,F) |
|---|---|---|---|---|
| (A,G) | 3, 8 | 3, 8 | 8, 3 | 8, 3 |
| (A,H) | 3, 8 | 3, 8 | 8, 3 | 8, 3 |
| (B,G) | 5, 5 | 2, 10 | 5, 5 | 2, 10 |
| (B,H) | 5, 5 | 1, 0 | 5, 5 | 1, 0 |

# Perfect-information games in extesive form

- Given our new denition of pure strategy, we are able to reuse our old denitions of:
  - mixed strategies
  - best response
  - Nash equilibrium

# Perfect-information games in extesive form

- The temporal structure can result in a certain redundancy within the normal form
  - different # of outcomes: 5 vs. 16




| | (C,E) | (C,F) | (D,E) | (D,F) |
|---|---|---|---|---|
| (A,G) | 3, 8 | 3, 8 | 8, 3 | 8, 3 |
| (A,H) | 3, 8 | 3, 8 | 8, 3 | 8, 3 |
| (B,G) | 5, 5 | 2, 10 | 5, 5 | 2, 10 |
| (B,H) | 5, 5 | 1, 0 | 5, 5 | 1, 0 |

# Perfect-information games in extesive form

- General lesson
  - While a transformation can always be performed, it can result in an exponential blowup of the game representation
  - This is an important lesson, since the didactic examples of normal-form games are very small, wrongly suggesting that this form is more compact

# Perfect-information games in extesive form

- Can we transform a normal form game into a extensive form game?
  - Not always
    - e.g.: Battle of the Sexes
  - Intuitively, the problem is that perfect-information extensive form games cannot model simultaneity
  - The general characterization of the class of normal-form games for which there exist corresponding perfect-information games in extensive form is somewhat complex

# Perfect-information games in extesive form

- ***Theorem***
  - Every (finite) perfect-information game in extensive form has a pure-strategy Nash equilibrium

# Perfect-information games in extesive form

- This is perhaps the earliest result in game theory, due to Zermelo in 1913
- Intuition: since players take turns, and history is available to everyone, it is never necessary to introduce randomness into action selection in order to find an equilibrium
- Both this intuition and the theorem will cease to hold when we discuss more general classes of games such as imperfect-information games in extensive form

# Sub-game perfect equilibrium

- What are the NE of this game?

| | (C,E) | (C,F) | (D,E) | (D,F) |
|---|---|---|---|---|
| (A,G) | 3, 8 | 3, 8 | 8, 3 | 8, 3 |
| (A,H) | 3, 8 | 3, 8 | 8, 3 | 8, 3 |
| (B,G) | 5, 5 | 2, 10 | 5, 5 | 2, 10 |
| (B,H) | 5, 5 | 1, 0 | 5, 5 | 1, 0 |

# Sub-game perfect equilibrium

- What are the NE of this game?

| | (C, E) | (C, F) | (D, E) | (D, F) |
|---|---|---|---|---|
| (A, G) | 3, 8 | 3, 8 | 8, 3 | 8, 3 |
| (A, H) | 3, 8 | 3, 8 | 8, 3 | 8, 3 |
| (B, G) | 5, 5 | 2, 10 | 5, 5 | 2, 10 |
| (B, H) | 5, 5 | 1, 0 | 5, 5 | 1, 0 |

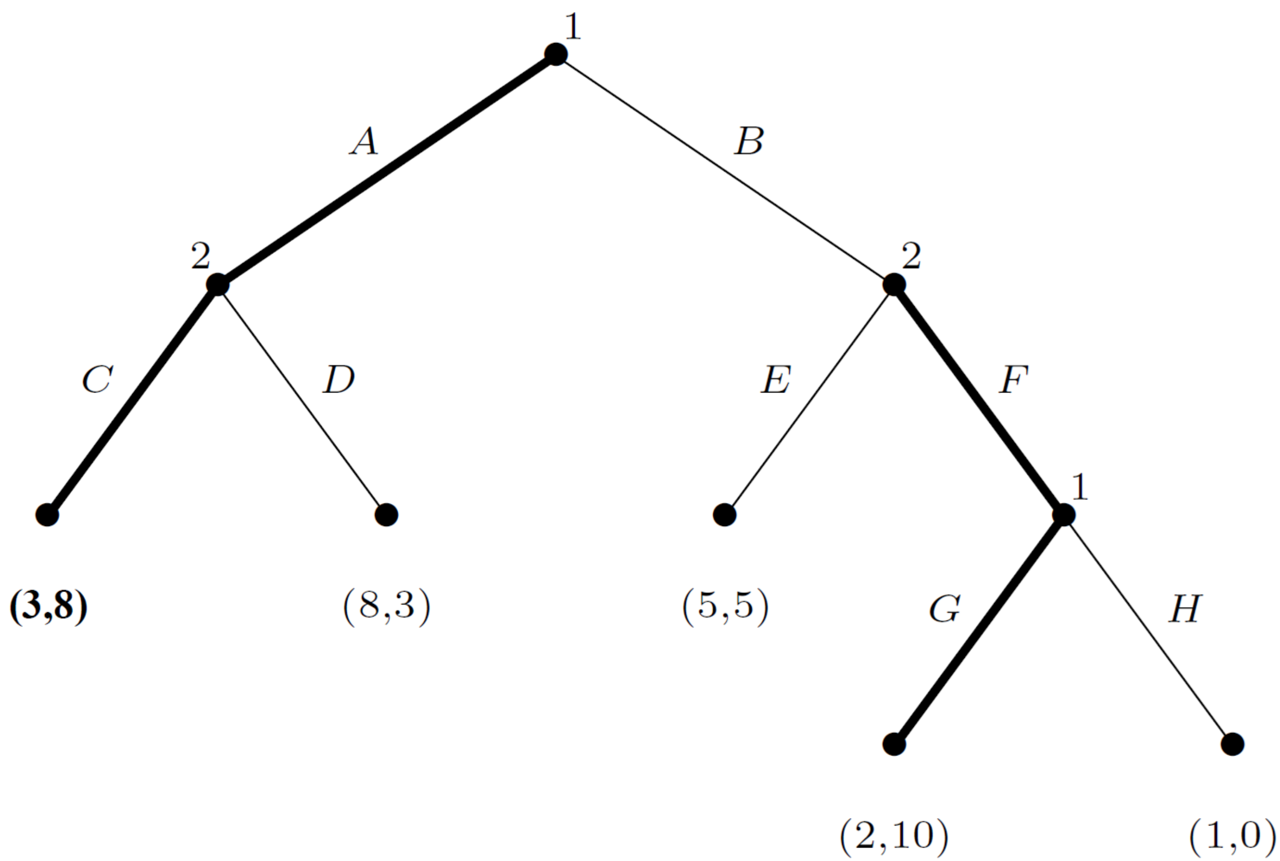
# Sub-game perfect equilibrium

- What are the NE of this game?

| | (C, E) | (C, F) | (D, E) | (D, F) |
|---|---|---|---|---|
| (A, G) | 3, 8 | 3, 8 | 8, 3 | 8, 3 |
| (A, H) | 3, 8 | 3, 8 | 8, 3 | 8, 3 |
| (B, G) | 5, 5 | 2, 10 | 5, 5 | 2, 10 |
| (B, H) | 5, 5 | 1, 0 | 5, 5 | 1, 0 |

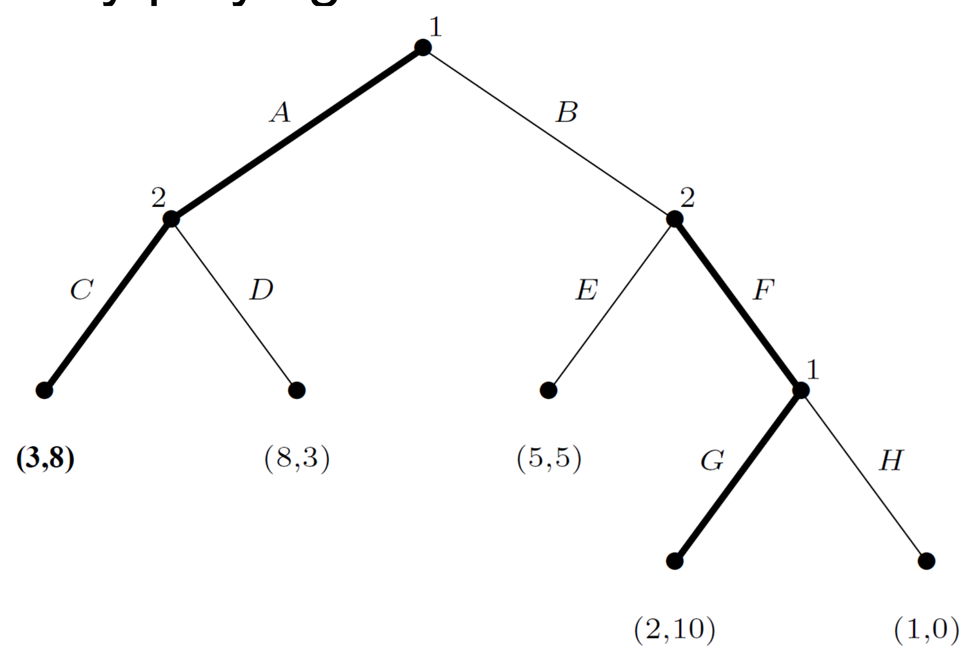
# Sub-game perfect equilibrium

- What are the NE of this game?
  - ***{(A,G), (C,F)}***



# Sub-game perfect equilibrium

- What are the NE of this game?
  - ***{(A,G), (C,F)}***
    - If player ***2*** played ***(C,E)*** rather than ***(C, F)***, then player ***1*** would prefer ***B***
    - as it is, player ***1*** gets a payoff of ***3*** by playing ***A*** rather than a payoff of ***2*** by playing ***B***



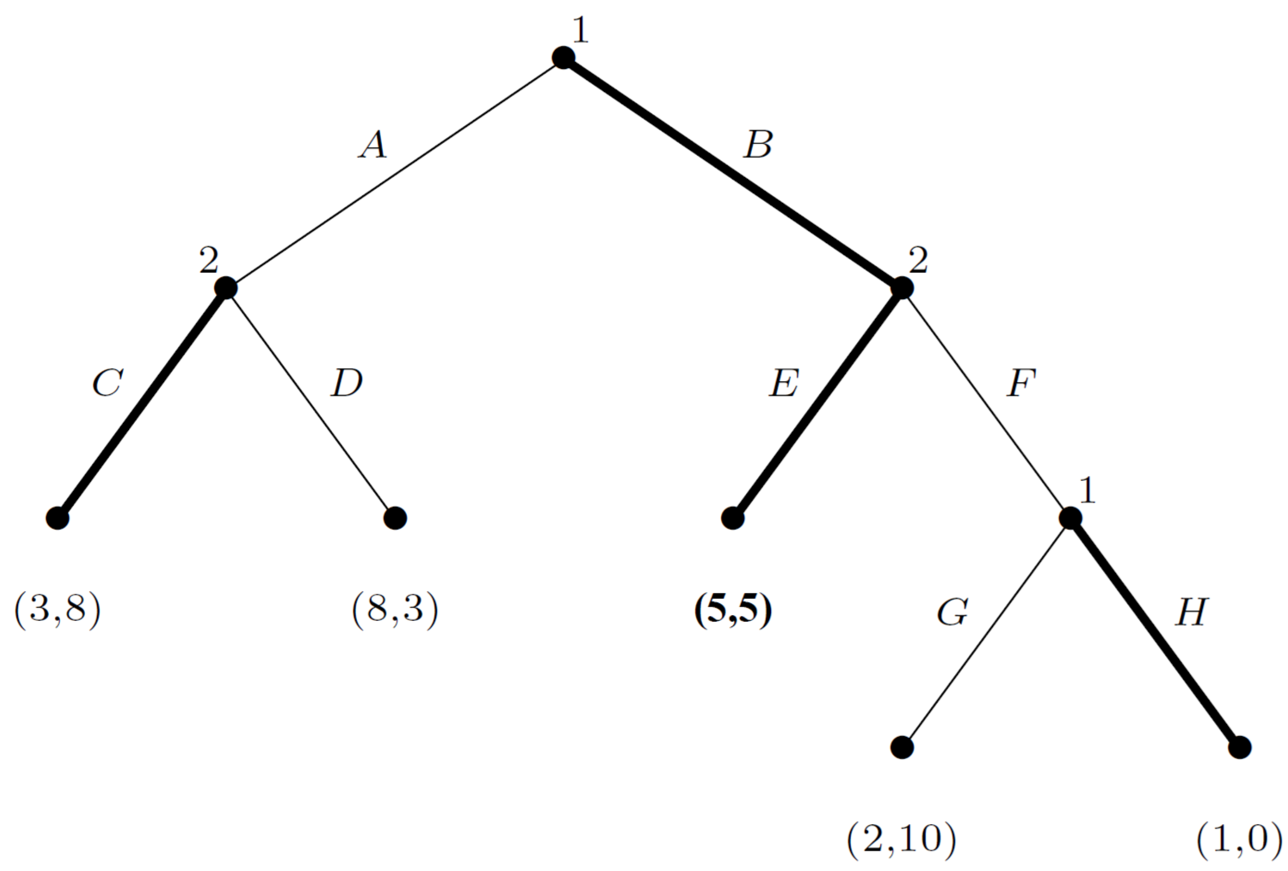
# Sub-game perfect equilibrium

- What are the NE of this game?

| | (C, E) | (C, F) | (D, E) | (D, F) |
|---|---|---|---|---|
| (A, G) | 3, 8 | 3, 8 | 8, 3 | 8, 3 |
| (A, H) | 3, 8 | 3, 8 | 8, 3 | 8, 3 |
| (B, G) | 5, 5 | 2, 10 | 5, 5 | 2, 10 |
| (B, H) | 5, 5 | 1, 0 | 5, 5 | 1, 0 |

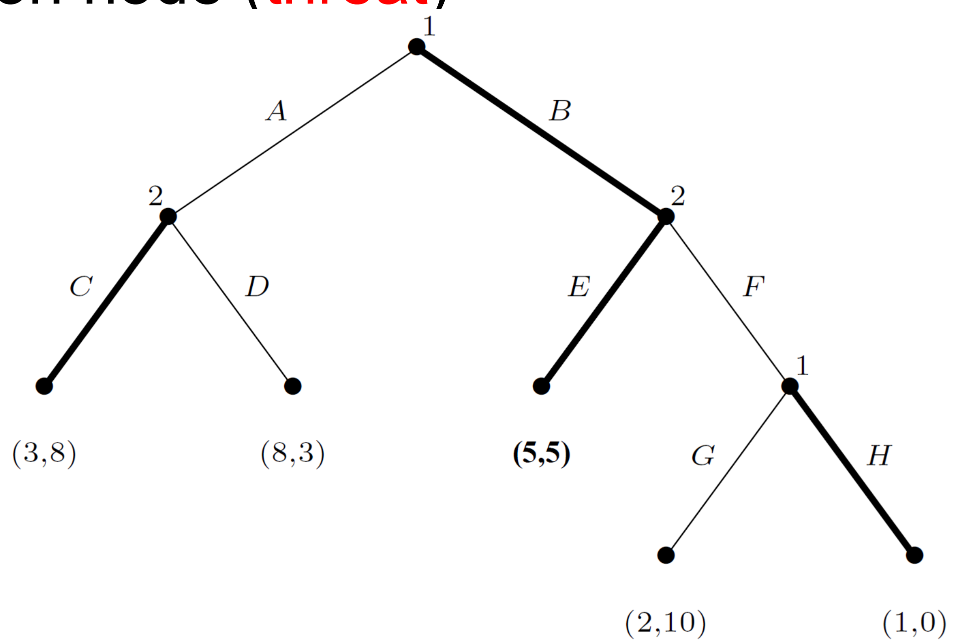
# Sub-game perfect equilibrium

- What are the NE of this game?
  - ***{(B,H), (C,E)}***



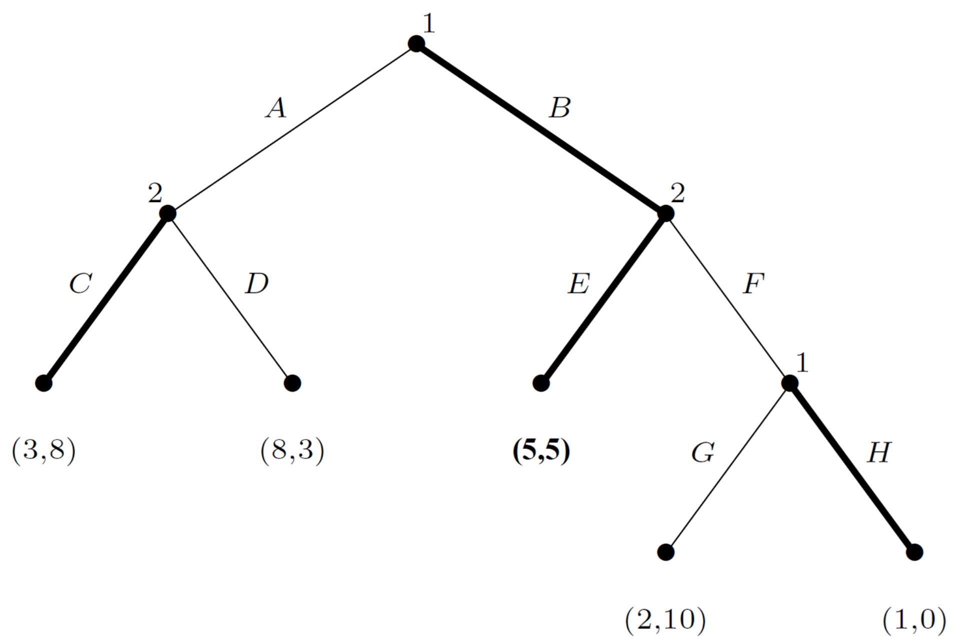
# Sub-game perfect equilibrium

- What are the NE of this game?
  - ***{(B,H), (C,E)}***
    - ***{(B,G), (C,E)}*** is not an equilibrium
    - the only reason that player ***2*** chooses to play the action ***E*** is that he knows that player ***1*** would play ***H*** at his second decision node (threat)



# Sub-game perfect equilibrium

- What are the NE of this game?
  - ***{(B,H), (C,E)}***
    - If player ***2*** played ***F***, would player ***1*** really follow through on his threat and play ***H***, or would he relent and pick ***G*** instead?



# Sub-game perfect equilibrium

- To formally capture the reason why the ***{(B,H), (C,E)}*** equilibrium is unsatisfying, and
- to define an equilibrium refinement concept that does not suffer from this problem:
  - ***Definition***
    - Given a perfect-information extensive-form game ***G***, the subgame of ***G*** rooted at node ***h*** is the restriction of ***G*** to the descendants of ***h***
    - The set of subgames of ***G*** consists of all of subgames of ***G*** rooted at some node in ***G***

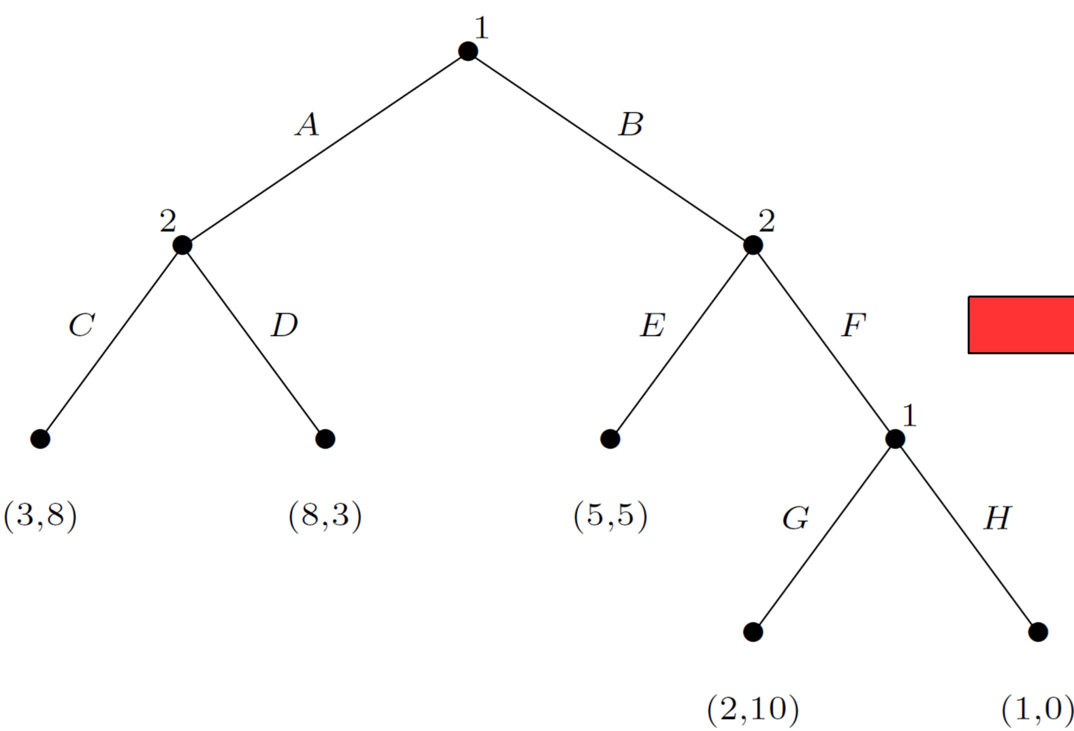
# Sub-game perfect equilibrium

***Definition***

- The subgame-perfect equilibrium (SPE) of a game ***G*** are all strategy profiles ***s*** such that for any subgame ***G'*** of ***G***, the restriction of ***s*** to ***G'*** is a Nash equilibrium of ***G'***
- Since ***G*** is its own subgame, every SPE is also a Nash equilibrium
- SPE is a stronger concept than Nash equilibrium
  - every SPE is a NE, but not every NE is a SPE
- Every perfect-information extensive-form game has at least one SPE
- This definition rules out “noncredible threats”

# Sub-game perfect equilibrium

- What are the SPE?
  - It is not credible that player ***1*** will play ***H***




| | (C, E) | (C, F) | (D, E) | (D, F) |
|---|---|---|---|---|
| (A, G) | 3, 8 | 3, 8 | 8, 3 | 8, 3 |
| (A, H) | 3, 8 | 3, 8 | 8, 3 | 8, 3 |
| (B, G) | 5, 5 | 2, 10 | 5, 5 | 2, 10 |
| (B, H) | 5, 5 | 1, 0 | 5, 5 | 1, 0 |

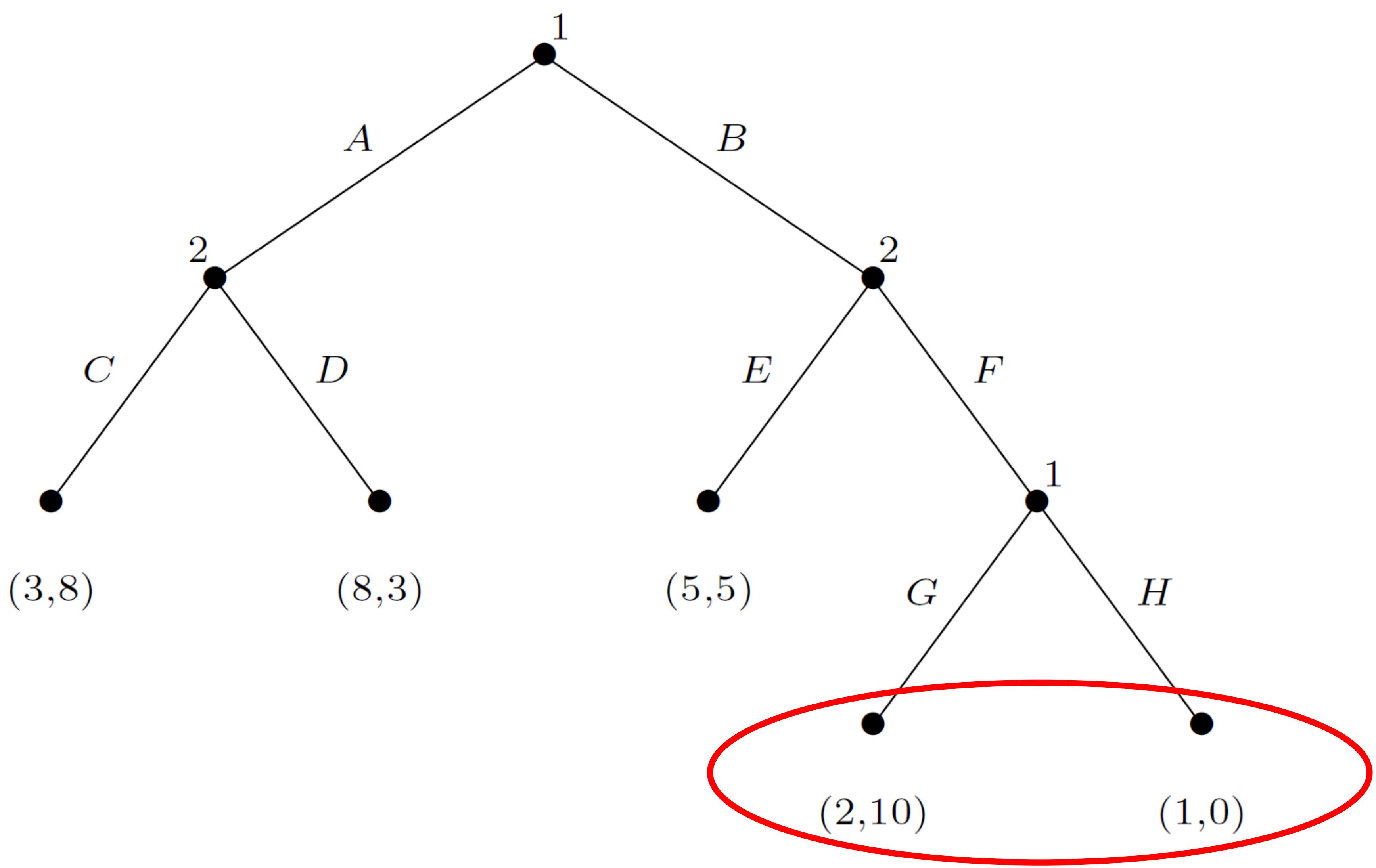
# Computing equilibria: backward induction

- Inherent in the concept of SPE is the principle of **backward induction**
- One identifies the equilibria in the "bottom-most" subgame trees, and assumes that those equilibria will be played as one backs up and considers increasingly larger trees
- We can use this procedure to compute a sample Nash equilibrium
- This is good news: not only are we guaranteed to find a SPE, but also this procedure is computationally simple

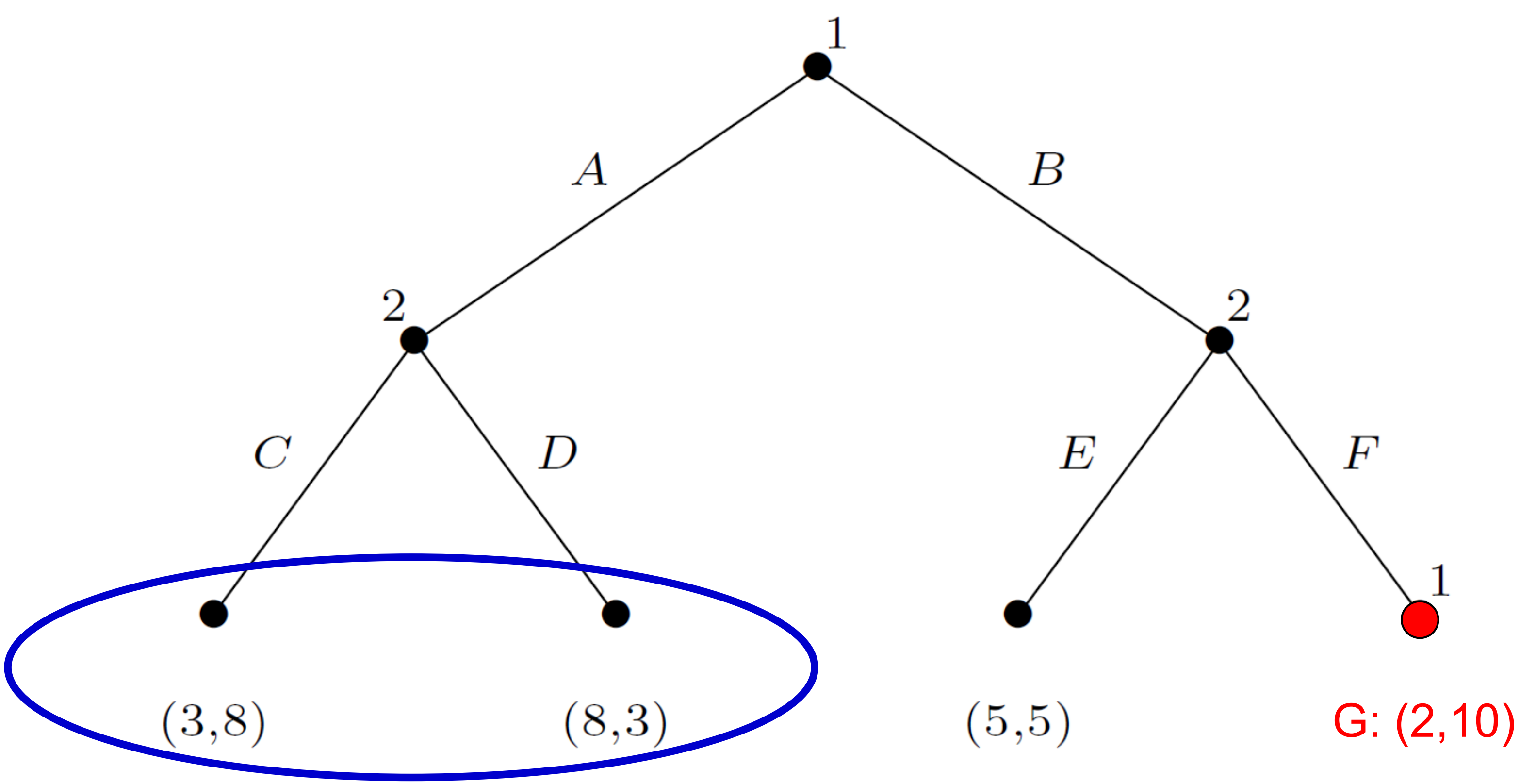
# Computing equilibria: backward induction

- It can be implemented as a single depth-first traversal of the game tree and thus requires time linear in the size of the game representation
- Recall in contrast that the best known methods for finding Nash equilibria of general games require time exponential in the size of the normal form
- The induced normal form of an extensive-form game is exponentially larger than the original representation

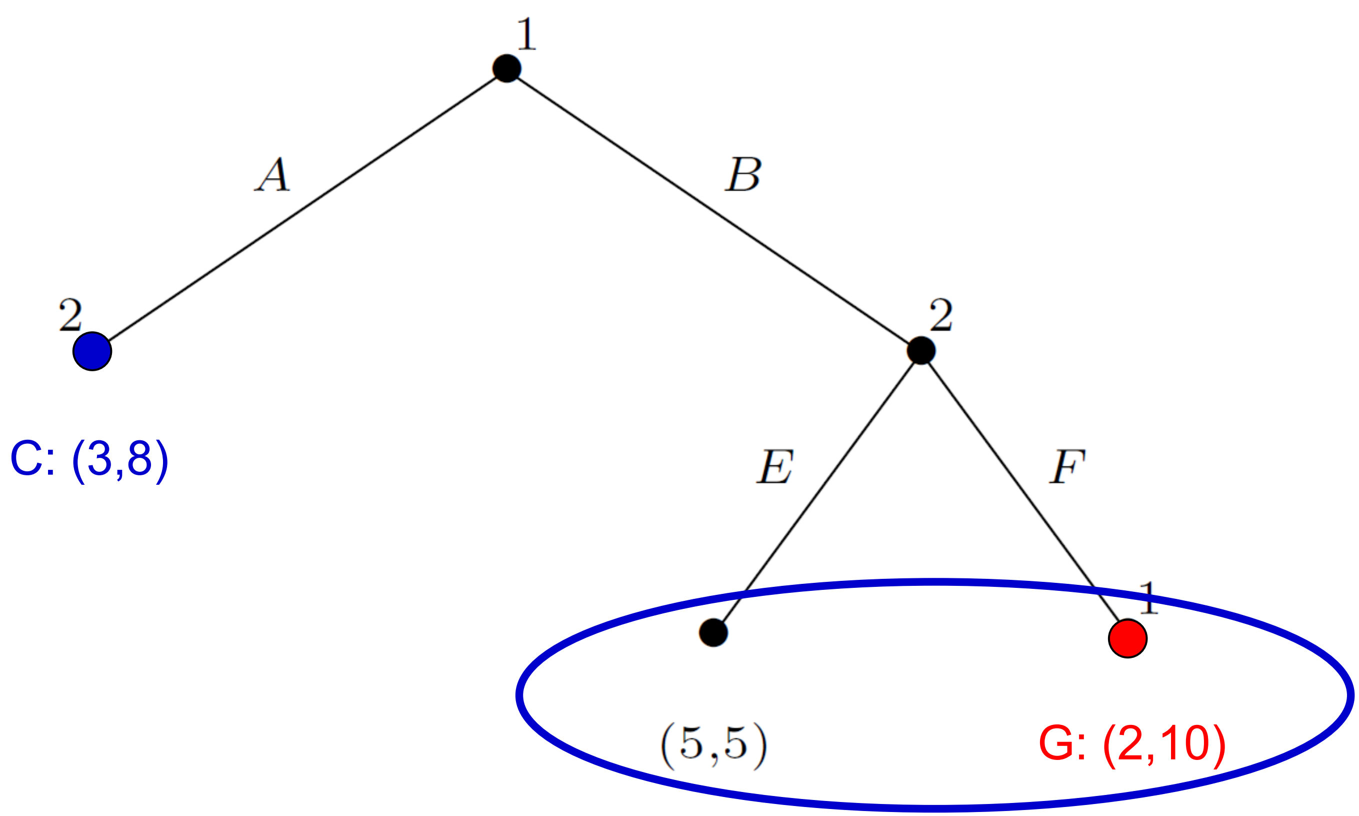
# Computing equilibria: backward induction



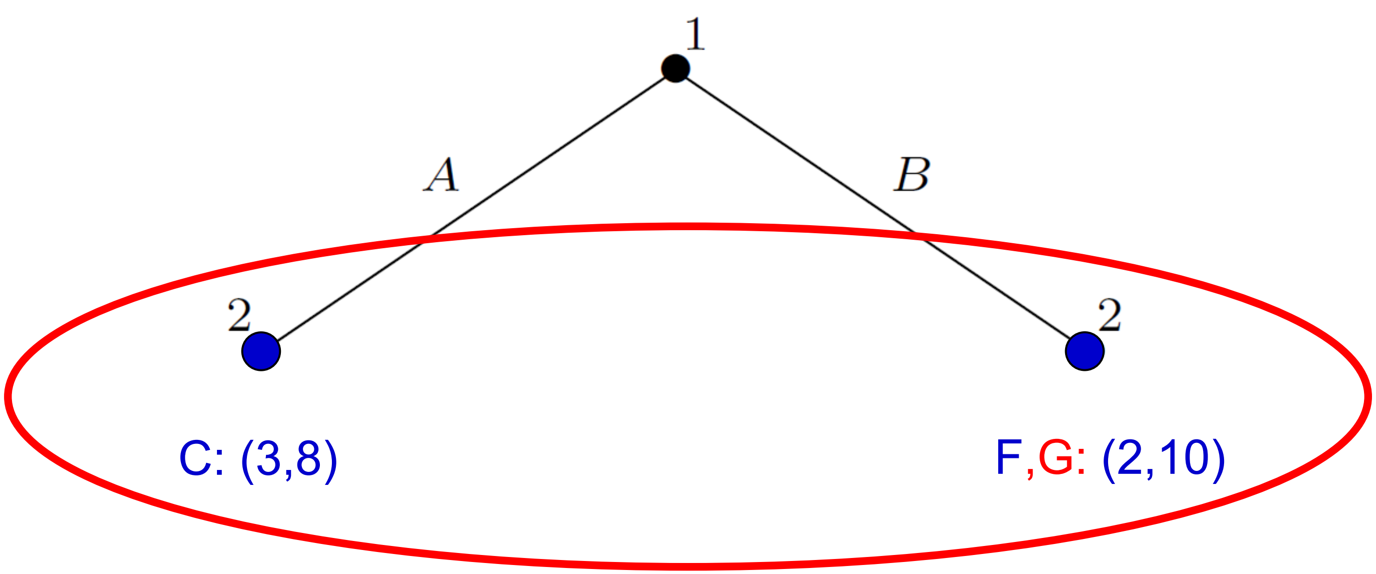
# Computing equilibria: backward induction



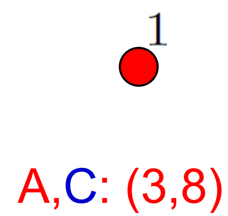
# •Computing equilibria: backward induction



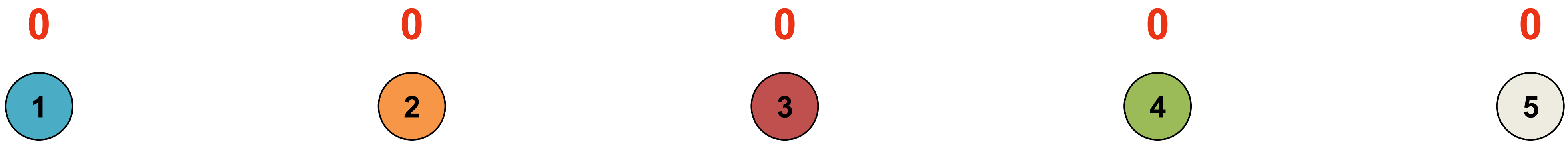
# Computing equilibria: backward induction



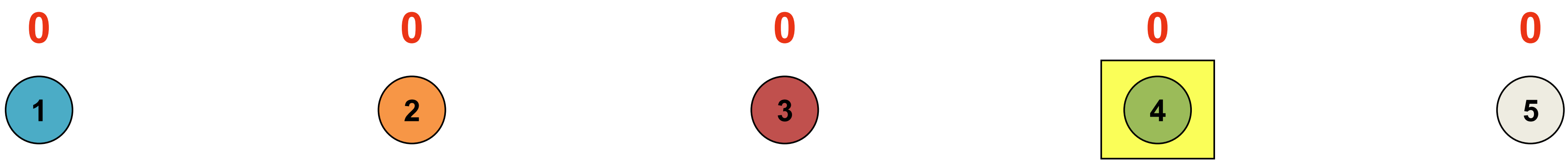
- Computing equilibria: backward induction



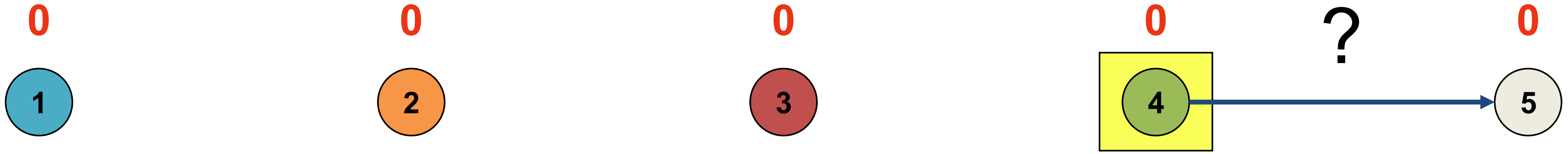
# Five pirates' game



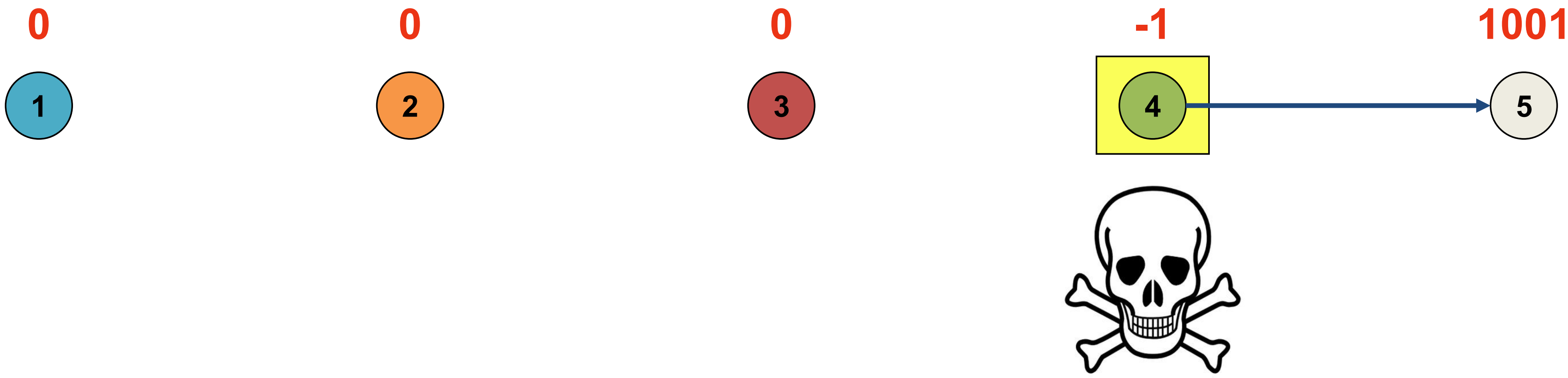
# Five pirates' game



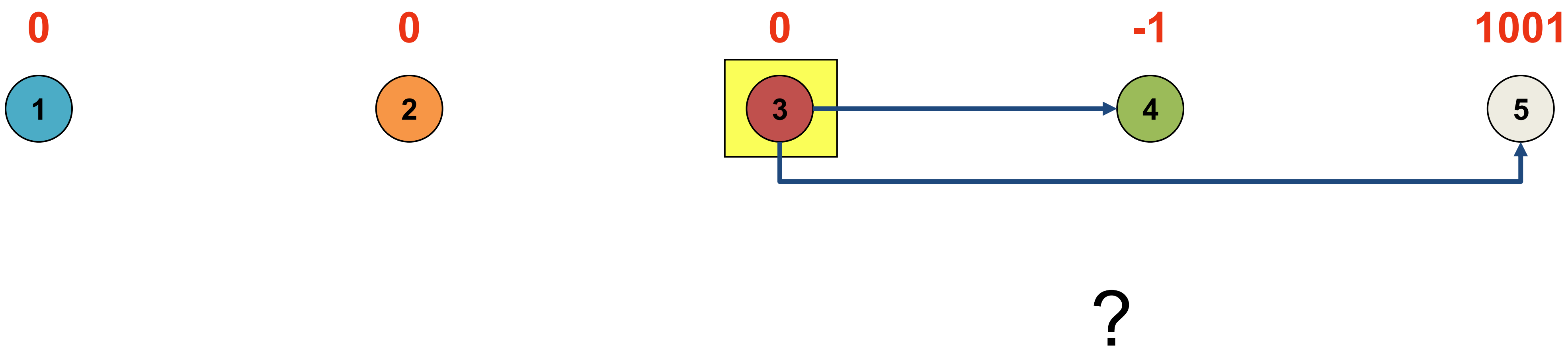
# Five pirates' game



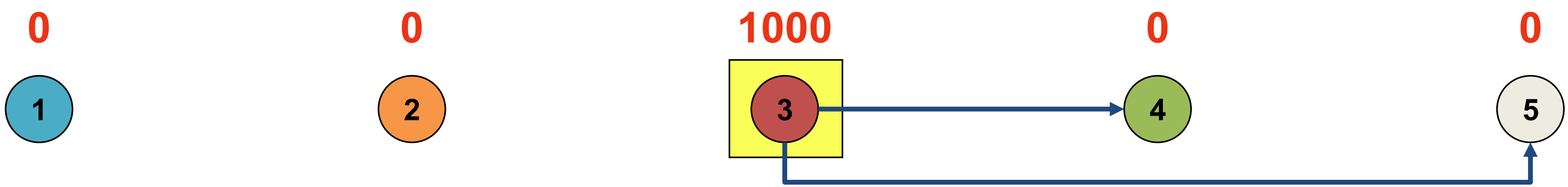
# Five pirates' game



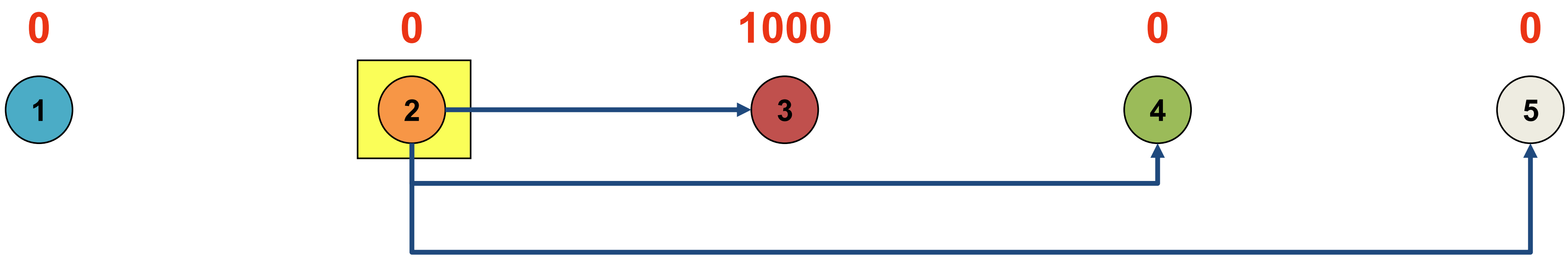
# Five pirates' game



# Five pirates' game

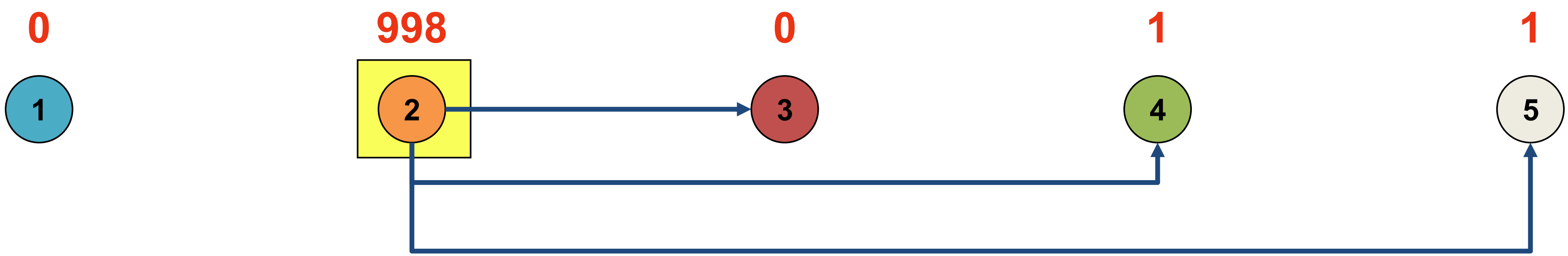


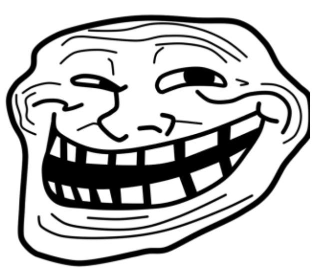

# Five pirates' game



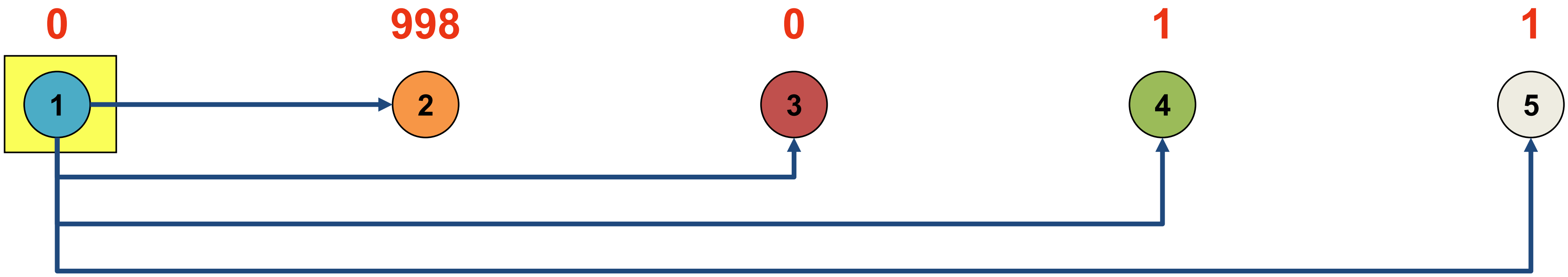

?

# Five pirates' game



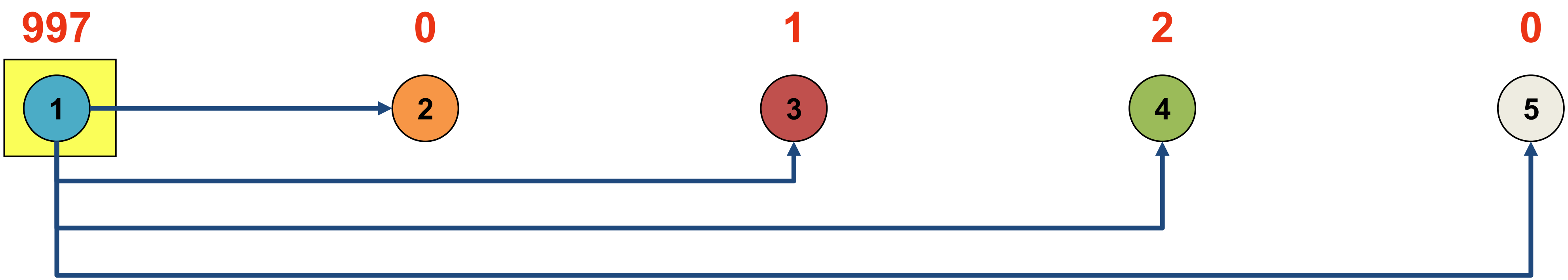



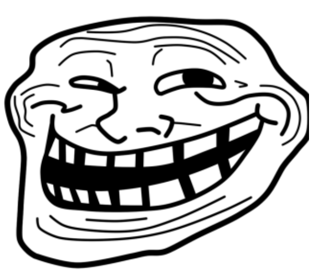
# Five pirates' game




?

# Five pirates' game



# Ultimatum game

Split $10 between you and your partner

- If your partner accepts the split, done
- If your partner rejects, both gets $0

- Oosterbeek, H., Sloof, R., & Van De Kuilen, G. (2004). *Cultural differences in ultimatum game experiments: Evidence from a meta-analysis.* **Experimental economics**, 7(2), 171-188.

| Country | *N* (1) | Mean offer (2) | Mean reject (3) | IDV (4) | PDI (5) | AUTH (6) | TRUST (7) | COMP (8) | GDP pc (9) | GINI index (10) |
|---|---|---|---|---|---|---|---|---|---|---|
| Austria | 1 | 39.21 | 16.10 | 55 | 11 | −0.05 | 0.32 | 6.78 | 12955 | 23.1 |
| Bolivia | 1 | 37.00 | 0.00 | | | | | | 1721 | 42.0 |
| Chile | 1 | 34.00 | 6.70 | 23 | 63 | 1.10 | 0.23 | 5.94 | 4890 | 56.5 |
| Ecuador | 2 | 34.50 | 7.50 | 8 | 78 | | | | 2830 | 46.6 |
| France | 3 | 40.24 | 30.78 | 71 | 68 | −0.15 | 0.23 | 5.97 | 13918 | 32.7 |
| Germany | 1 | 36.70 | 9.52 | 67 | 35 | −1.30 | 0.38 | 6.75 | 11666 | 30.0 |
| Honduras | 1 | 45.70 | 23.05 | | | | | | 1385 | 53.7 |
| Indonesia | 4 | 46.63 | 14.63 | 14 | 78 | | | | 2102 | 36.5 |
| Israel | 5 | 41.71 | 17.73 | 54 | 13 | | | | 9843 | 35.5 |
| Japan | 3 | 44.73 | 19.27 | 46 | 54 | −1.58 | 0.42 | 5.52 | 15105 | 24.9 |
| Yugoslavia | 1 | 44.33 | 26.67 | 27 | 76 | −0.65 | 0.30 | 7.07 | 4548 | 31.9 |
| Kenya | 1 | 44.00 | 4.00 | 27 | 64 | | | | 914 | 57.5 |
| Mongolia | 2 | 35.50 | 5.00 | | | | | | 1842 | 33.2 |
| Netherlands | 2 | 42.25 | 9.24 | 80 | 38 | −0.55 | 0.56 | 5.60 | 13281 | 31.5 |
| Papua New-Guinea | 2 | 40.50 | 33.50 | | | | | | 1606 | 50.9 |
| Paraguay | 1 | 51.00 | 0.00 | | | | | | 2178 | 59.1 |
| Peru | 1 | 26.00 | 4.80 | 16 | 64 | 1.75 | 0.05 | 6.54 | 2092 | 46.2 |
| Romania | 2 | 36.95 | 23.50 | | | | 0.16 | 7.32 | 2043 | 28.2 |
| Slovakia | 3 | 43.17 | 12.67 | | | −0.55 | 0.23 | 6.97 | 4095 | 19.5 |
| Spain | 1 | 26.66 | 29.17 | 51 | 57 | 0.60 | 0.34 | 5.70 | 9802 | 38.5 |
| Sweden | 1 | 35.23 | 18.18 | 71 | 31 | −1.35 | 0.66 | 6.78 | 13986 | 25.0 |
| Tanzania | 4 | 37.50 | 19.25 | 27 | 64 | | | | 534 | 38.2 |
| UK | 2 | 34.33 | 23.38 | 89 | 35 | 0.10 | 0.44 | 6.19 | 12724 | 32.6 |
| US East | 22 | 40.54 | 17.15 | 91 | 40 | 1.11 | 0.50 | 6.70 | 17945 | 40.1 |
| US West | 6 | 42.64 | 9.41 | 91 | 40 | 1.11 | 0.50 | 6.70 | 17945 | 40.1 |
| Zimbabwe | 2 | 43.00 | 8.50 | | | | | | 1162 | 56.8 |

# Computing equilibria: backward induction

**function** BACKWARDINDUCTION (node $h$) **returns** $u(h)$
**if** $h \in Z$ **then**
  **return** $u(h)$ // $h$ is a terminal node
$best\_util \leftarrow -\infty$
**forall** $a \in \chi(h)$ **do**
  $util\_at\_child \leftarrow$ BACKWARDINDUCTION$(\sigma(h, a))$
  **if** $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**
    $best\_util \leftarrow util\_at\_child$
**return** $best\_util$

for each available action ***a*** at non-terminal node ***h***

# Computing equilibria: backward induction

**function** $\textsc{BackwardInduction}$ (node $h$) **returns** $u(h)$
**if** $h \in Z$ **then**
  **return** $u(h)$    // $h$ is a terminal node
$best\_util \leftarrow -\infty$
**forall** $a \in \chi(h)$ **do**
  $util\_at\_child \leftarrow \textsc{BackwardInduction}(\sigma(h, a))$
  **if** $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**
    $best\_util \leftarrow util\_at\_child$
**return** $best\_util$

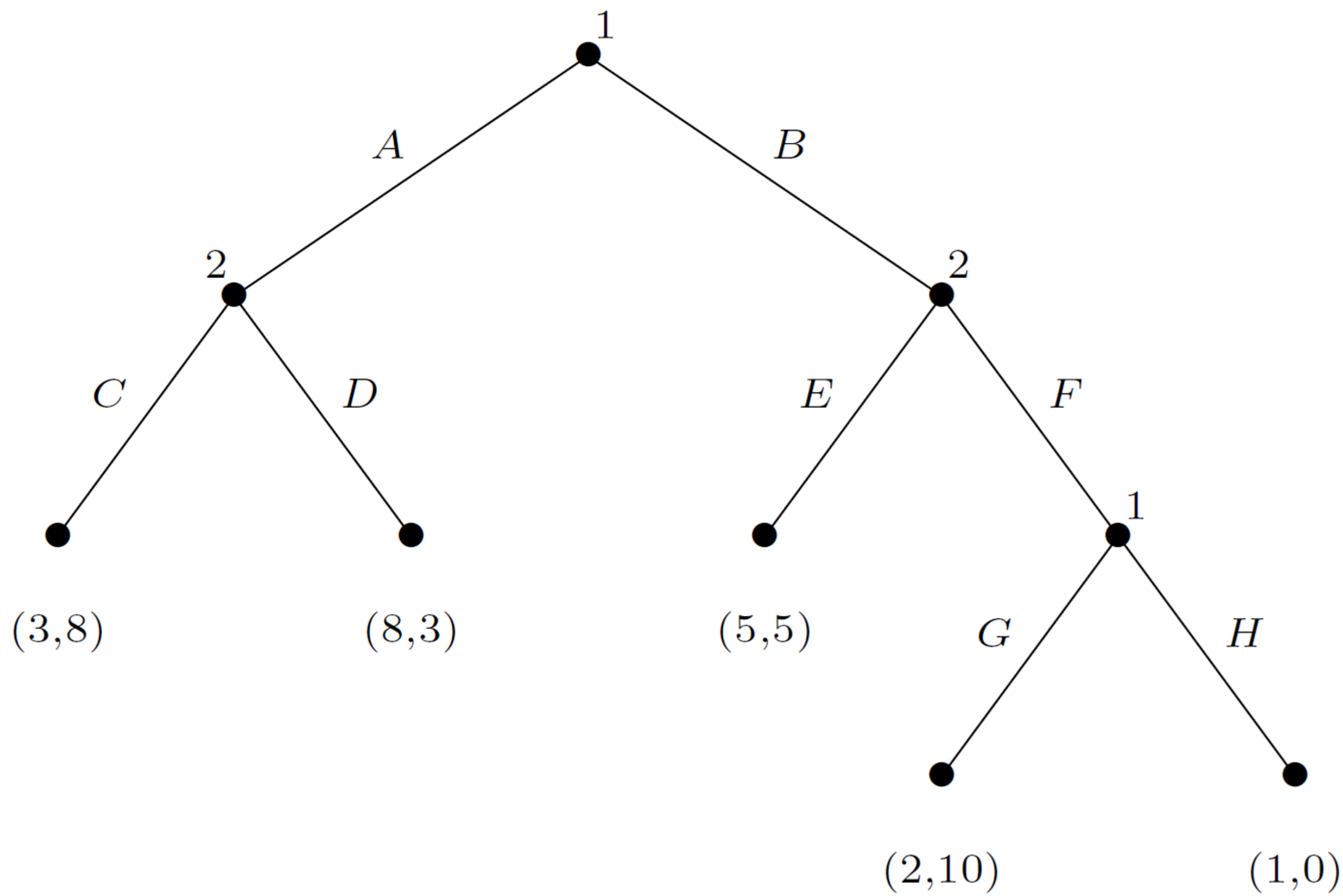
recursively fetch the utility for player ***ρ(h)*** at child node ***σ(h,a)***

# Computing equilibria: backward induction

**function** BACKWARDINDUCTION (node $h$) **returns** $u(h)$
**if** $h \in Z$ **then**
  **return** $u(h)$ // $h$ is a terminal node
$best\_util \leftarrow -\infty$
**forall** $a \in \chi(h)$ **do**
  $util\_at\_child \leftarrow$ BACKWARDINDUCTION$(\sigma(h, a))$
  **if** $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**
    $best\_util \leftarrow util\_at\_child$
**return** $best\_util$

returns the highest possible utility for player ***ρ(h)*** at node ***h***

**function** BACKWARDINDUCTION (node $h$) **returns** $u(h)$
**if** $h \in Z$ **then**
  **return** $u(h)$ // $h$ is a terminal node
$best\_util \leftarrow -\infty$
**forall** $a \in \chi(h)$ **do**
  $util_at_child \leftarrow$BACKWARDINDUCTION$(\sigma(h, a))$
  **if** $util\_at\_child_{\rho(h)} > best\_util_{\rho(h)}$ **then**
    $best\_util \leftarrow util\_at\_child$
**return** $best\_util$



# Computing equilibria: backward induction

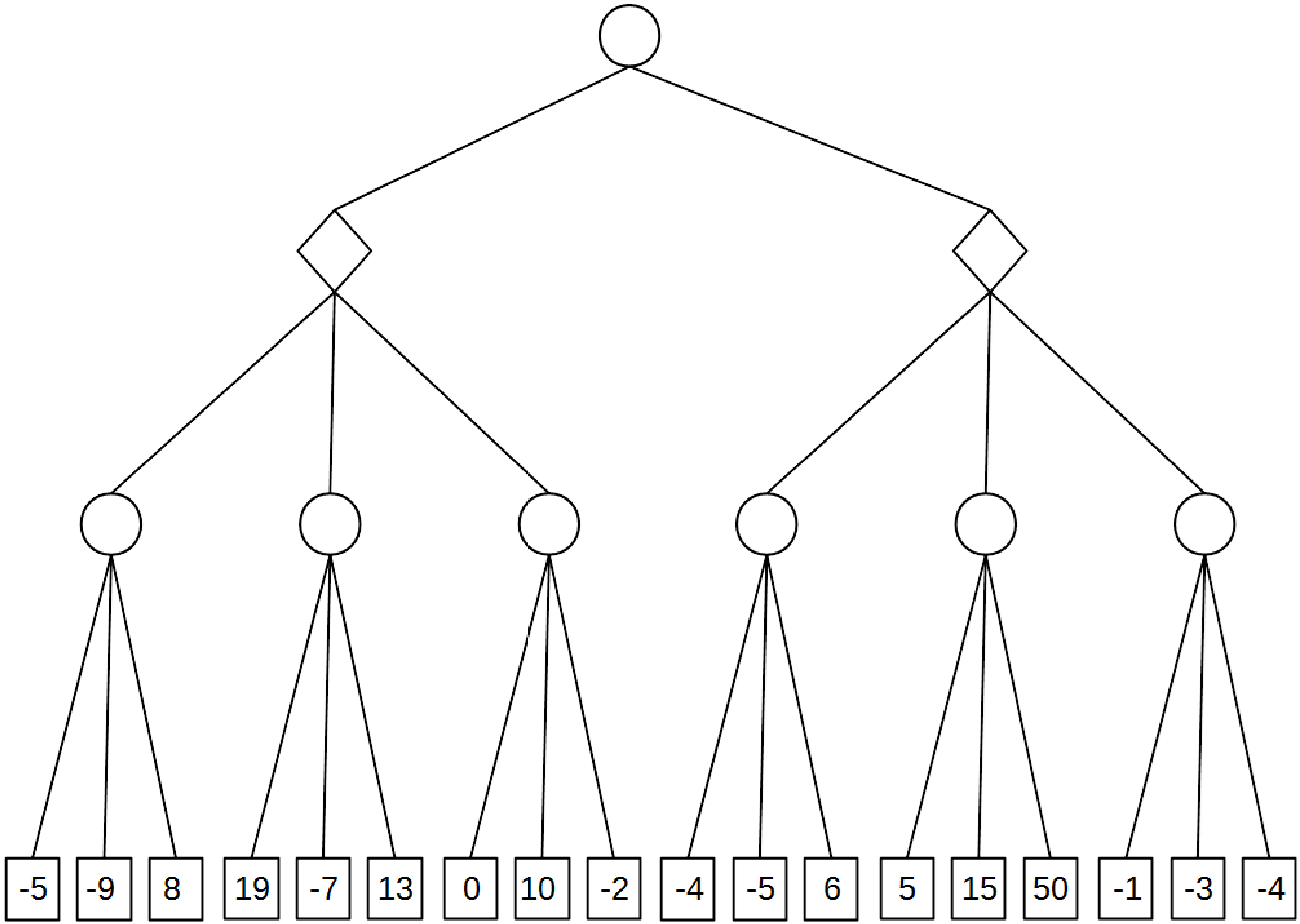
- Demonstrates that in principle a sample SPE is effectively computable
- However, in practice many game trees are not enumerated in advance and are hence unavailable for backward induction
- For example, the extensive-form representation of chess has around $10^{150}$ nodes, which is vastly too large to represent explicitly

# Computing equilibria: backward induction

- For such games it is more common to discuss
  - the size of the game tree in terms of the average branching factor ***b*** (the average number of actions which are possible at each node)
  - a maximum depth ***m*** (the maximum number of sequential actions)
- A procedure which requires time linear in the size of the representation thus expands $O(b^m)$ nodes
- Unfortunately, we can do no better than this on arbitrary perfect-information games

# Two-player, zero-sum games: minimax and alpha-beta pruning

- BACKWARDINDUCTION has another name in the two-player, zero-sum context: the minimax algorithm
  - In such games, only a single payoff number is required to characterize any outcome
  - **Player 1** wants to maximize this number, while **player 2** wants to minimize it
  - Propagates these single payoff numbers from the leaves up to the root


-5
-9
8
19
-7
13
0
10
-2
-4
-5
6
5
15
50
-1
-3
-4

# Two-player, zero-sum games: minimax and alpha-beta pruning

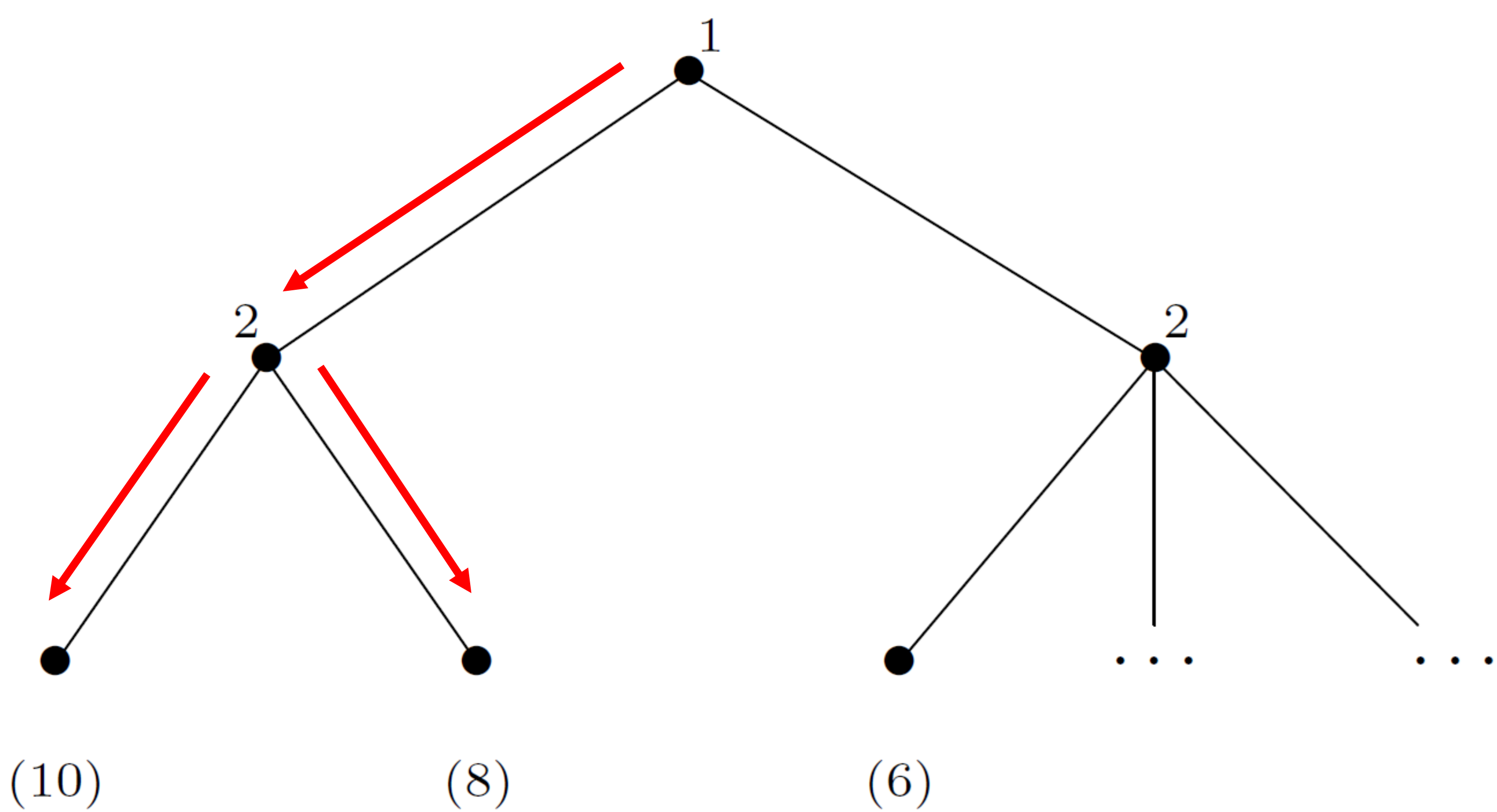
- Each decision node for player 1 is labeled with the maximum of the labels of its child nodes
- Each decision node for player 2 is labeled with the minimum of that node's children's labels
- The label on the root node is the value of the game
  - player 1's payoff in equilibrium

# Two-player, zero-sum games: minimax and alpha-beta pruning

- How can we improve on the minimax algorithm?
- The fact that player 1 and player 2 always have strictly opposing interests means that we can prune away some parts of the game tree
- We can recognize that certain subtrees will never be reached in equilibrium, even without examining the nodes in these subtrees
- This leads us to a new algorithm called **ALPHABETAPRUNING**

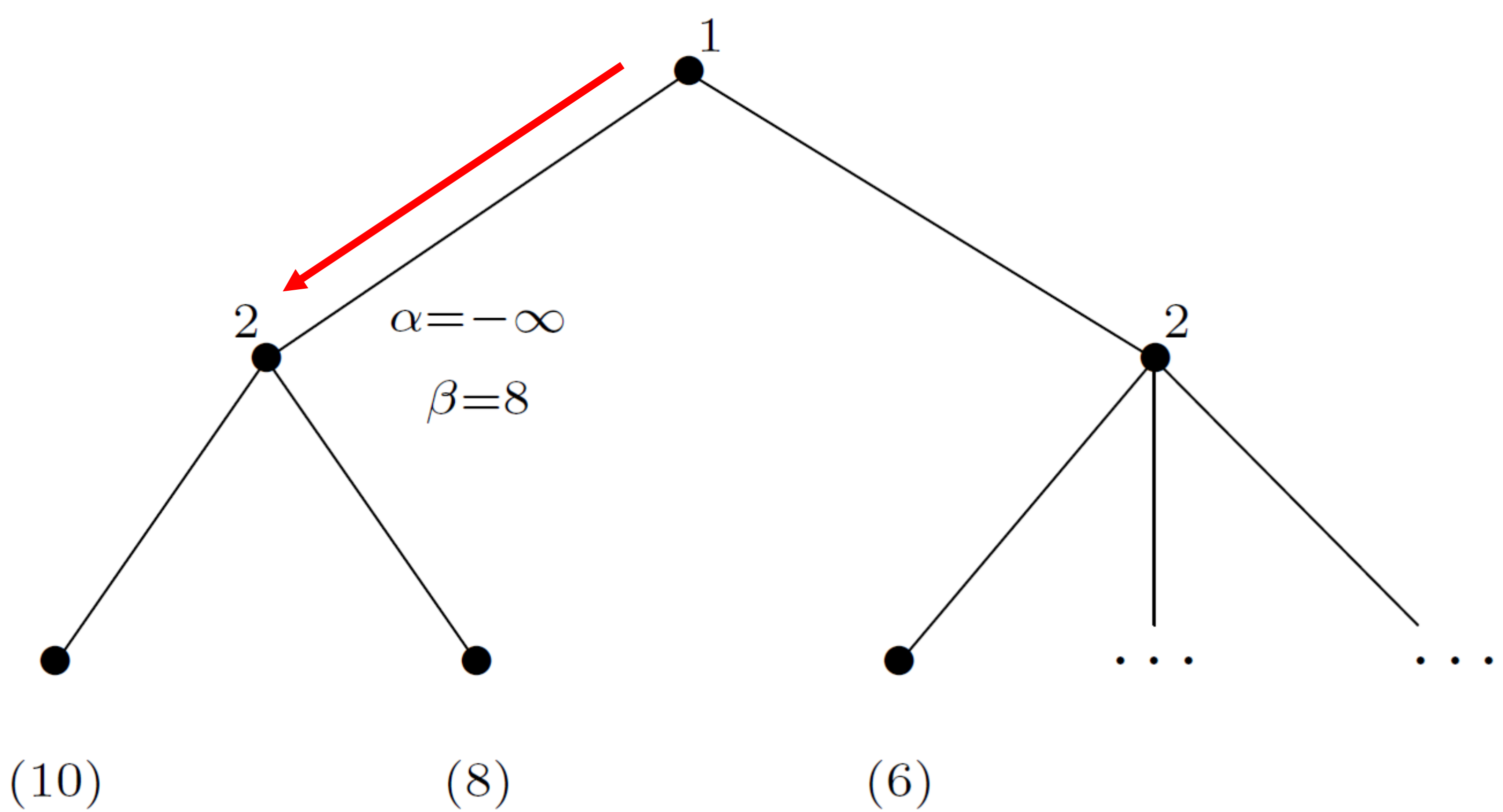
# Two-player, zero-sum games: minimax and alpha-beta pruning

At each node ***h*** either ***α*** or ***β*** is updated with the value of the previously encountered node that their corresponding player (player ***1*** for ***α*** and player ***2*** for ***β***) would most prefer to choose instead of ***h***



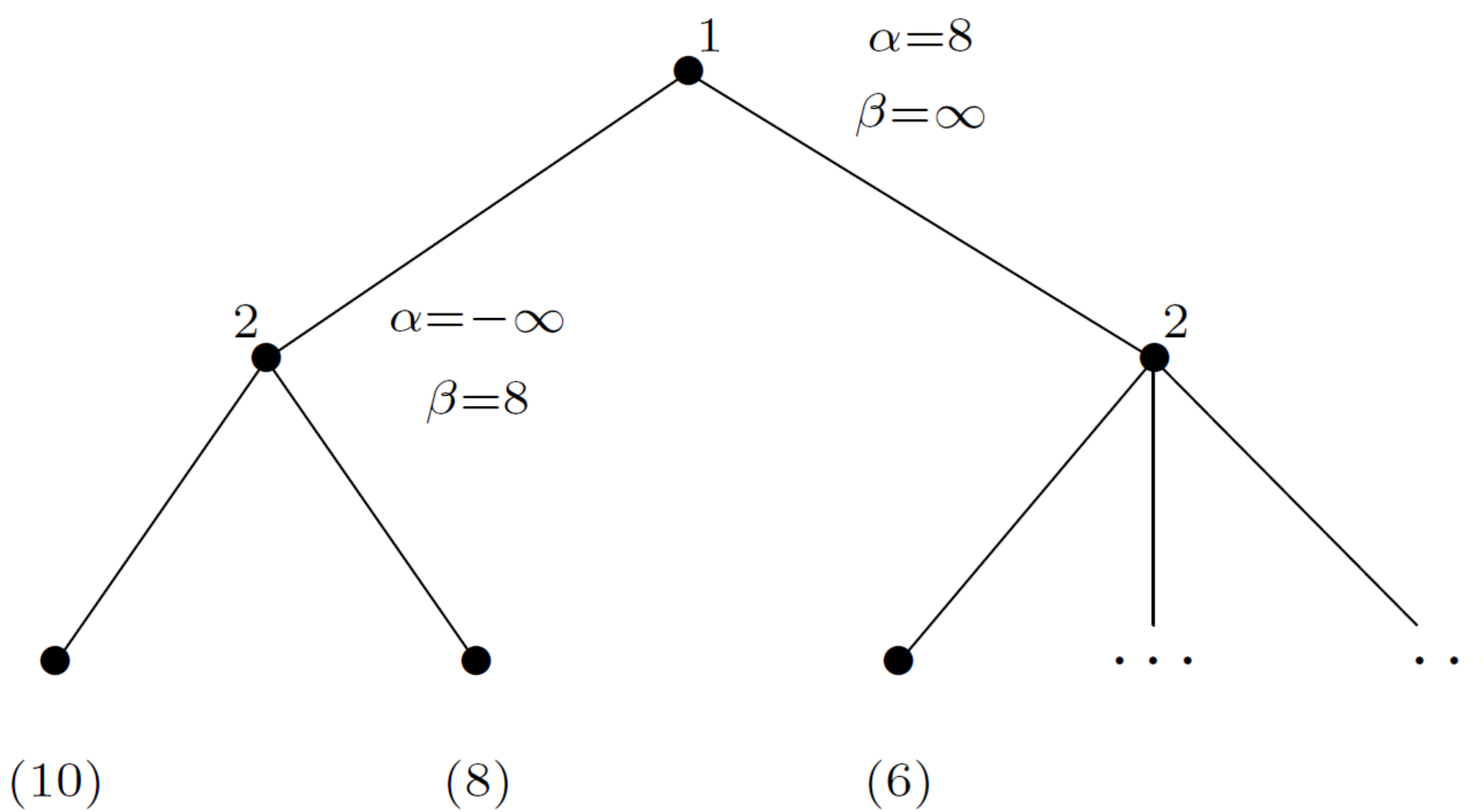
# Two-player, zero-sum games: minimax and alpha-beta pruning

At each node ***h*** either ***α*** or ***β*** is updated with the value of the previously encountered node that their corresponding player (player ***1*** for ***α*** and player ***2*** for ***β***) would most prefer to choose instead of ***h***



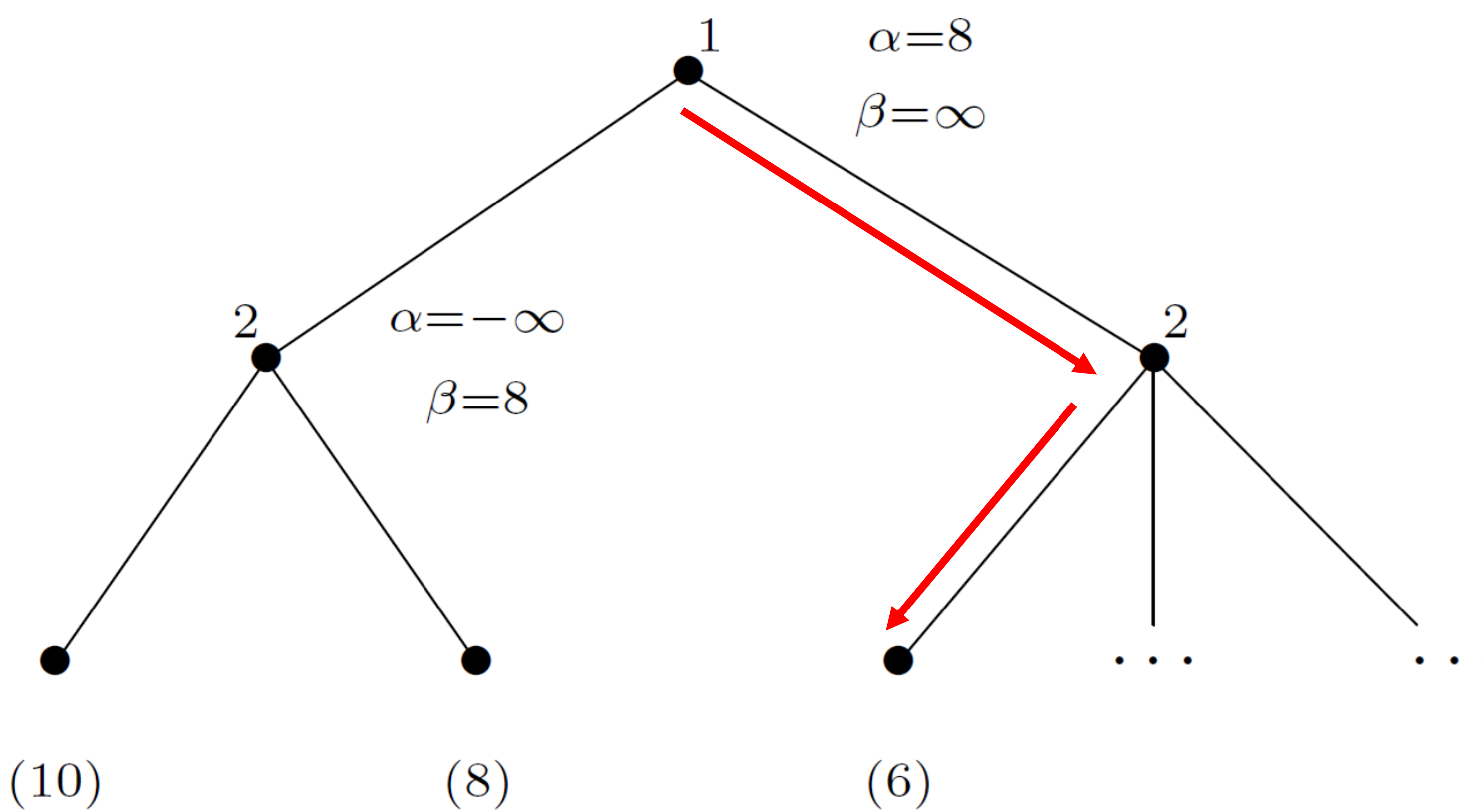
# Two-player, zero-sum games: minimax and alpha-beta pruning

At each node ***h*** either ***α*** or ***β*** is updated with the value of the previously encountered node that their corresponding player (player ***1*** for ***α*** and player ***2*** for ***β***) would most prefer to choose instead of ***h***



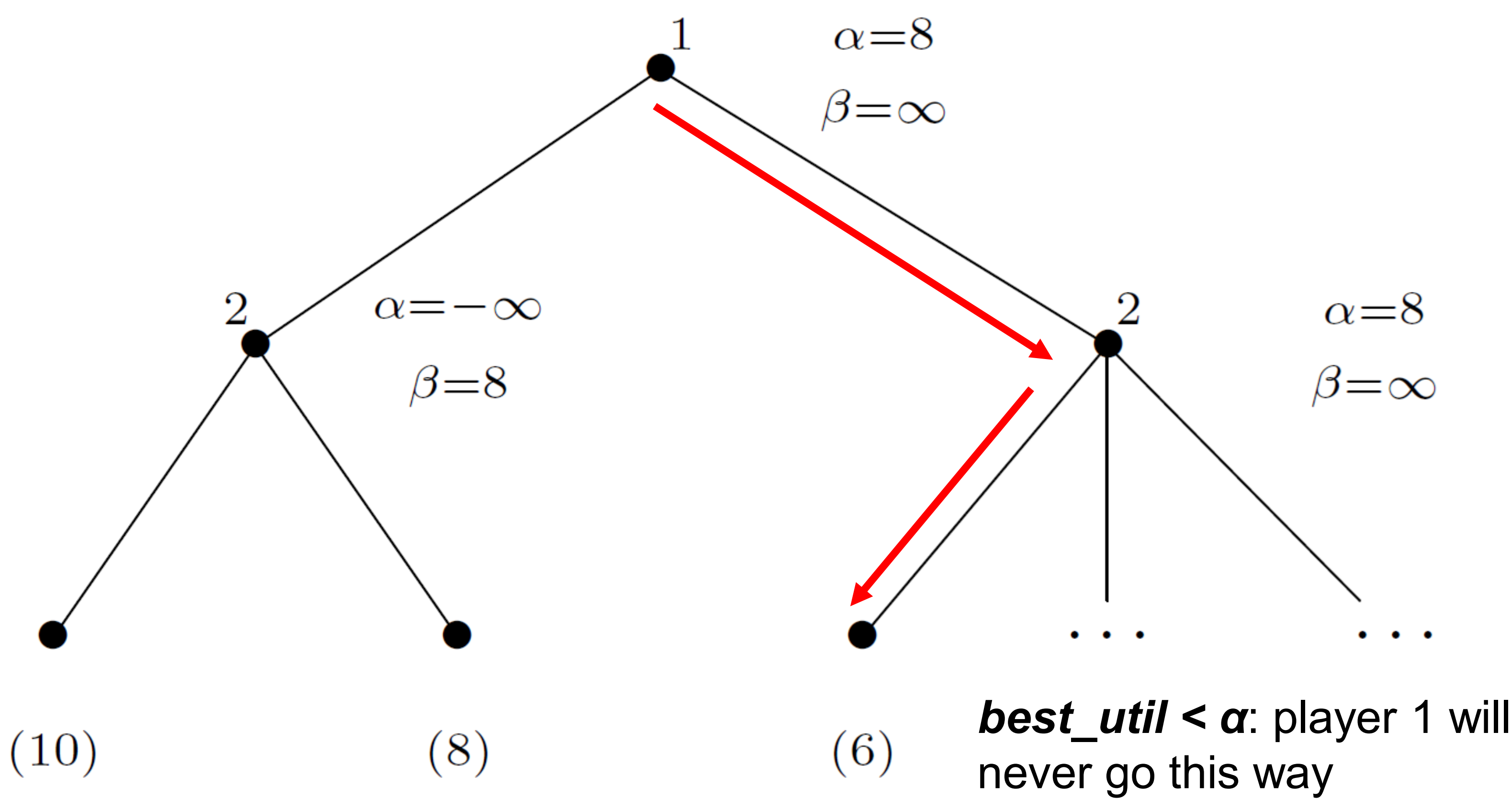
# Two-player, zero-sum games: minimax and alpha-beta pruning

At each node ***h*** either ***α*** or ***β*** is updated with the value of the previously encountered node that their corresponding player (player ***1*** for ***α*** and player ***2*** for ***β***) would most prefer to choose instead of ***h***



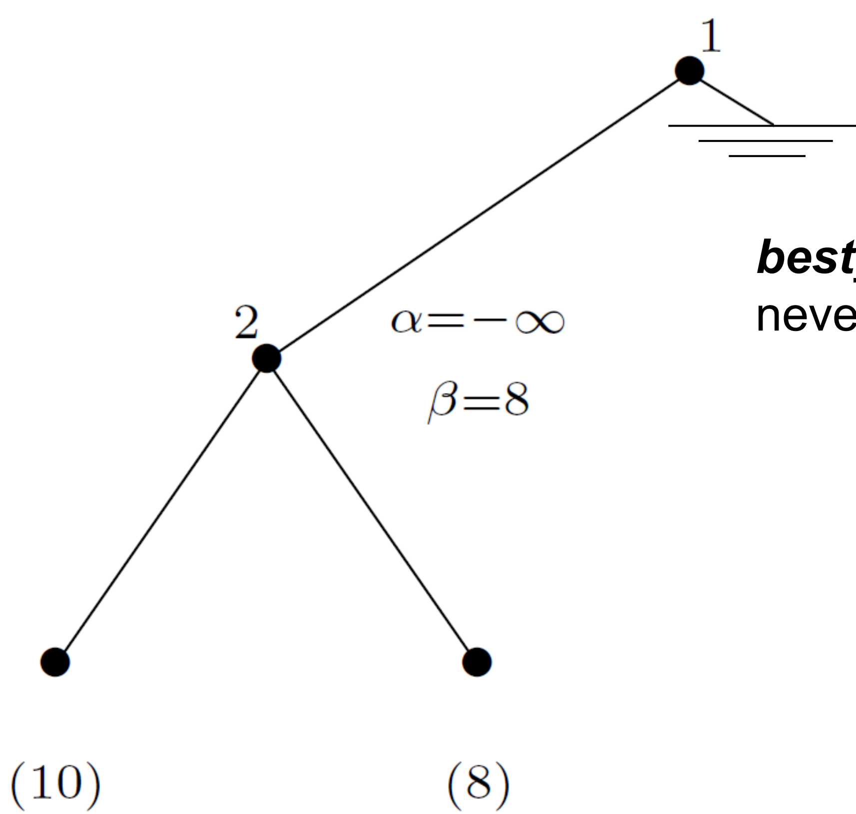
# Two-player, zero-sum games: minimax and alpha-beta pruning

At each node ***h*** either ***α*** or ***β*** is updated with the value of the previously encountered node that their corresponding player (player ***1*** for ***α*** and player ***2*** for ***β***) would most prefer to choose instead of ***h***



# Two-player, zero-sum games: minimax and alpha-beta pruning

At each node ***h*** either ***α*** or ***β*** is updated with the value of the previously encountered node that their corresponding player (player ***1*** for ***α*** and player ***2*** for ***β***) would most prefer to choose instead of ***h***




***best_util < α***: player 1 will never go this way

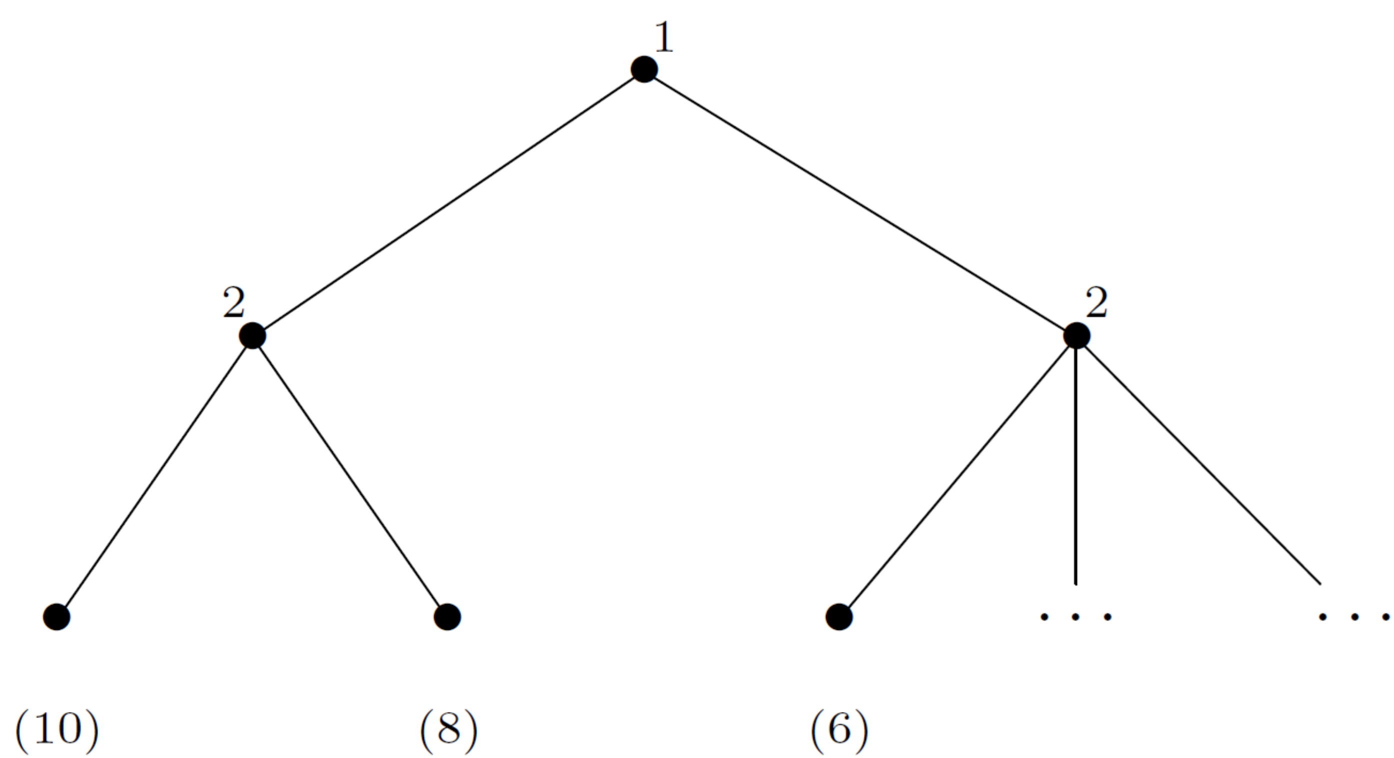
# Two-player, zero-sum games: minimax and alpha-beta pruning

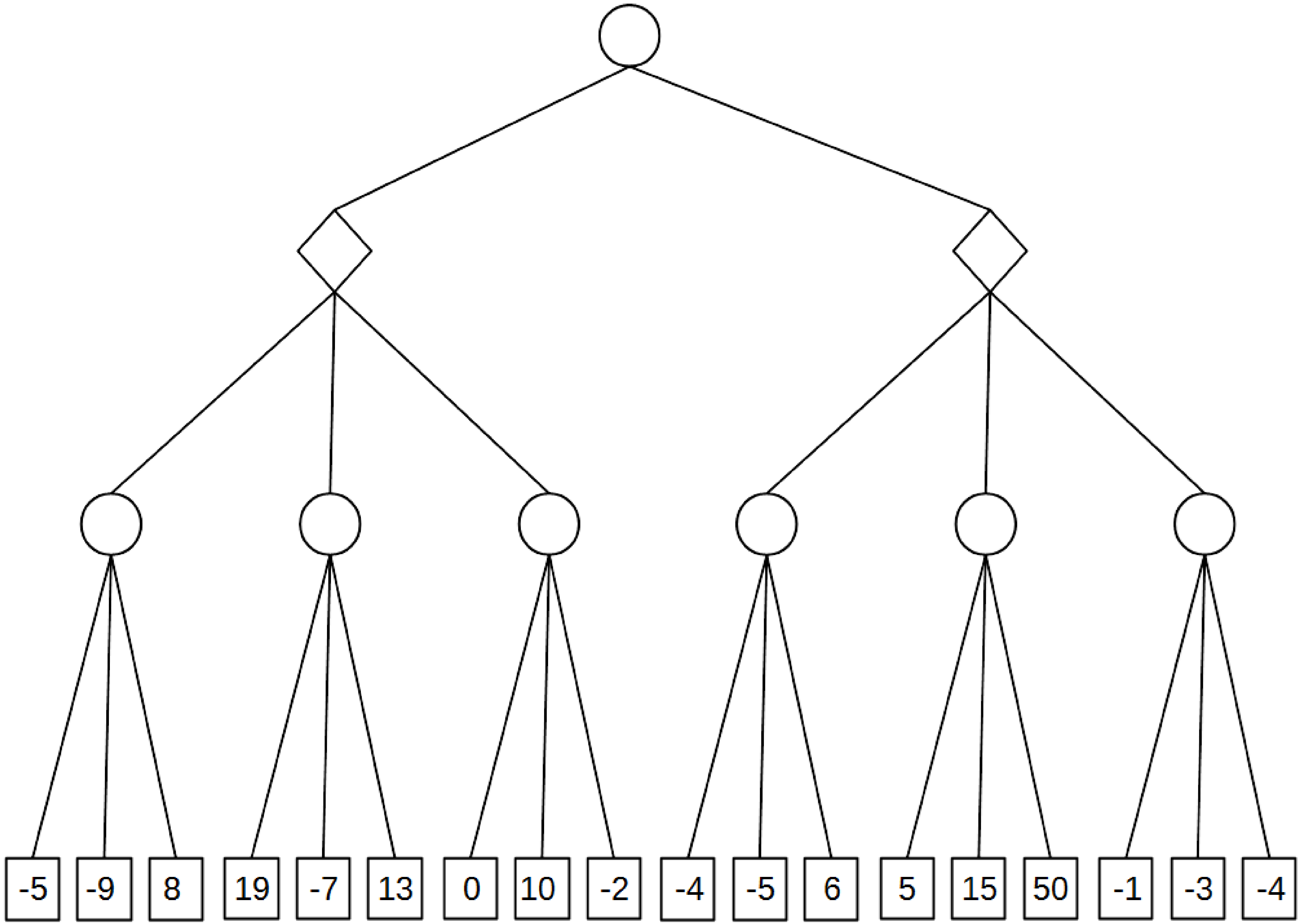
```
function AlphaBetaPruning (node h, real α, real β) returns u_1(h)
if h ∈ Z then
    return u_1(h)                                              // h is a terminal node
best_util ← (2ρ(h) − 3) × ∞                                    // −∞ for player 1; ∞ for player 2
forall a ∈ χ(h) do
    if ρ(h) = 1 then
        best_util ← max(best_util, AlphaBetaPruning(σ(h, a), α, β))
        if best_util ≥ β then
            return best_util
        α ← max(α, best_util)
    else
        best_util ← min(best_util, AlphaBetaPruning(σ(h, a), α, β))
        if best_util ≤ α then
            return best_util
        β ← min(β, best_util)
return best_util
```

```
function ALPHABETAPRUNING (node h, real α, real β) returns u_1(h)
if h ∈ Z then
    return u_1(h)                                   // h is a terminal node
best_util ← (2ρ(h) − 3) × ∞                         // −∞ for player 1; ∞ for player 2
forall a ∈ χ(h) do
    if ρ(h) = 1 then
        best_util ← max(best_util, ALPHABETAPRUNING(σ(h, a), α, β))
        if best_util ≥ β then
            return best_util
        α ← max(α, best_util)
    else
        best_util ← min(best_util, ALPHABETAPRUNING(σ(h, a), α, β))
        if best_util ≤ α then
            return best_util
        β ← min(β, best_util)
return best_util
```

**α** = best already explored option along path to the root for maximizer

**β** = best already explored option along path to the root for minimizer




-5
-9
8
19
-7
13
0
10
-2
-4
-5
6
5
15
50
-1
-3
-4

# Two-player, zero-sum games: minimax and alpha-beta pruning

- The effectiveness depends on the order in which nodes are considered
  - If player 1 (2) considers nodes in increasing (decreasing) order of their value, then no nodes will ever be pruned
  - In the best case (where nodes are ordered in decreasing (increasing) value for player 1 (2), it has complexity of $O(b^{m/2})$
  - If nodes are examined in random order, then the analysis becomes more complicated
    - when $b$ is fairly small, the complexity of alpha-beta pruning is $O(b^{3m/4})$, which is still an exponential improvement

# Two-player, zero-sum games: minimax and alpha-beta pruning

- In practice, performance is between the best case and the random case
- Offers substantial practical benefit in two-player, zero-sum games for which the game tree is represented implicitly

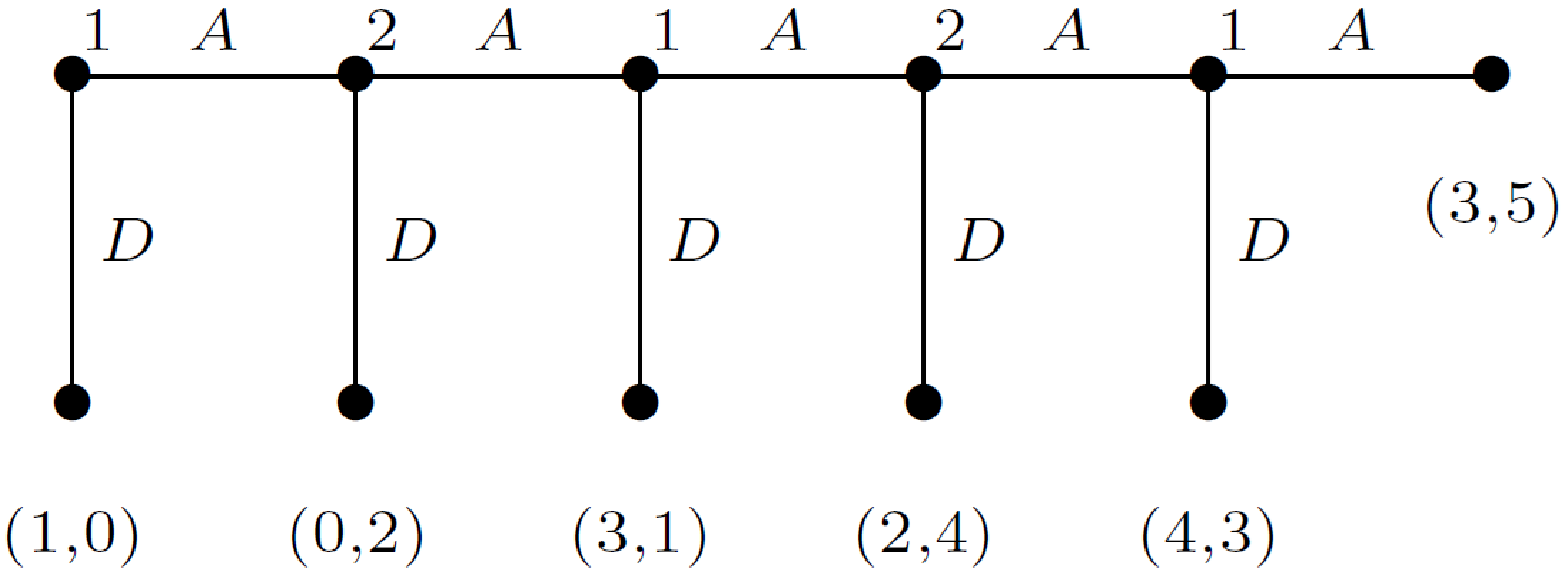
# Two-player, zero-sum games: minimax and alpha-beta pruning

- Commonly used to build strong computer players for two-player board games such as chess
  - The game tree in practical games can be so large that it is infeasible to search all the way down to leaf nodes
  - Instead, the search proceeds to some shallower depth
  - Where do we get the node values to propagate up using backward induction?

# Two-player, zero-sum games: minimax and alpha-beta pruning

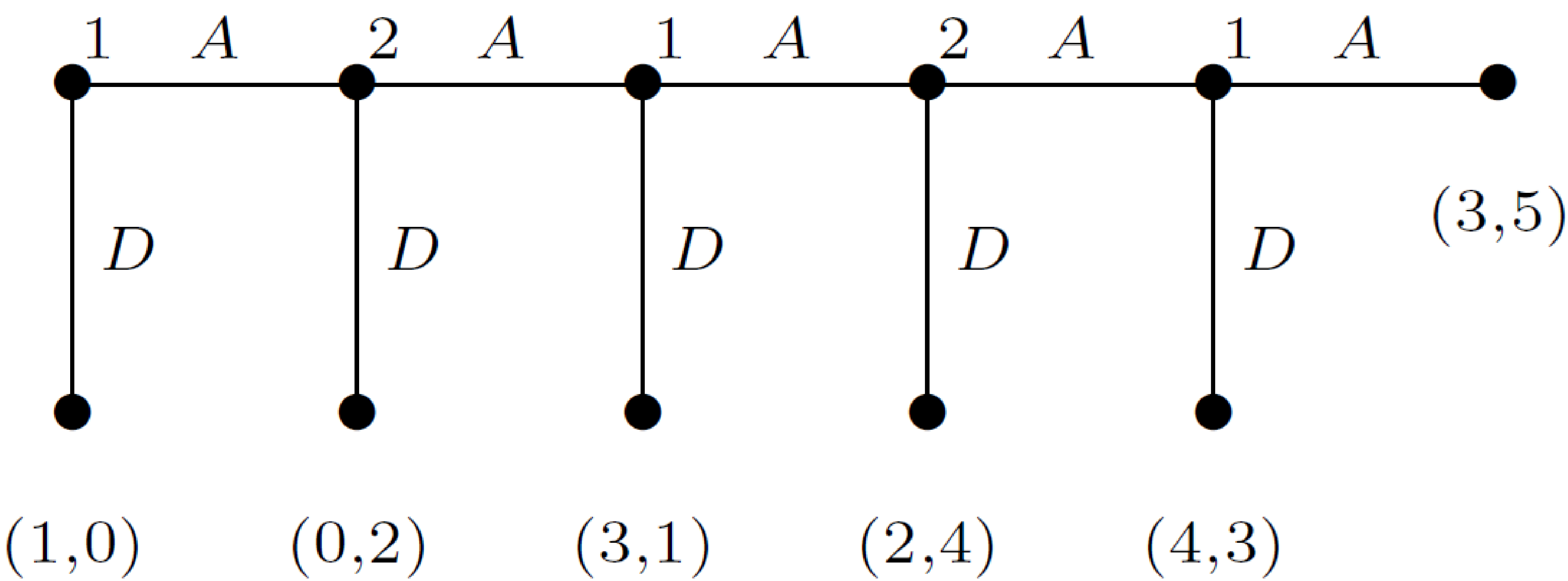
- The trick is to use an **evaluation function** to estimate the value of the deepest evaluation node reached
  - taking into function account game-relevant features such as board position, number of pieces for each player, who gets to move next, etc., and either built by hand or learned)

# Centipede game



# Centipede game

- What backward induction says about this game?



# Imperfect Information Games

- In many situations we may want to model agents
  - needing to act with partial or no knowledge of the actions taken by others
  - or even agents with limited memory of their own past actions

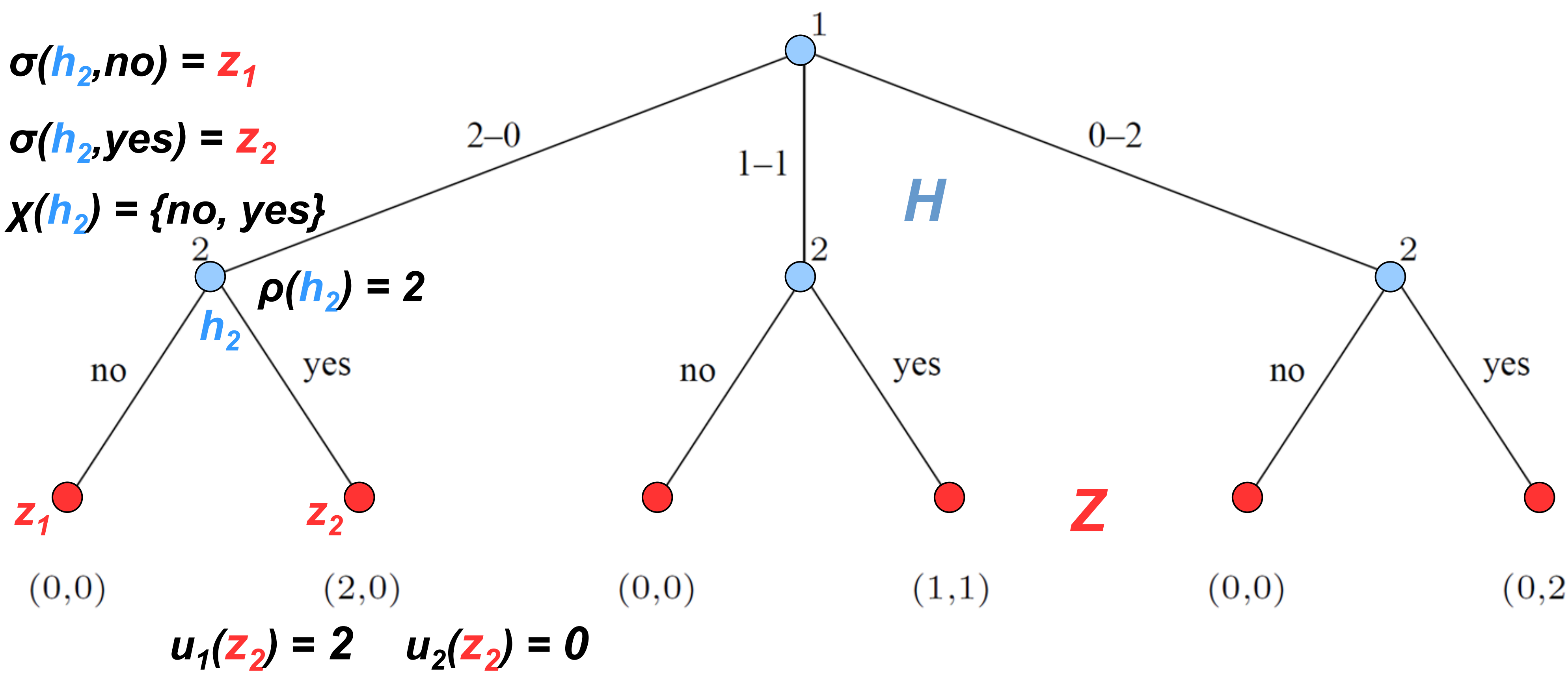
# Imperfect Information Games

- Each player's choice nodes are partitioned into **information sets** $I_i = \{I_{i,1}, \ldots, I_{i,k_i}\}$
- Intuitively, if two choice nodes are in the same information set then the agent cannot distinguish between them

I have to decide
without knowing
which
cards my
opponents have

# Perfect-information games in extesive form

- The sharing game
  - ***N* = {1,2}, $A_1$ = {2-0, 1-1, 0-2}, $A_1$ = {no, yes}**



# Imperfect Information Games

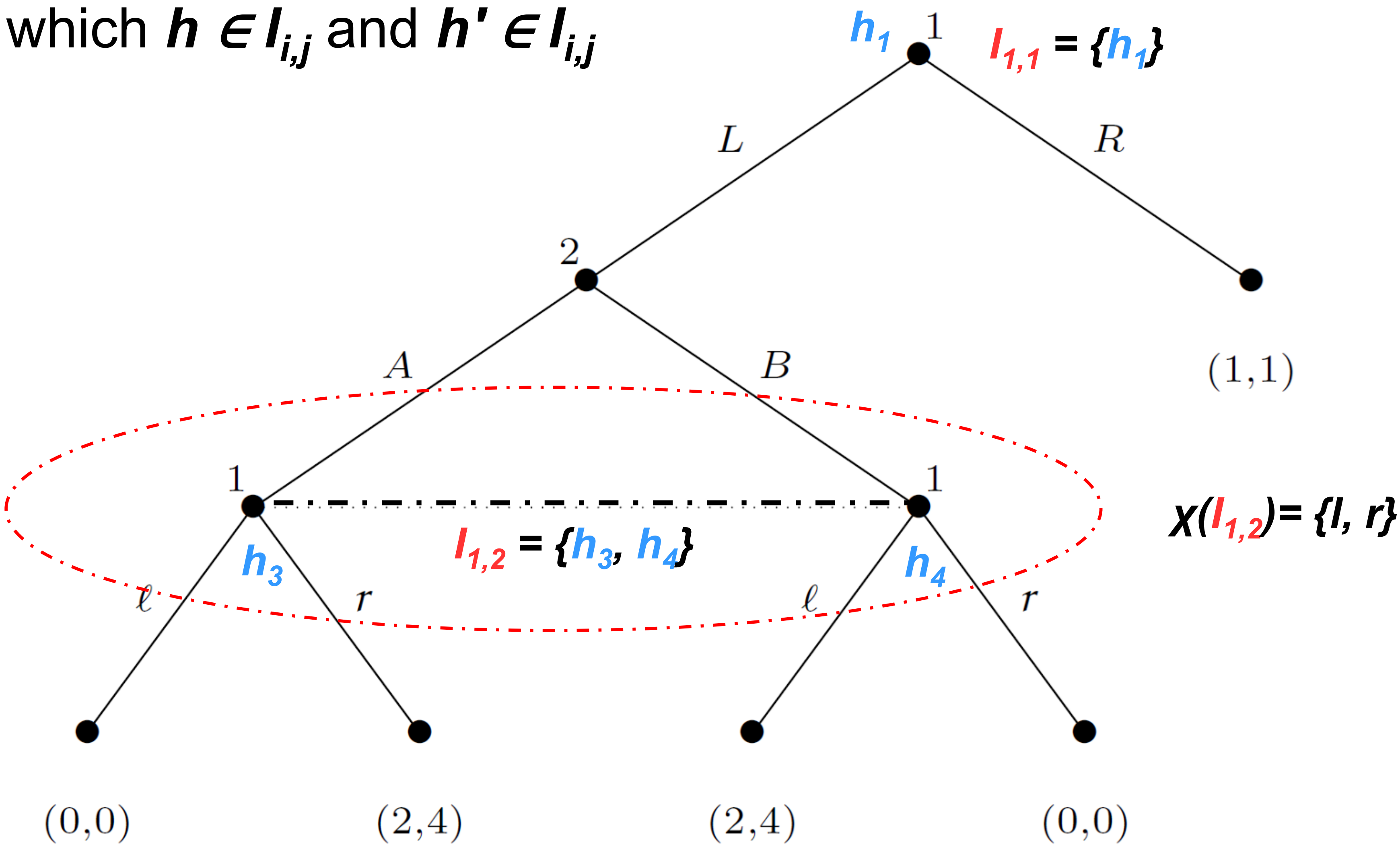
- An **imperfect-information game** (in extensive form) is a tuple $(N, A, H, Z, \chi, \rho, \sigma, u, I)$, where:
  - $(N, A, H, Z, \chi, \rho, \sigma, u)$ is a perfect-information extensive-form game; and
  - $I = (I_1, \ldots, I_n)$,

    where $I_i = (I_{i,1}, \ldots, I_{i,k_i})$ is a set of equivalence classes on (i.e., a partition of) $\{h \in H : \rho(h) = i\}$

    with the property that $\chi(h) = \chi(h')$ and $\rho(h) = \rho(h')$ whenever there exists a $j$ for which $h \in I_{i,j}$ and $h' \in I_{i,j}$

# Imperfect Information Games

- ***χ(h) = χ(h')*** and ***ρ(h) = ρ(h')*** whenever there exists a ***j*** for which $h \in I_{i,j}$ and $h' \in I_{i,j}$



# Strategies and Equilibria

- Let ***G* = (*N*, *A*, *H*, *Z*, *χ*, *ρ*, *σ*, *u*, *I*)** be an imperfect-information extensive-form game
- Then the **pure strategies** of player ***i*** consist of the Cartesian product

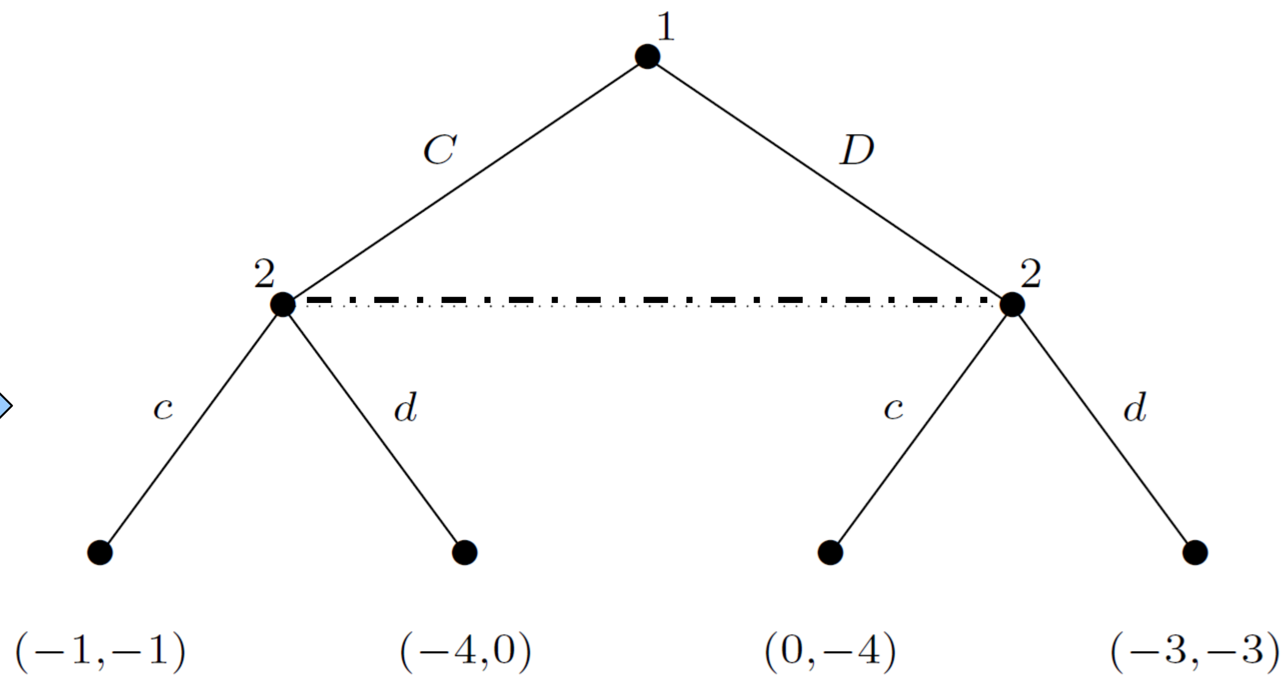
$$\prod_{I_{i,j} \in I_i} \chi(I_{i,j})$$

- Perfect-information games is a imperfect-information game that every equivalence class of each partition is a **singleton**

# Strategies and Equilibria

- Prisoner's Dilemma

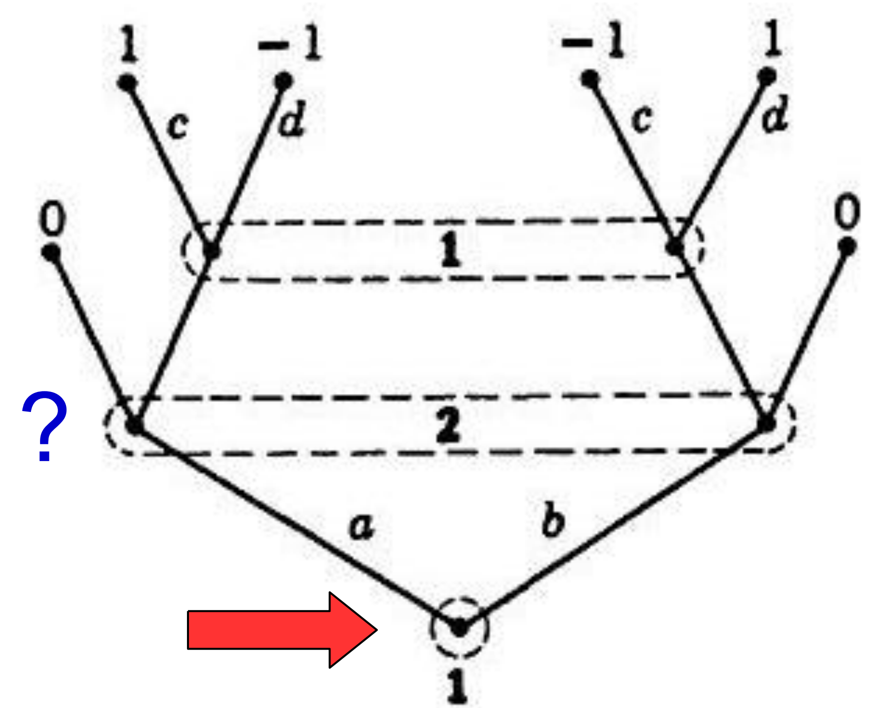
| | $C$ | $D$ |
|---|---|---|
| $C$ | $-1, -1$ | $-4, 0$ |
| $D$ | $0, -4$ | $-3, -3$ |




Any normal-form game can be trivially transformed into an equivalent imperfect-information game

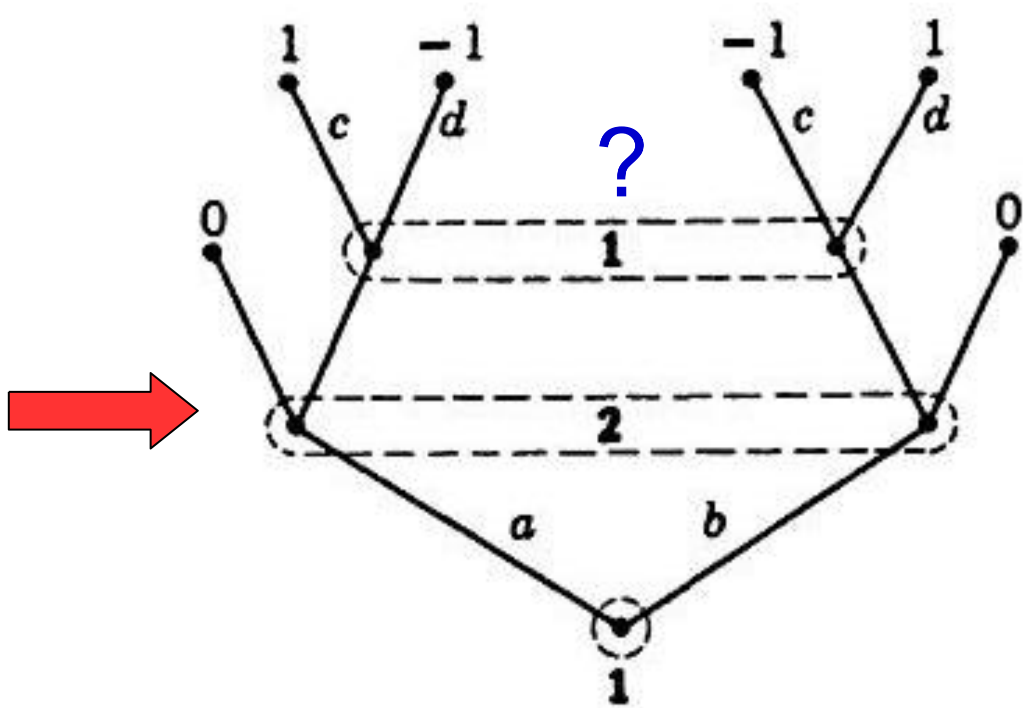
# Strategies and Equilibria

- Let's think about this game...
  - What player ***2*** knows about player ***1*** action at the root node?



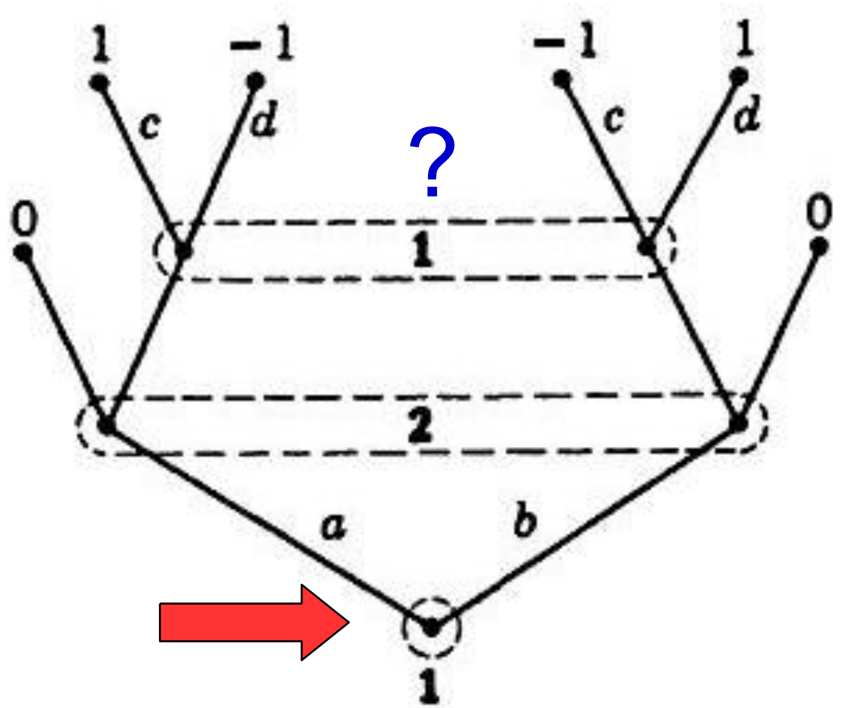
# Strategies and Equilibria

- Let's think about this game...
  - What player ***1*** knows about player ***2*** action?



# Strategies and Equilibria

- Let's think about this game...
  - What player ***1*** knows about his previous action?



# Behavioral Strategies

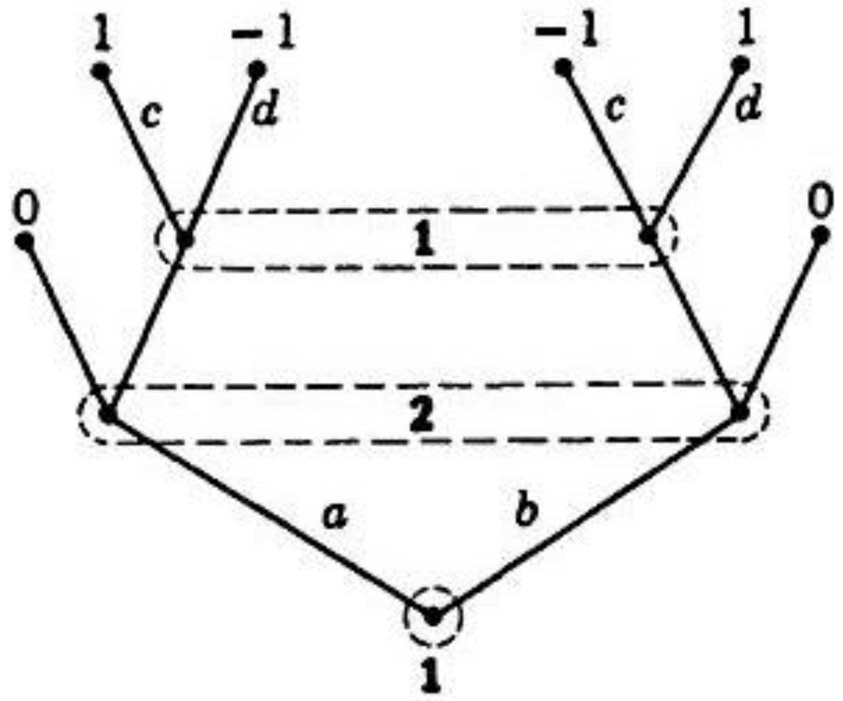
- Independent coin toss every time an information set is encountered
  - A vector of probability distributions over the information sets
  - in some games there are outcomes that are achieved via <u>mixed strategies</u> but not by any <u>behavioral strategies</u>, and vice-versa

# Behavioral Strategies

- A **behavioral strategy** of player $\boldsymbol{i \in N}$ is a map $\boldsymbol{b_i}$ assigning to each information set $\boldsymbol{I_{i,j} \in I_i}$ a probability distribution over the set of actions $\boldsymbol{\chi(I_{i,j})}$

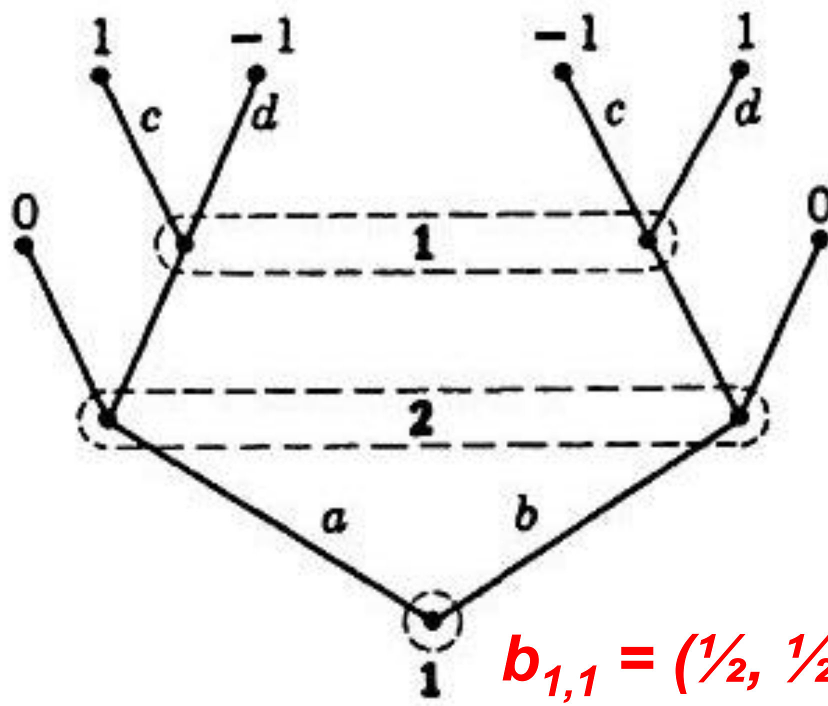
# Strategies and Equilibria

- Which mixed strategy profile gives the best payoff for player ***1*** for a given $s_2$= "*always continue*"
  - $s_1'$: ¼ (a,c), ¼ (a,d), ¼ (b,c), ¼ (b,d)
  - $s_1''$: ½ (a,c), 0 (a,d), 0 (b,c), ½ (b,d)



# Strategies and Equilibria

- What is the equivalent behavioral strategy for $s_1'$?
  - $s_1'$: ¼ (a,c), ¼ (a,d), ¼ (b,c), ¼ (b,d)

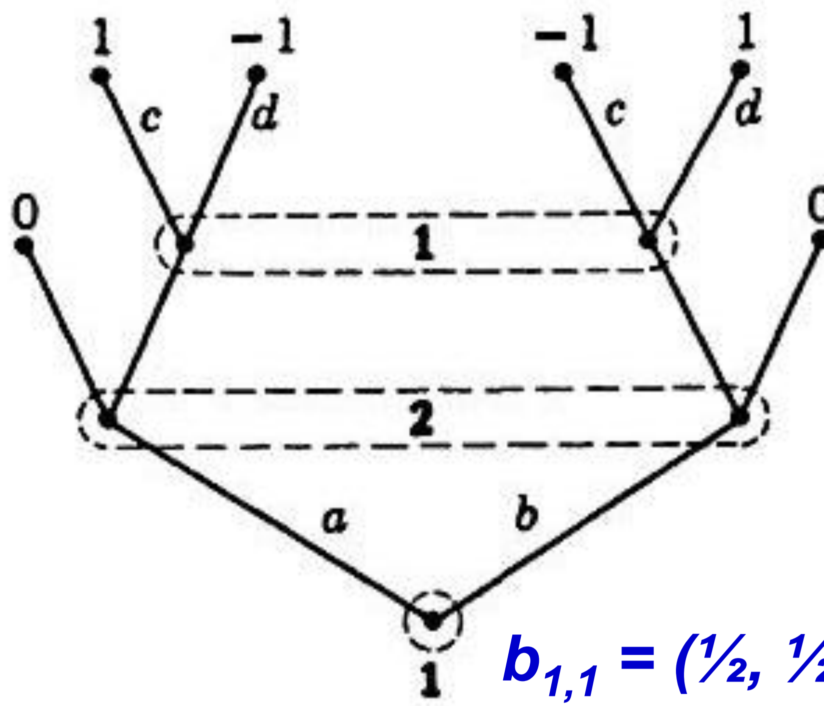



$b_{1,2} = (½, ½)$

$b_{1,1} = (½, ½)$

# Strategies and Equilibria

- What is the equivalent behavioral strategy for $s_1''$ ?
  - $s_1''$: ½ (a,c), 0 (a,d), 0 (b,c), ½ (b,d)

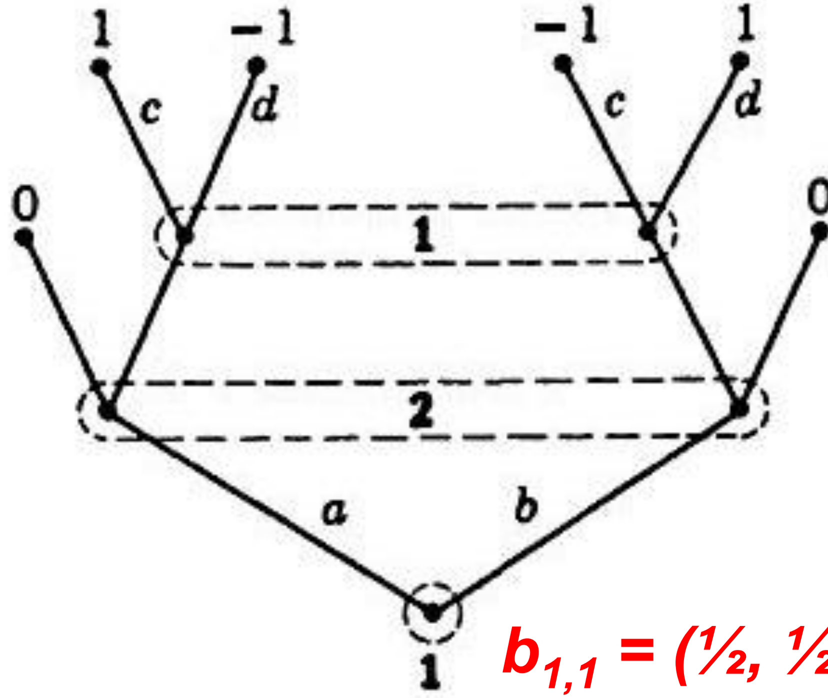



$b_{1,2} = (½, ½)$

$b_{1,1} = (½, ½)$

# Strategies and Equilibria

- What are the expected payoffs of these strategies?
  - $s_1'$: ¼ (a,c), ¼ (a,d), ¼ (b,c), ¼ (b,d)
  - $u_1(b) = (½ * ½ * 1) + (½ * ½ * -1) + (½ * ½ * -1) + (½ * ½ * 1) = 0$
  - $u_1(s_1', s_2) = (¼ * 1) + (¼ * -1) + (¼ * -1) + (¼ * 1) = 0$

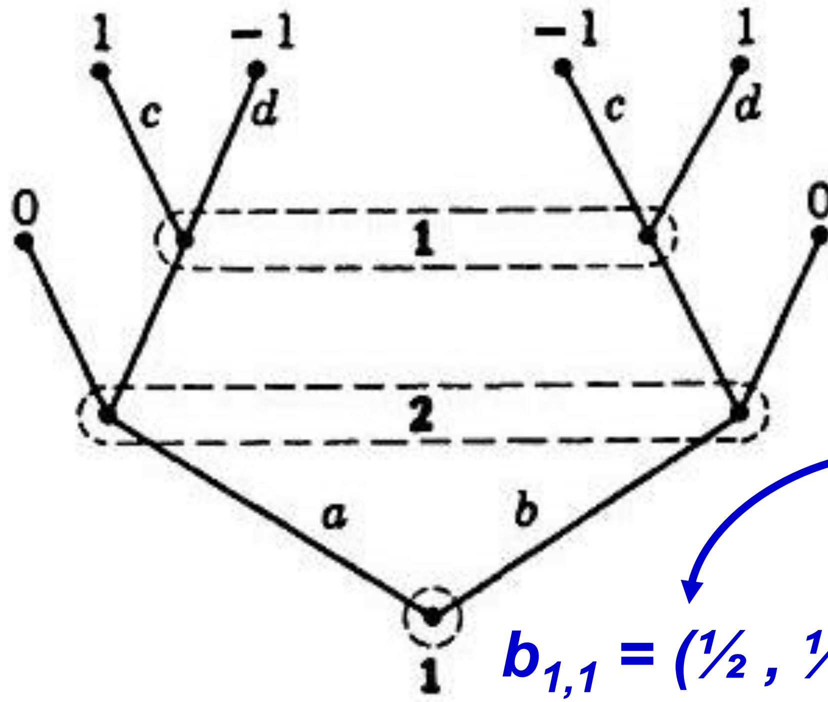



$b_{1,2} = (½, ½)$

$b_{1,1} = (½, ½)$

# Strategies and Equilibria

- What are the expected payoffs of these strategies?
  - $s_1''$: $\frac{1}{2}$ *(a,c)*, $0$ *(a,d)*, $0$ *(b,c)*, $\frac{1}{2}$ *(b,d)*
  - $u_1(b) = (\frac{1}{2} * \frac{1}{2} * 1) + (\frac{1}{2} * \frac{1}{2} * -1) + (\frac{1}{2} * \frac{1}{2} * -1) + (\frac{1}{2} * \frac{1}{2} * 1) = 0$
  - $u_1(s_1'', s_2) = (\frac{1}{2} * 1) + (\frac{1}{2} * 1) = 1$




$b_{1,2} = (\frac{1}{2}, \frac{1}{2})$

**can't be correlated!**

$b_{1,1} = (\frac{1}{2}, \frac{1}{2})$

# Perfect Recall

- Games in which no player forgets any information they knew about moves made so far
  - in particular, they remember precisely all their own moves

# Perfect Recall #1

- Player $i$ has **perfect recall** in an imperfect-information game $G$ if for any two nodes $h$, $h'$ that are in the same information set for player $i$,
- for any path $h_0, a_0, h_1, a_1, \ldots, h_m, a_m, h$ from the root of the game to $h$ and for any path $h_0', a_0', h_1', a_1', \ldots, h_{m'}', a_{m'}', h'$ from the root to $h'$ it must be the case that:
  - $m = m'$
  - for all $0 \le j \le m$, if $\rho(h_j) = i$, then $h_j$ and $h_j'$ are in the same equivalence class for $i$
  - for all $0 \le j \le m$, if $\rho(h_j) = i$, then $a_j = a_j'$
- $G$ is a game of perfect recall if every player has perfect recall in it

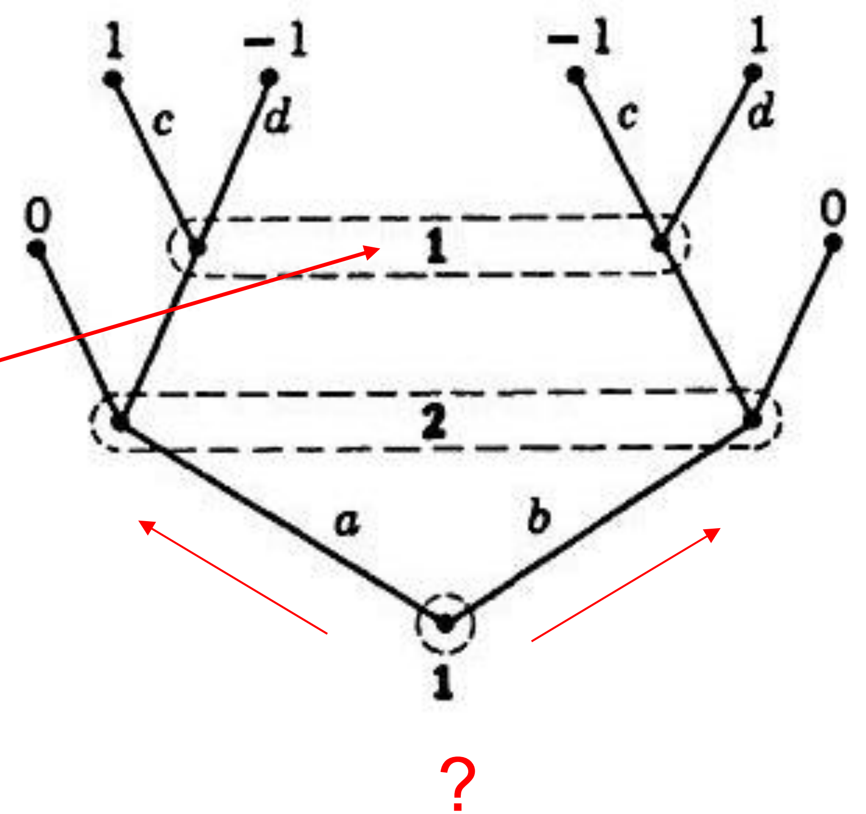
# Perfect Recall #2

- An information set $I_{i,j}$ with a given branch (action) $r$ is a **signaling information set** if there is a later information set $I_{i,k}$ such that:
  - There is at least one node of $I_{i,k}$ that **can** be reached by a path starting with $r$
  - There is at least one node of $I_{i,k}$ that **cannot** be reached by any path starting with $r$
- A game is said to have **perfect recall** if there are no signaling information sets

# Strategies and Equilibria

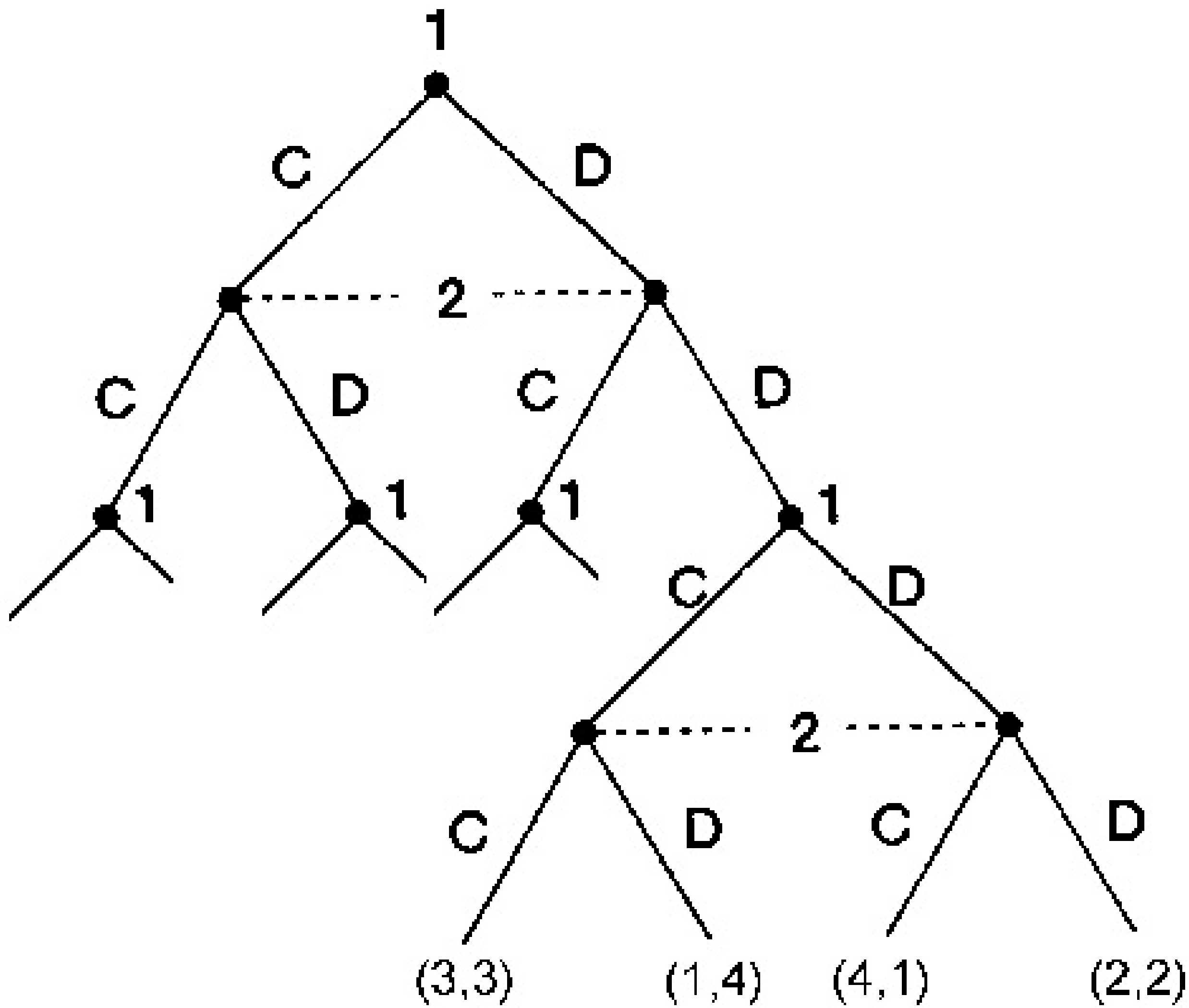
- Does not have **perfect recall**

***1* cannot remember from where he came from**



# Strategies and Equilibria

- Does have **perfect recall**



# Perfect Recall

- **Theorem 5.2.4 (Kuhn, 1953)**
  - In a game of **perfect recall**, any mixed strategy of a given agent can be replaced by an equivalent behavioral strategy, and vice-versa
  - These two strategies yield the same probabilities on the set of outcomes

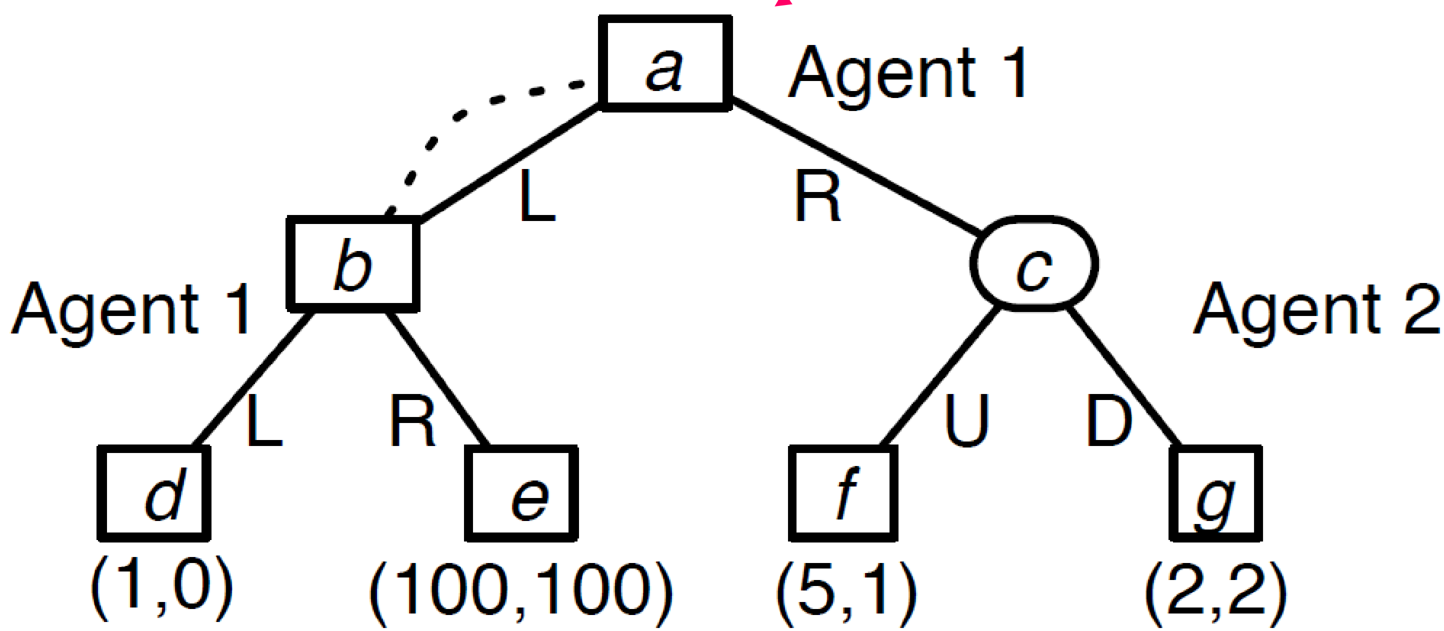
# Perfect Recall

- **Theorem 5.2.4 (Kuhn, 1953)**
  - Q: What does it mean?
  - A: This result means that for such games it does not matter to the players whether they take the global view of mixed strategies or the more restricted (and plausible) view of behavioral strategies

# Strategies and Equilibria

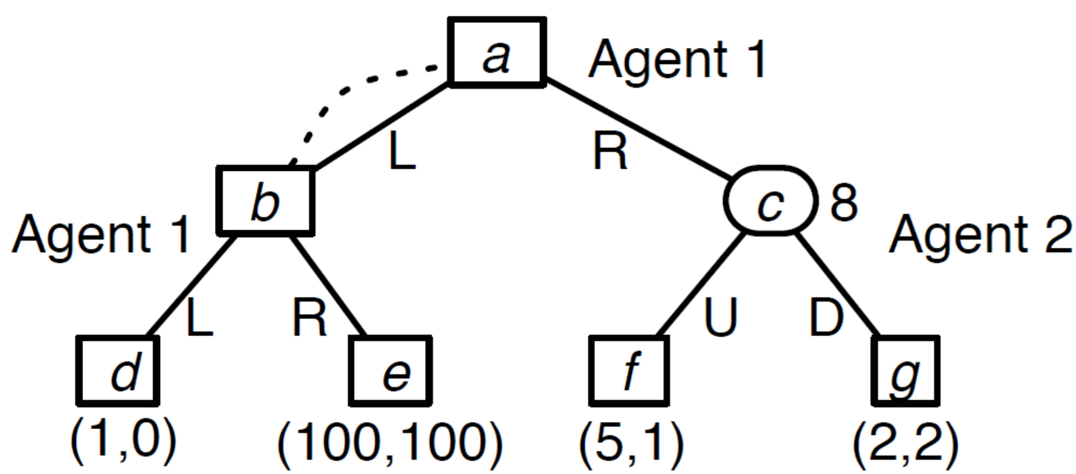
- Q: What is the equilibrium using mixed strategies?
  - ***D*** is a strictly best response for ***2***
  - ***R*** is a strictly best response for ***1***
- A: ***(R,D)***

**1 arrives here, but does not know that!**



# Behavioral Strategies

- What is an equilibrium in behavioral strategies?
  - Again, ***D*** strongly dominant for ***2***
  - If ***1*** uses the behavioural strategy ***(p; 1 - p)***, his expected utility is
  - $1 * p^2 + 100 * p(1 - p) + 2 * (1 - p)$
  - It simplifies to $-99p^2 + 98p + 2$
  - Its maximum is at $p = 98/198$
  - Thus equilibrium is ***(98/198; 100/198); (0, 1)***



# Computing equilibria: the sequence form

- An obvious way to find an equilibrium of an extensive-form game is to first convert it into a normal-form game, and then find the equilibria using, for example, the Lemke–Howson algorithm
- This method is inefficient, since the number of actions in the normal-form game is exponential in the size of the extensive-form game
  - The normal-form game is created by considering all combinations of information set actions

# Computing equilibria: the sequence form

- One way to avoid this problem is to operate directly on the extensive-form representation by employing behavioral strategies to express a game
  - using a description called the **sequence form**

# Computing equilibria: the sequence form

- Let $\boldsymbol{G}$ be an imperfect-information game of perfect recall
- The sequence-form representation of $\boldsymbol{G}$ is a tuple $(\boldsymbol{N}, \boldsymbol{\Sigma}, \boldsymbol{g}, \boldsymbol{C})$, where
  - $\boldsymbol{N}$ is a set of agents
  - $\boldsymbol{\Sigma} = (\Sigma_1, \Sigma_2, \ldots, \Sigma_n)$, where $\Sigma_i$ is the set of sequences available to agent $i$
  - $\boldsymbol{g} = (g_1, \ldots, g_n)$, where $g_i : \Sigma \rightarrow \mathbb{R}$ is the payoff function for agent $i$
  - $\boldsymbol{C} = (C_1, \ldots, C_n)$, where $C_i$ is a set of linear constraints on the realization probabilities of agent $i$

# Computing equilibria: the sequence form

- What is a sequence $\boldsymbol{\sigma} \in \boldsymbol{\Sigma}_i$?
  - **Insight**: while there are exponentially many pure strategies in an extensive-form game, there are only a small number of nodes in the game tree
    - Does not build a player's strategy around the idea of pure strategies
    - The sequence form builds it around paths in the tree from the root to each node

# Computing equilibria: the sequence form

- A sequence of actions of player $\boldsymbol{i} \in \boldsymbol{N}$, defined by a node $\boldsymbol{h} \in \boldsymbol{H} \cup \boldsymbol{Z}$ of the game tree, is the ordered set of player i's actions that lie on the path from the root to $\boldsymbol{h}$
- Let $\emptyset$ denote the sequence corresponding to the root node
- The set of sequences of player $\boldsymbol{i}$ is denoted $\boldsymbol{\Sigma_i}$
- $\boldsymbol{\Sigma_1} \times \cdot \cdot \cdot \times \boldsymbol{\Sigma_n}$ is the set of all sequences

# Computing equilibria: the sequence form

- A sequence can thus be thought of as a **string** listing the action choices that player ***i*** would have to take in order to get from the root to a given node ***h***
- ***h*** may or may not be a leaf node
- The other players' actions that form part of this path are not part of the sequence

# Computing equilibria: the sequence form

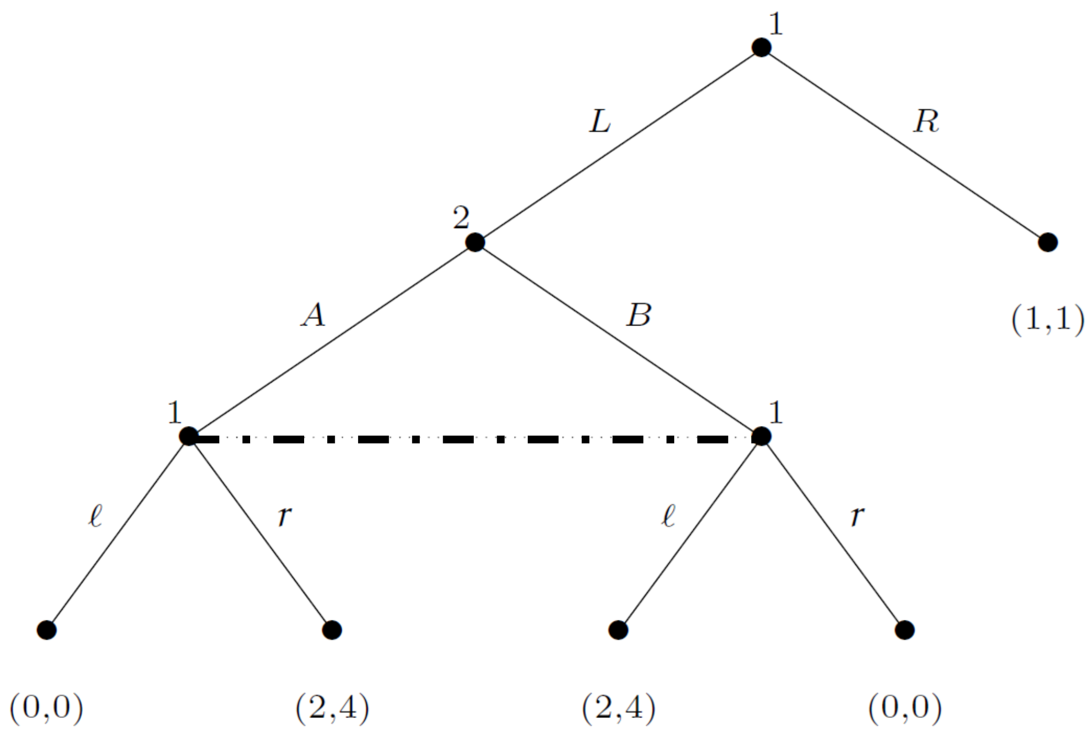
- **Definition**
  - The **payoff function** $g_i : \Sigma \to \mathbb{R}$ for agent $i$ is given by
    - $g_i(\sigma) = u_i(z)$ if a leaf node $z \in Z$ would be reached when each player played his sequence $\sigma_i \in \sigma$
    - $g_i(\sigma) = 0$ otherwise

# Computing equilibria: the sequence form

- Given the set of sequences $\boldsymbol{\Sigma}$ and the payoff function $\boldsymbol{g}$, we can think of the sequence form as defining a tabular representation of an imperfect-information extensive-form game, much as the induced normal form does.

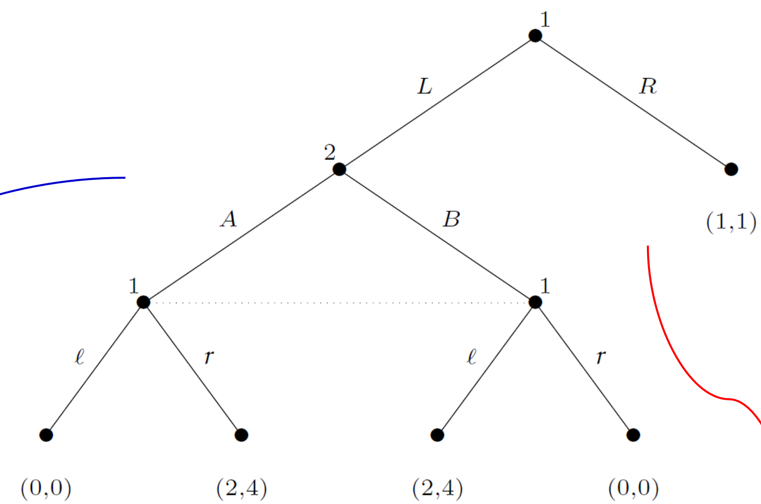
# Computing equilibria: the sequence form

- The sets of sequences for the two players are
  - $\Sigma_1 = \{\emptyset, L, R, L\ell, Lr\}$
  - $\Sigma_2 = \{\emptyset, A, B\}$



# Computing equilibria: the sequence form

Why the sequence form can be better?




| | $\emptyset$ | $A$ | $B$ |
|---|---|---|---|
| $\emptyset$ | $0, 0$ | $0, 0$ | $0, 0$ |
| $L$ | $0, 0$ | $0, 0$ | $0, 0$ |
| $R$ | $1, 1$ | $0, 0$ | $0, 0$ |
| $L\ell$ | $0, 0$ | $0, 0$ | $2, 4$ |
| $Lr$ | $0, 0$ | $2, 4$ | $0, 0$ |

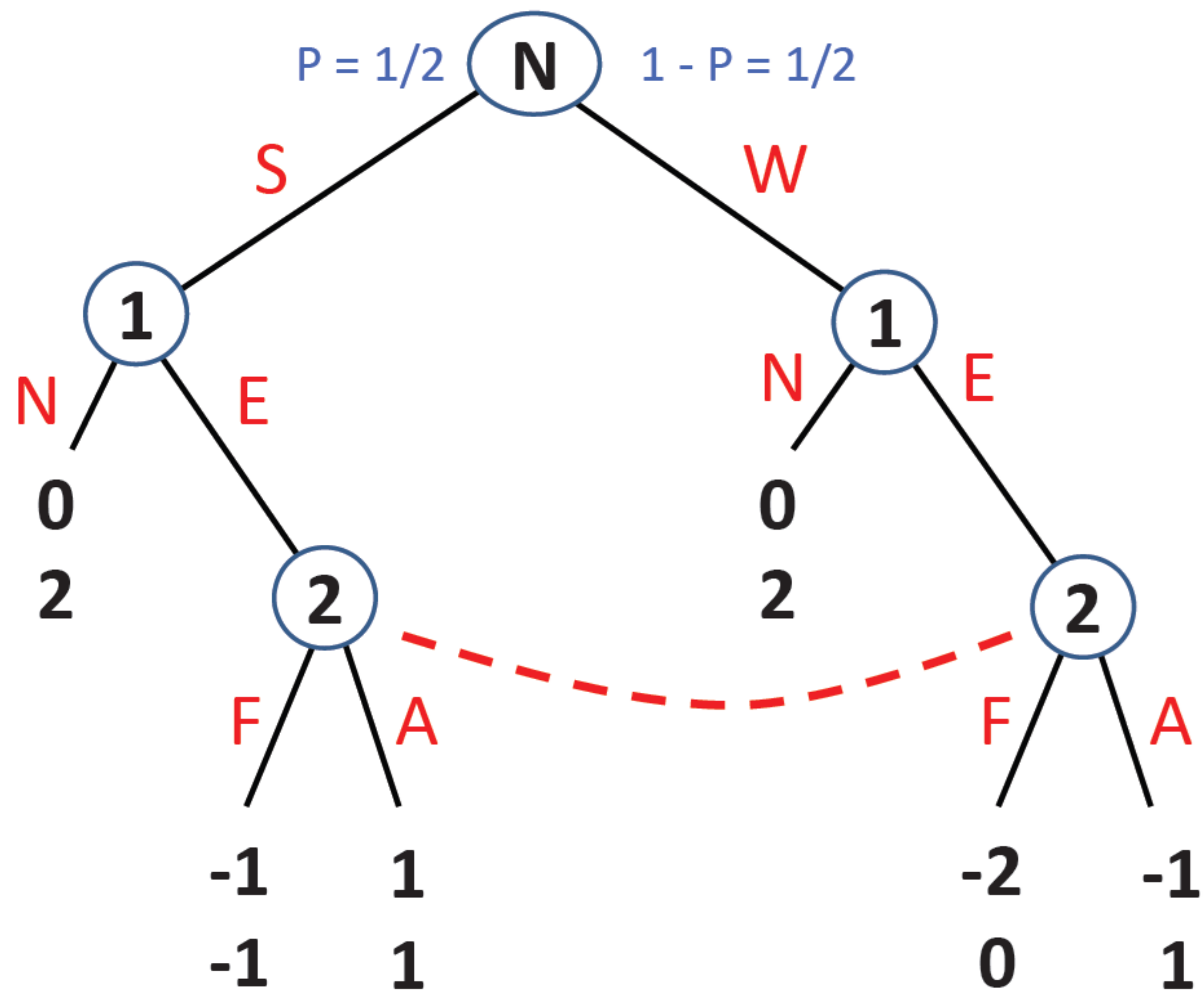
| | $A$ | $B$ |
|---|---|---|
| $L\ell$ | $0, 0$ | $2, 4$ |
| $Lr$ | $2, 4$ | $0, 0$ |
| $R\ell$ | $1, 1$ | $1, 1$ |
| $Rr$ | $1, 1$ | $1, 1$ |

# Computing equilibria: the sequence form

- Each payoff that is defined at a leaf in the game tree occurs **exactly once** in the sequence-form table
- If ***g*** was represented using a sparse encoding, only five values would have to be stored
- In induced normal form, all of the eight entries correspond to leaf nodes from the game tree

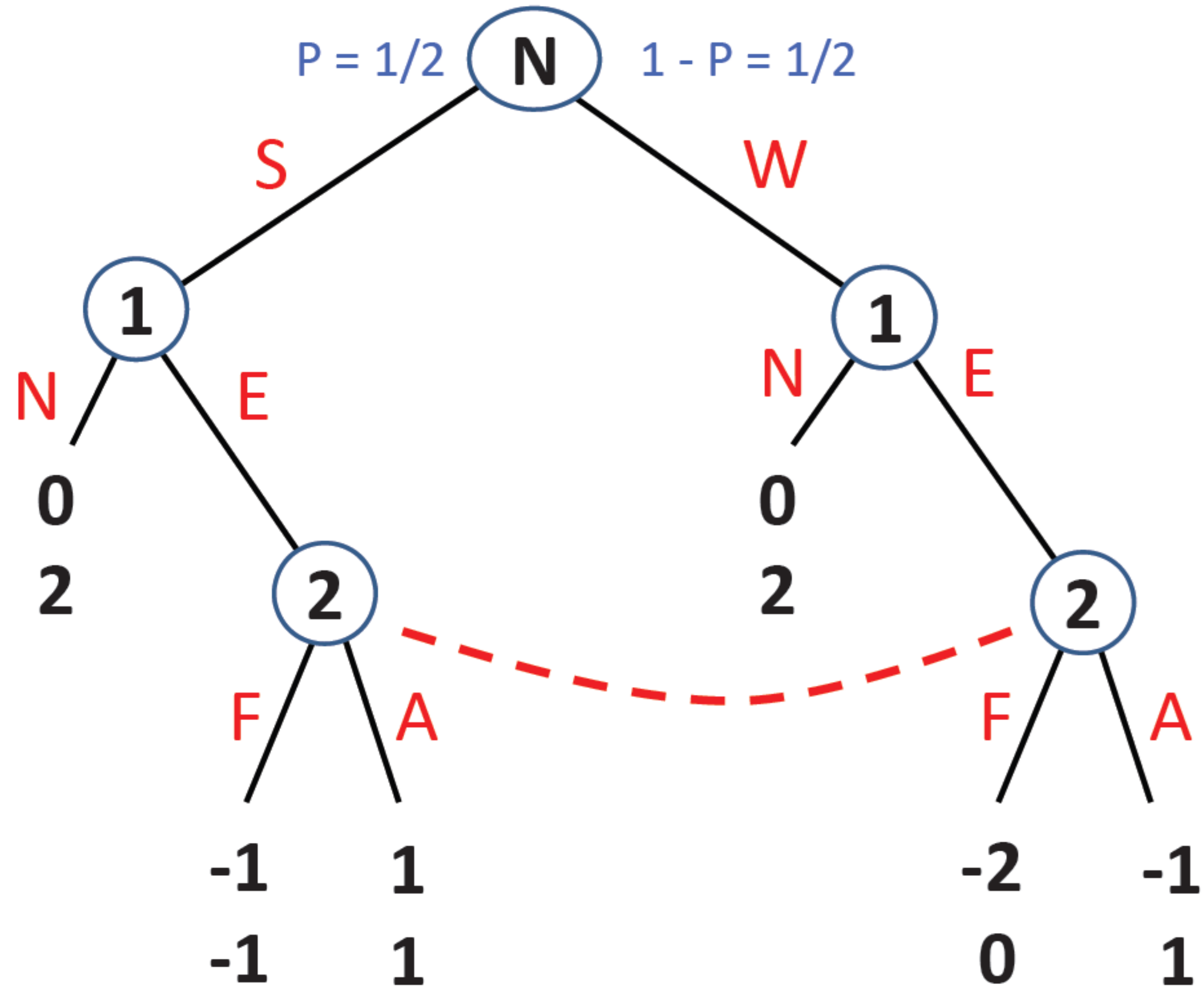
# Sequential Equilibrium

- Should firm ***1*** enter the market?



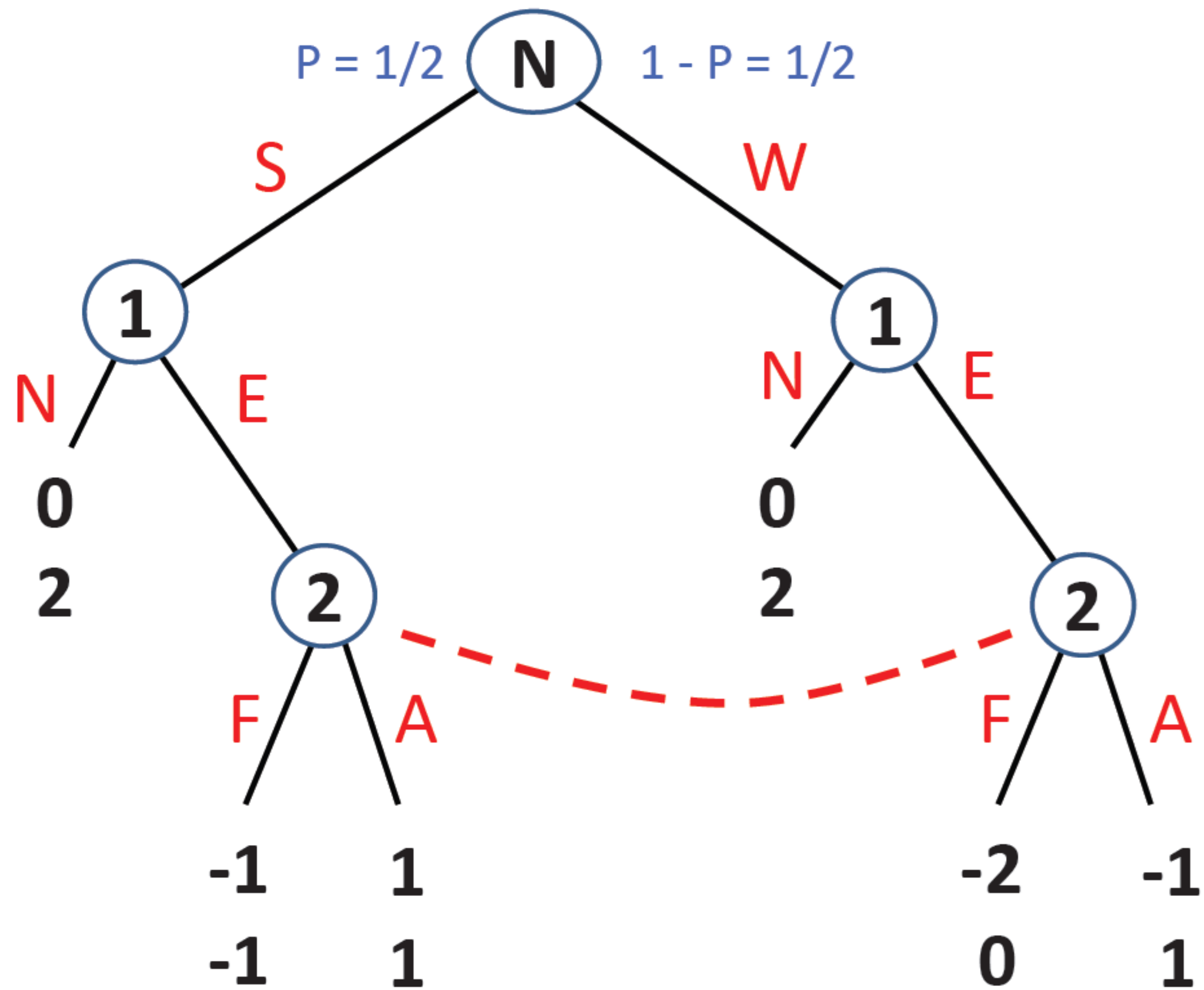
# Sequential Equilibrium

- What are the subgames of this game?



# Sequential Equilibrium

- What are the NE of this game?
  - 1) if 1==S or 1==W
    - ***(N,F)***
  - 2) if 1 == S
    - ***(E,A)***
  - if 1 == W
    - ***(N,A)***

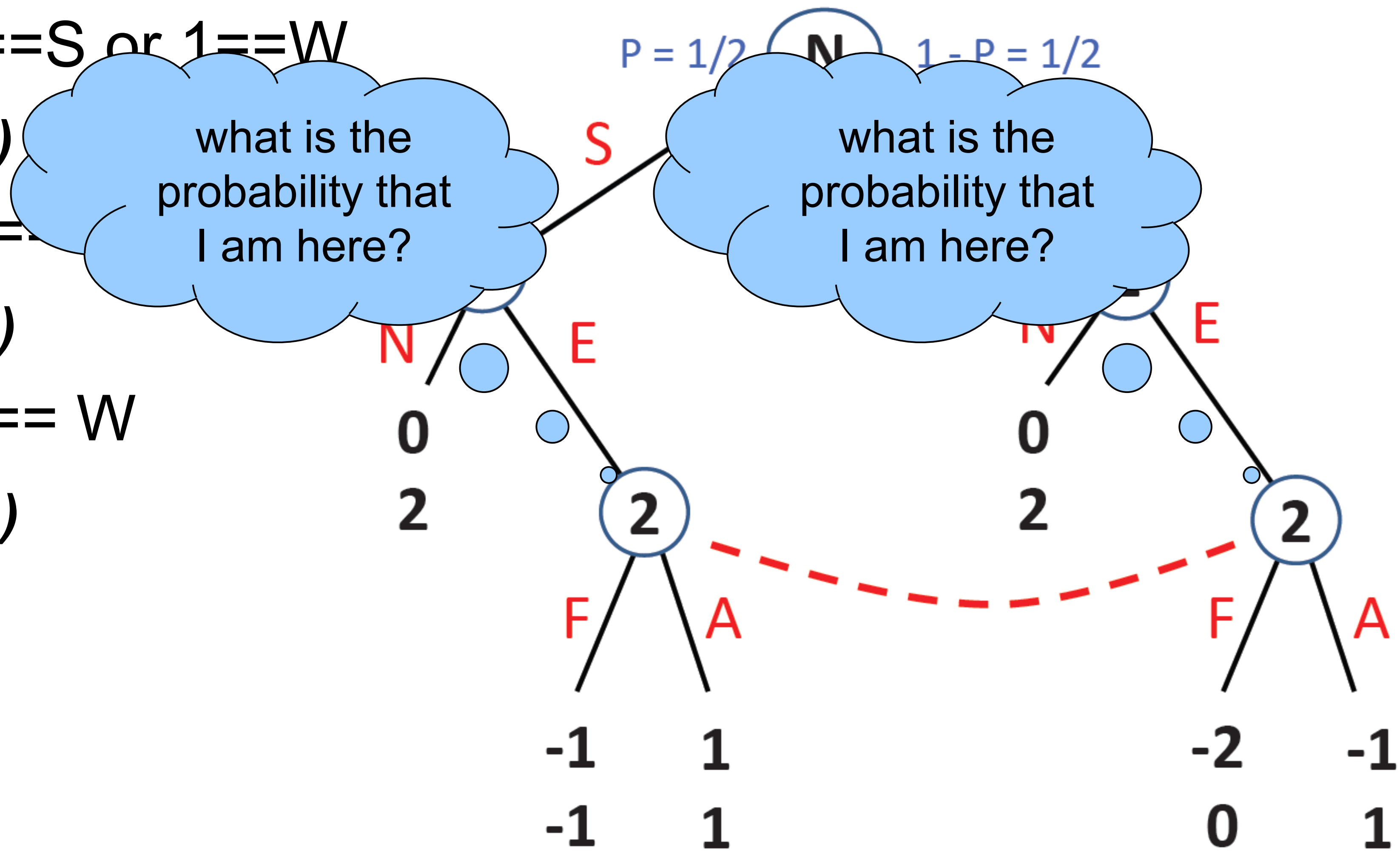


# Sequential Equilibrium

- Similar idea with **SPE**
- Equilibrium concept that explicitly model players' beliefs about
  - **where they are** in the tree for every information set (what the other players have done)
- Features:
  - **Beliefs** are not contradicted by the actual play of the game (on the equilibrium path)
  - Players **best respond** to their beliefs

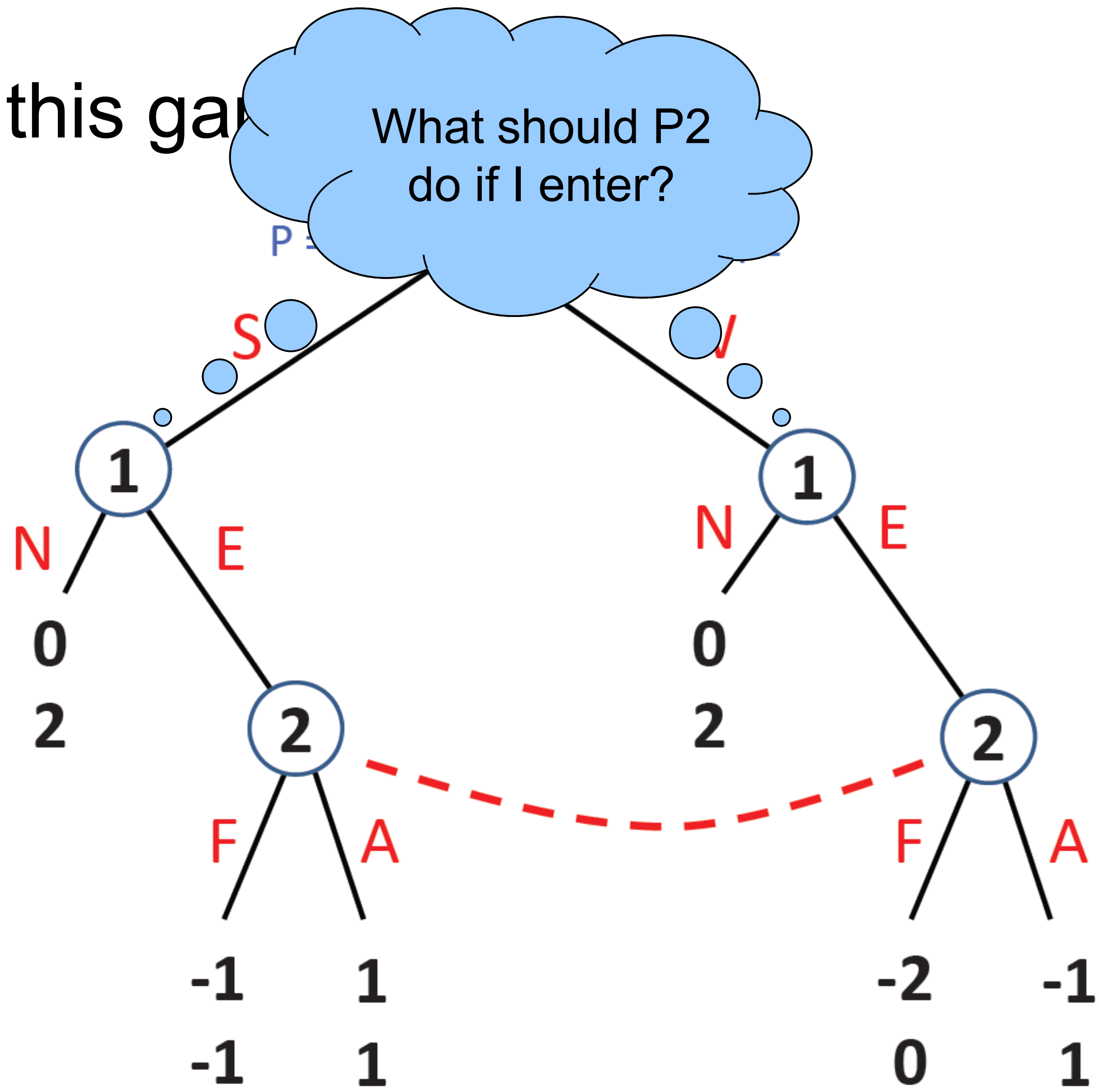
# Sequential Equilibrium

- What are the NE of this game?
  - 1) if 1==S or 1==W
    - *(N,F)*
  - 2) if 1 ==
    - *(E,A)*
  - if 1 == W
    - *(N,A)*



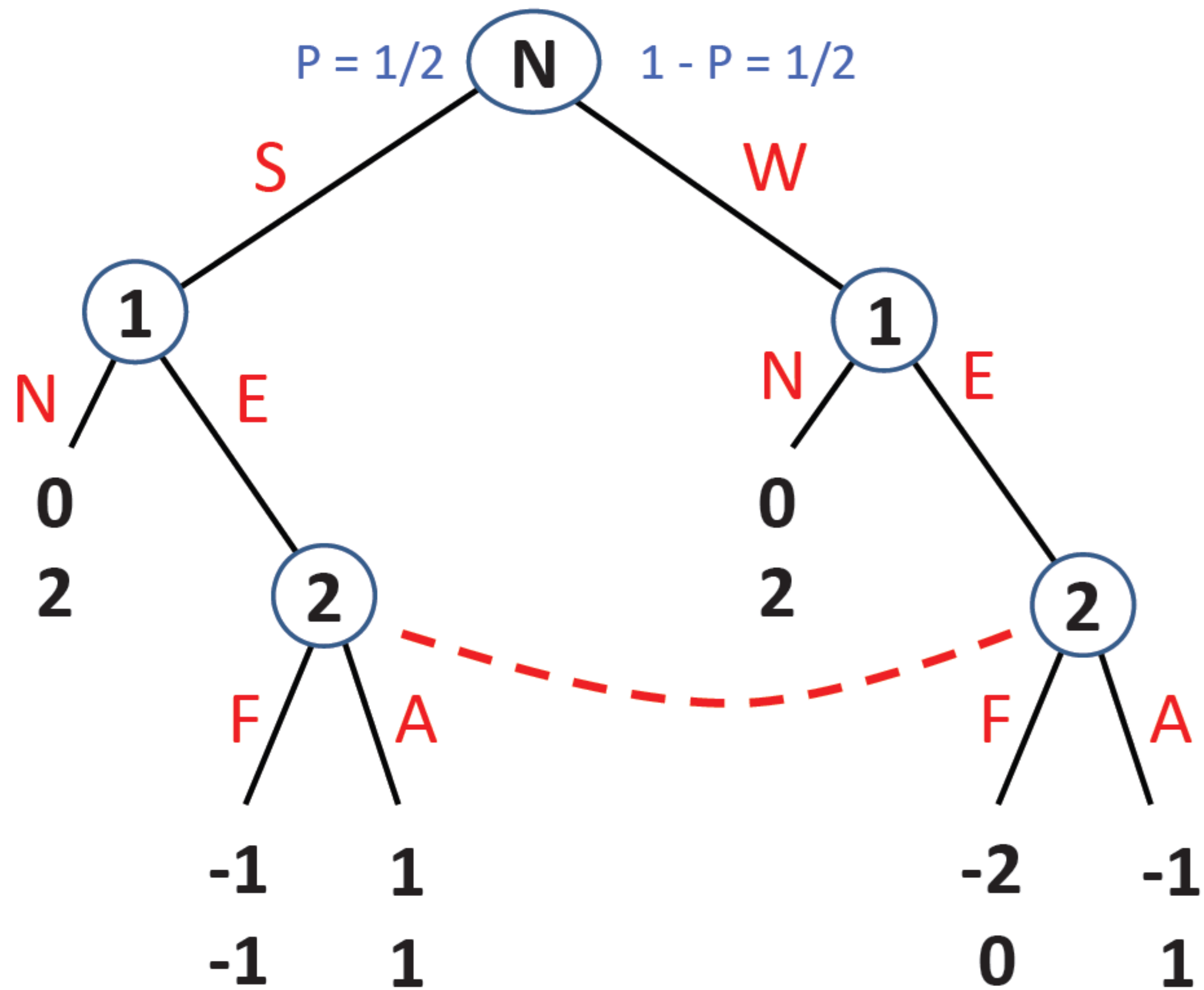
# Sequential Equilibrium

- What are the NE of this ga
  - 1) if 1==S or 1==W
    - ***(N,F)***
  - 2) if 1 == S
    - ***(E,A)***
  - if 1 == W
    - ***(N,A)***



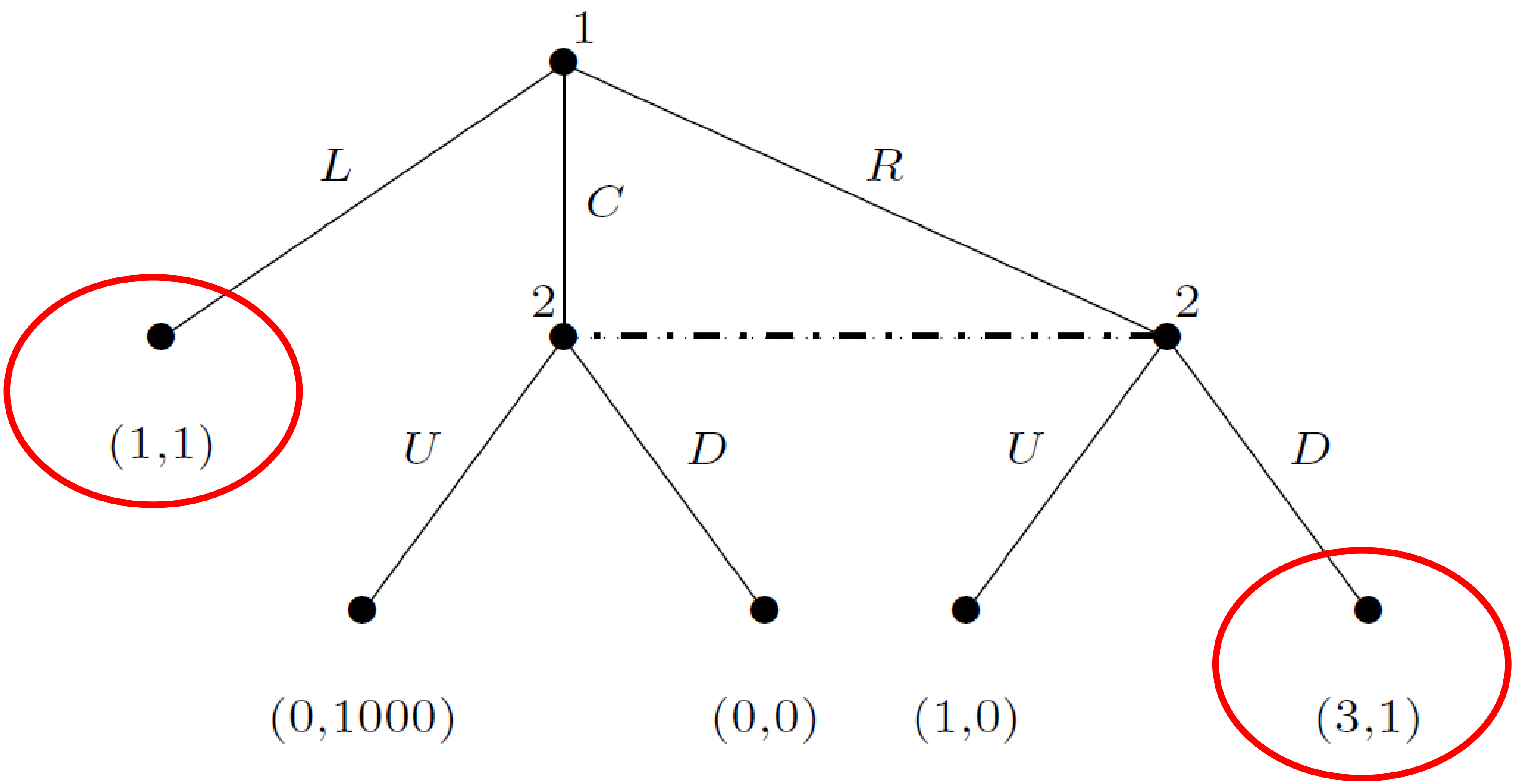
# Sequential Equilibrium

- What are the NE of this game?
  - 1) if 1==S or 1==W
    - *(N,F)*
  - 2) if 1 == S
    - ***(E,A)***
  - if 1 == W
    - ***(N,A)***



# Sequential Equilibrium

- Does player 2 know where he is?



# Sequential Equilibrium

- The best AI programs are starting to approach the level of human experts
  - Construct a statistical model of the opponent
    - What kinds of bets the opponent is likely to make under what kinds of circumstances
  - Combine with game-theoretic reasoning techniques, e.g.,

# Sequential Equilibrium

- Rather, at each info set, a “subforest” or a collection of subgames
- The best-known way for dealing with this is sequential equilibrium (SE)
  - More info in the text book
- Theorem: every finite game of perfect recall has a sequential equilibrium
- Theorem: in extensive-form games of perfect information, the sets of SPE and SE are always equivalent

# Sequential Equilibrium

- Works for games with perfect recall
  - for general-sum games, can compute equilibrium in time exponential in the size of the extensive form game
    - exponentially faster than converting to normal form
  - for zero-sum games, computing equilibrium is polynomial in the size of the extensive form game
    - exponentially faster than the LP formulation we saw before
- Solutions via linear programs