α-β prunnig – How much can we save?

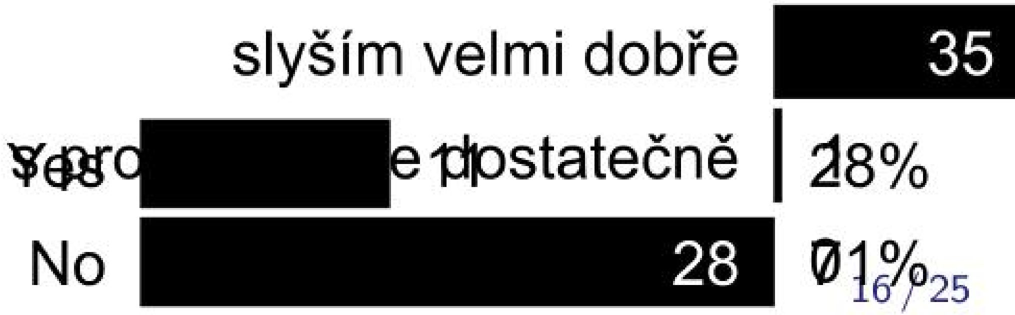
original: Time: $O(b^m)$

- how to consider next actions/moves (in what order)?
- perfect ordering?

α-β prunnig – How much can we save?

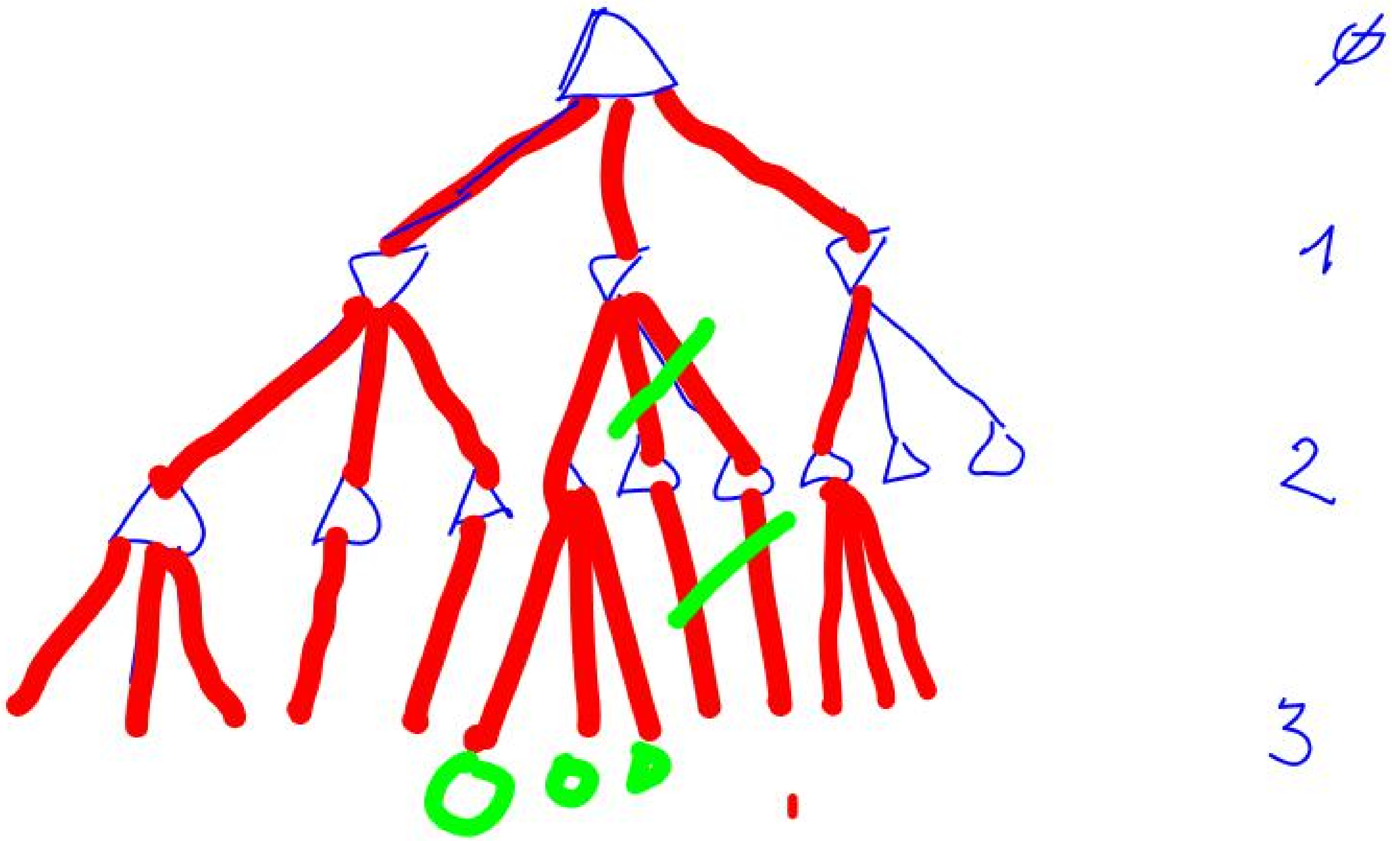
original: Time: $O(b^m)$

- how to consider next actions/moves (in what order)?
- perfect ordering?



α-β saving, sketch . . .

α-β saving, sketch . . .



je to správně	6	17%
mohu řezat dříve	29	82%

Imperfect but real-time decisions: iterative deepening

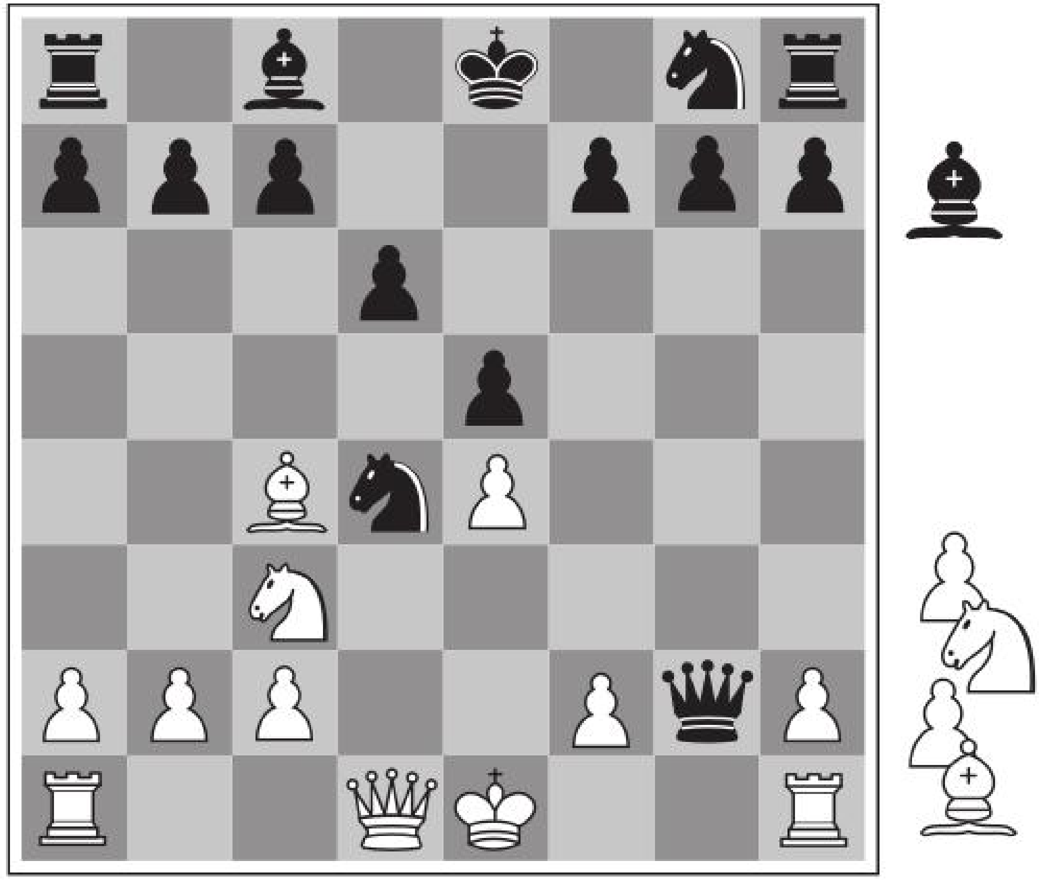
$$
\textsc{h-minimax}(s, d) =
\begin{cases}
\textsc{eval}(s) & \text{if } \textsc{cutoff-test}(s, d) \\
\max_{a \in \textsc{actions}(s)} \textsc{h-minimax}(\textsc{result}(s, a), d + 1) & \text{if } \textsc{player}(s) = \textsc{max} \\
\min_{a \in \textsc{actions}(s)} \textsc{h-minimax}(\textsc{result}(s, a, d + 1)) & \text{if } \textsc{player}(s) = \textsc{min}
\end{cases}
$$

Imperfect but real-time decisions: iterative deepening

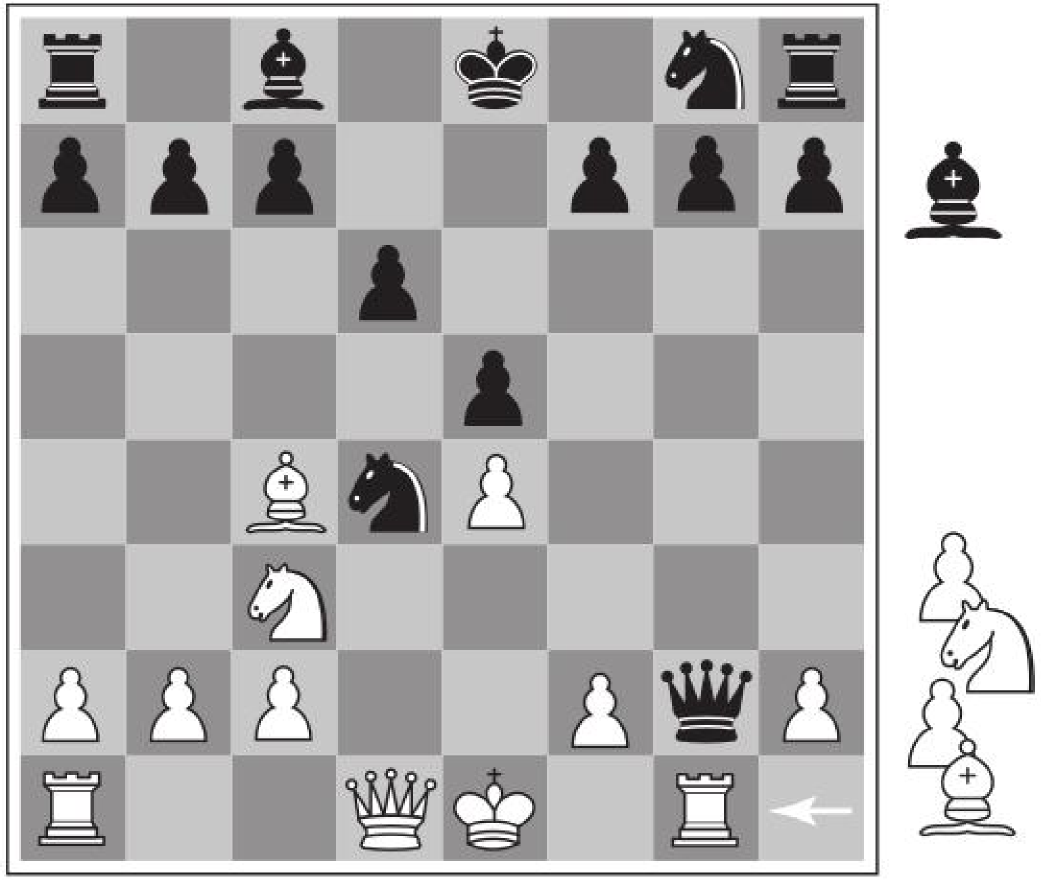
$$\text{H-MINIMAX}(s, d) = \begin{cases} \text{EVAL}(s) & \text{if } \text{CUTOFF-TEST}(s, d) \\ \max_{a \in \text{ACTIONS}(s)} \text{H-MINIMAX}(\text{RESULT}(s, a), d+1) & \text{if } \text{PLAYER}(s) = \text{MAX} \\ \min_{a \in \text{ACTIONS}(s)} \text{H-MINIMAX}(\text{RESULT}(s, a, d+1)) & \text{if } \text{PLAYER}(s) = \text{MIN} \end{cases}$$

EVAL(s) – Problems

What if something important happens just after the cut – in the next ply?



(a) White to move



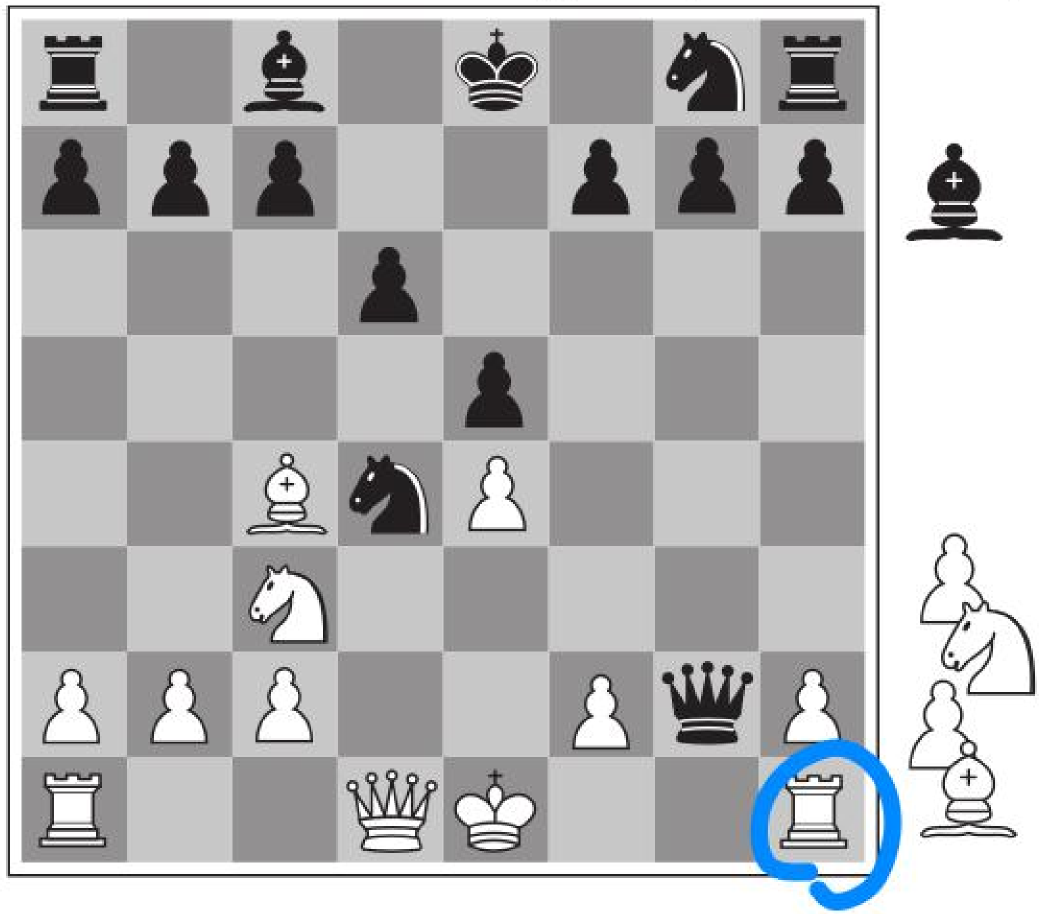
(b) White to move

Additional improvements:

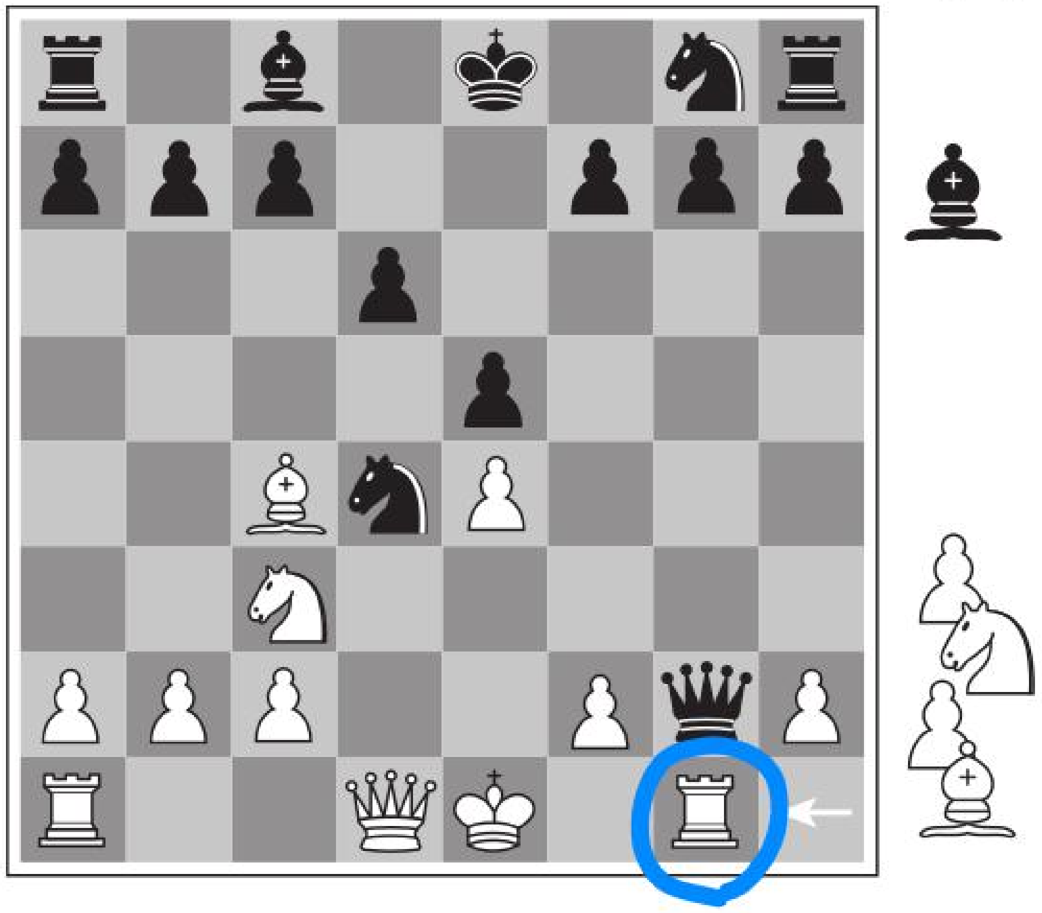
- "Killer moves"—capturing opponent's pieces, check etc.—should be considered first.
- *Quiescence search* – EVAL function should be applied only once things calm down. During capturing of pieces, depth should be locally increased.

EVAL(s) – Problems

What if something important happens just after the cut – in the next ply?



(a) White to move



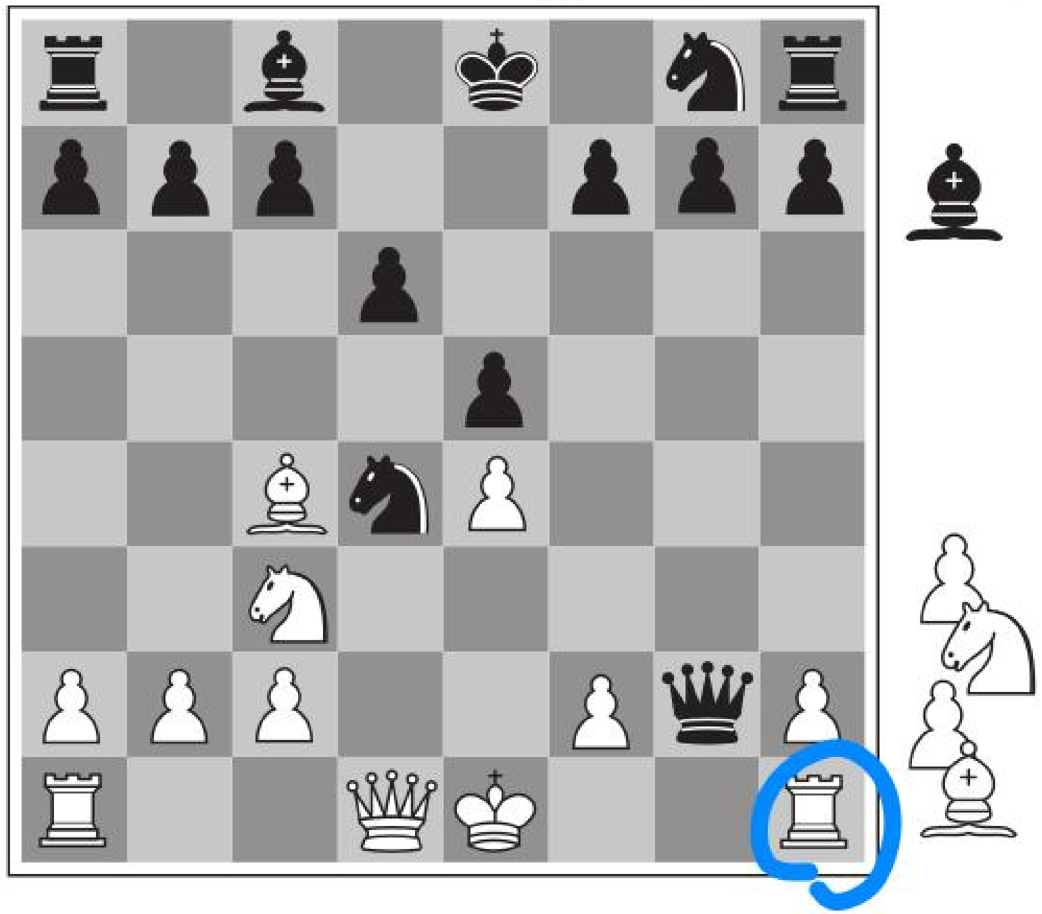
(b) White to move

Additional improvements:

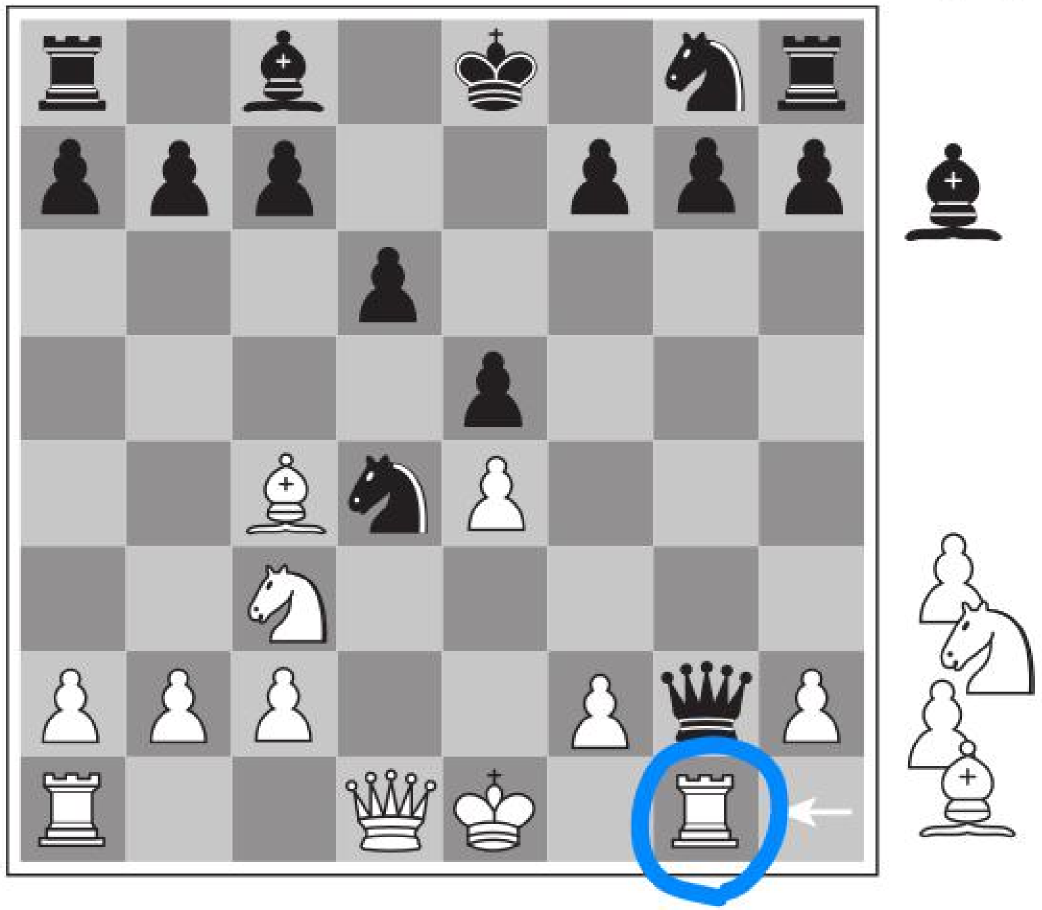
- "Killer moves"—capturing opponent's pieces, check etc.—should be considered first.
- *Quiescence search* – EVAL function should be applied only once things calm down. During capturing of pieces, depth should be locally increased.

EVAL(s) – Problems

What if something important happens just after the cut – in the next ply?



(a) White to move



(b) White to move

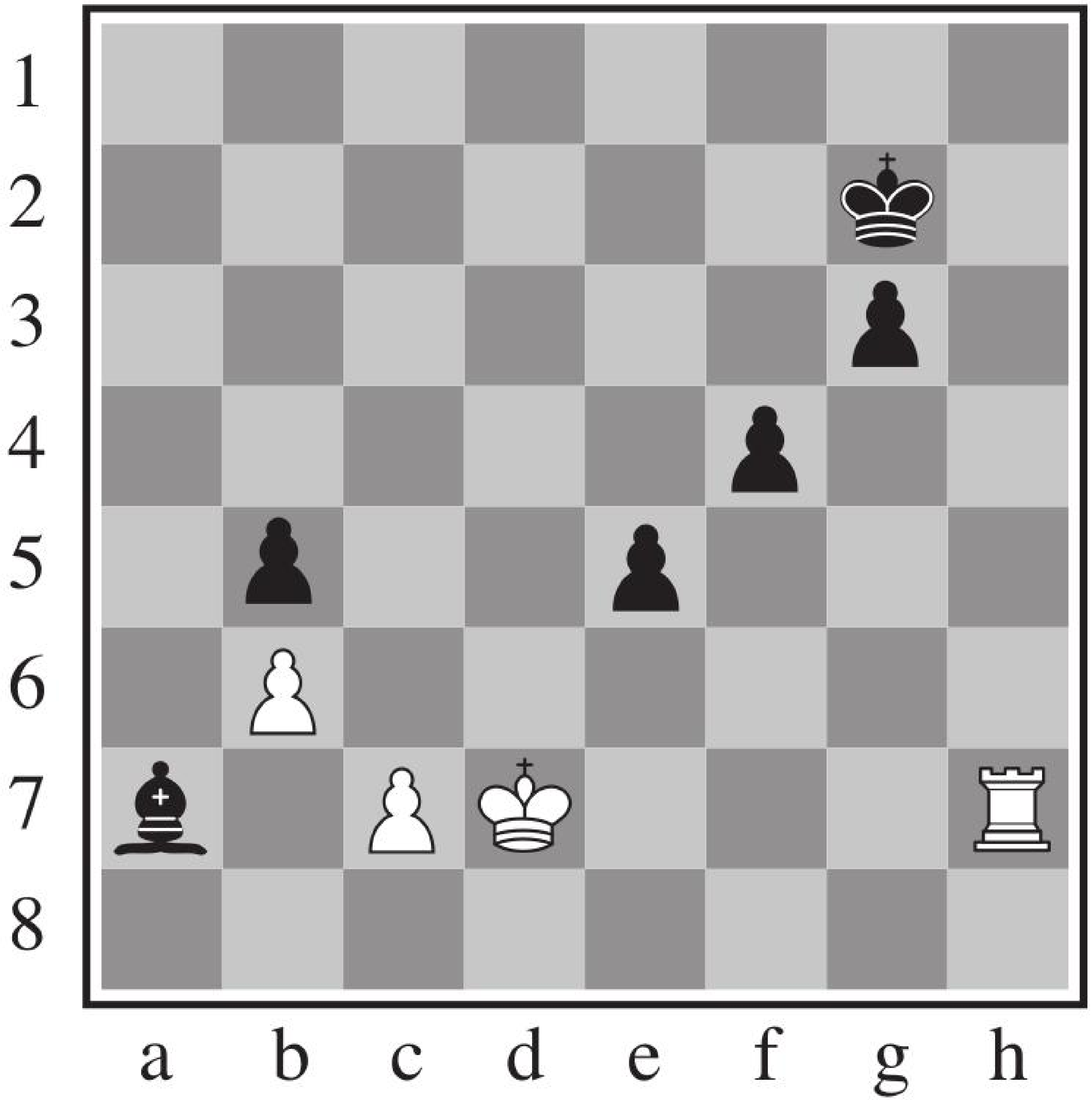
Additional improvements:

- "Killer moves"—capturing opponent's pieces, check etc.—should be considered first.
- *Quiescence search* – EVAL function should be applied only once things calm down. During capturing of pieces, depth should be locally increased.

Horizon effect

Pushing unavoidable loss deeper in tree by a delaying tactics. We know it is useless but does the machine?

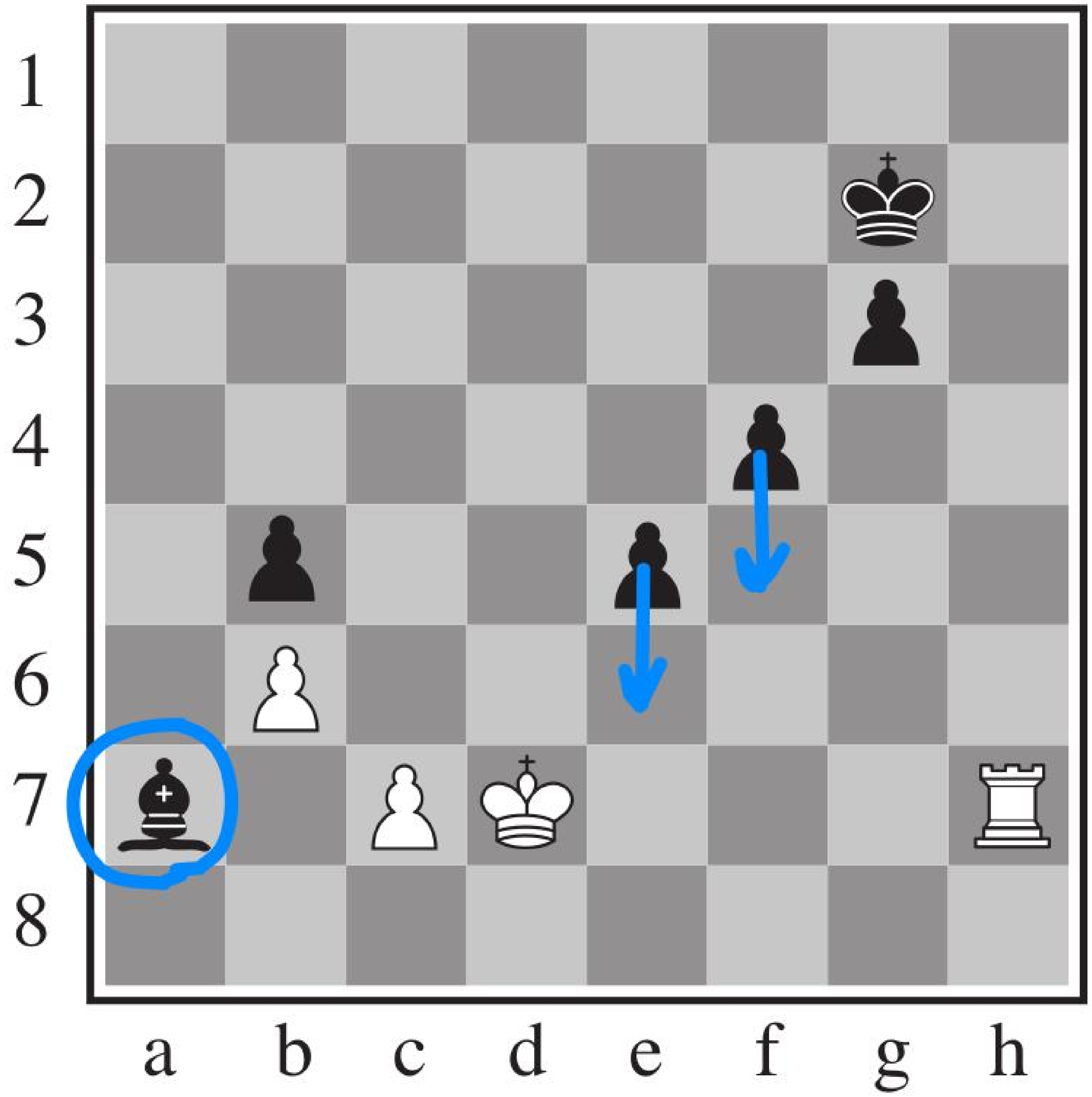
See the situation on right. Black is on move, her bishop is surely doomed. However, the inevitable loss can be postponed by moving her pawns and checking the white king. Depending on the searchable depth this may put the loss over the horizon and moving pawns may look promising.



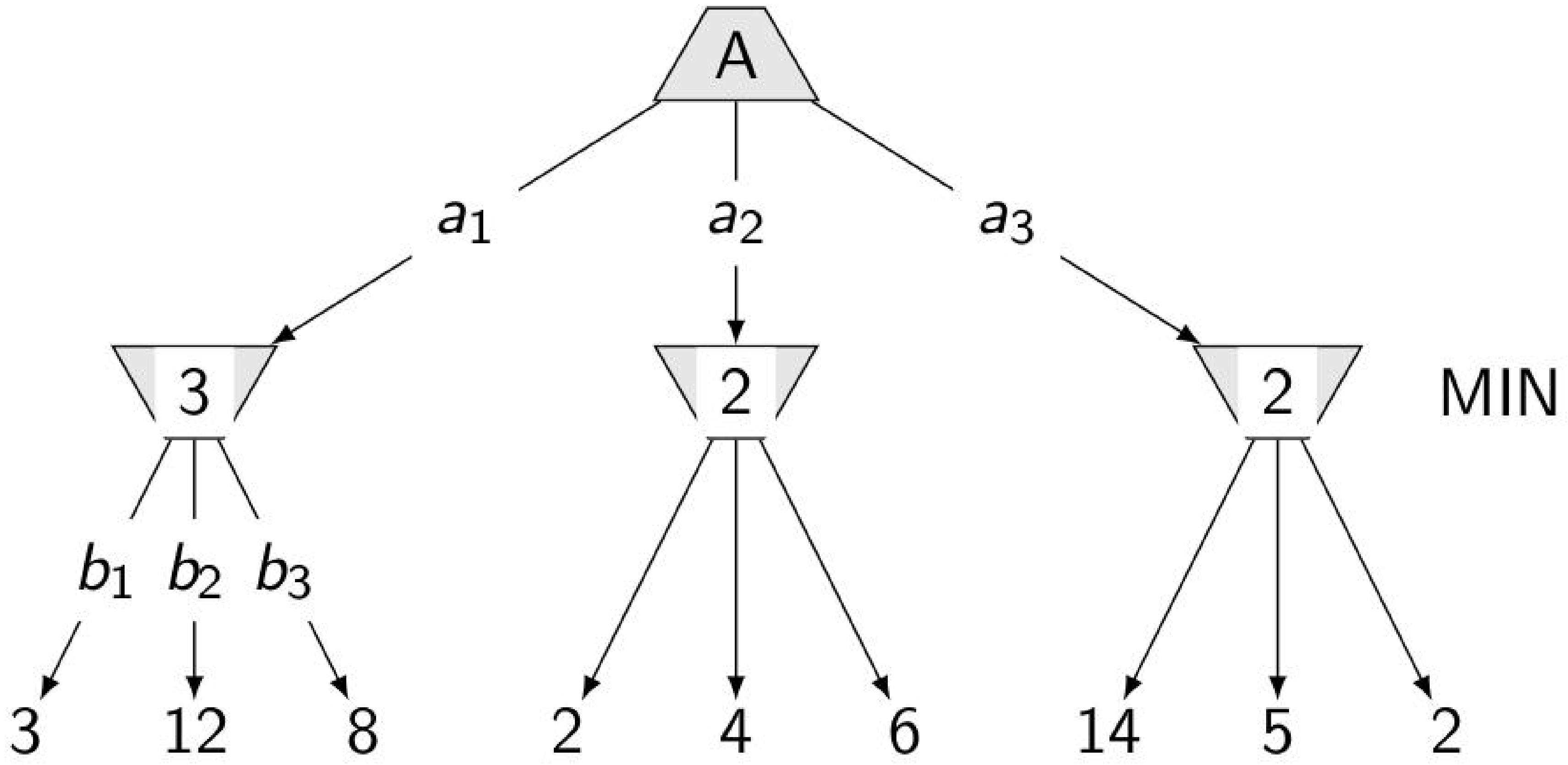
Horizon effect

Pushing unavoidable loss deeper in tree by a delaying tactics. We know it is useless but does the machine?

See the situation on right. Black is on move, her bishop is surely doomed. However, the inevitable loss can be postponed by moving her pawns and checking the white king. Depending on the searchable depth this may put the loss over the horizon and moving pawns may look promising.

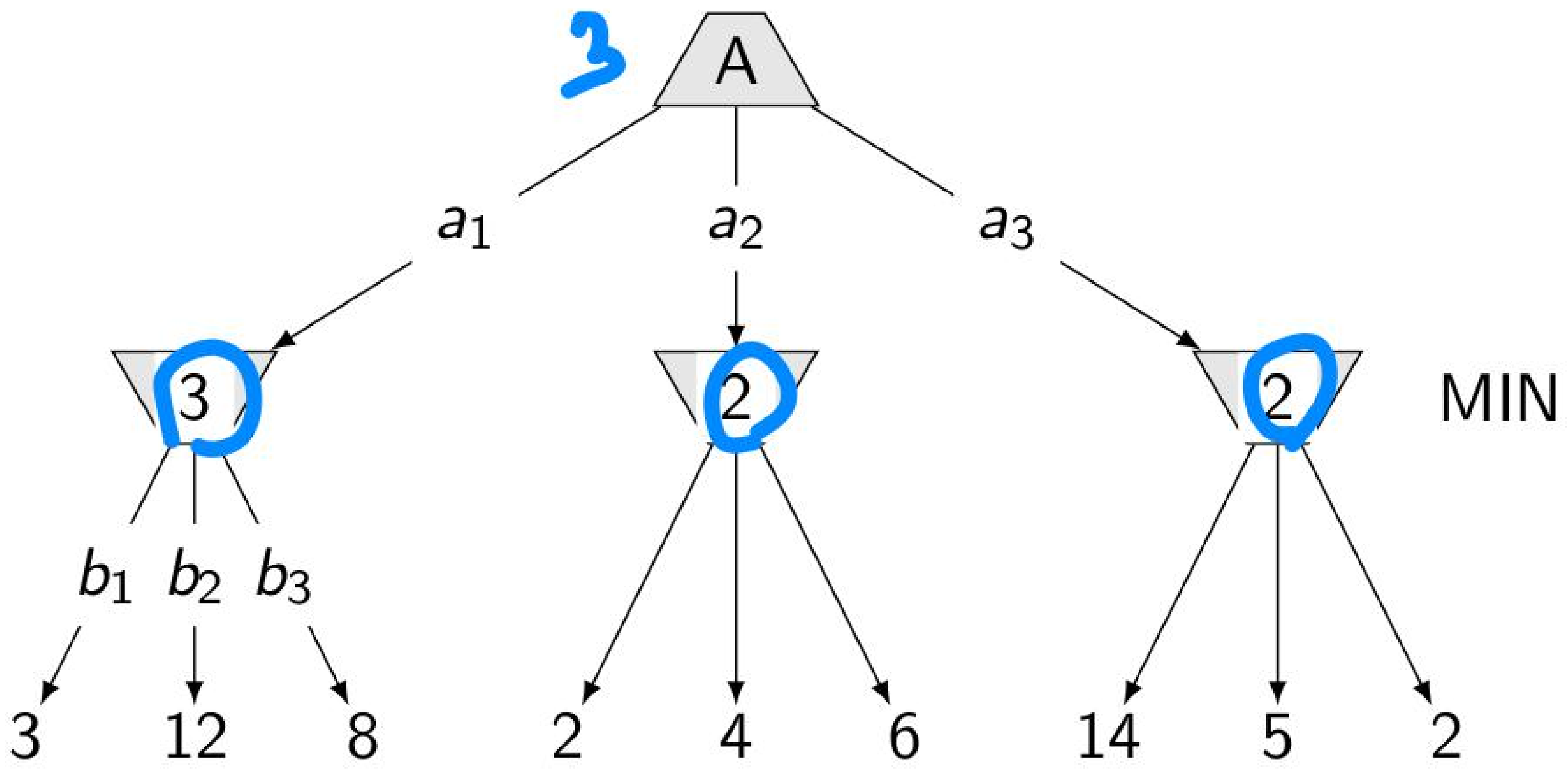


Deterministic opponent → stochastic environment



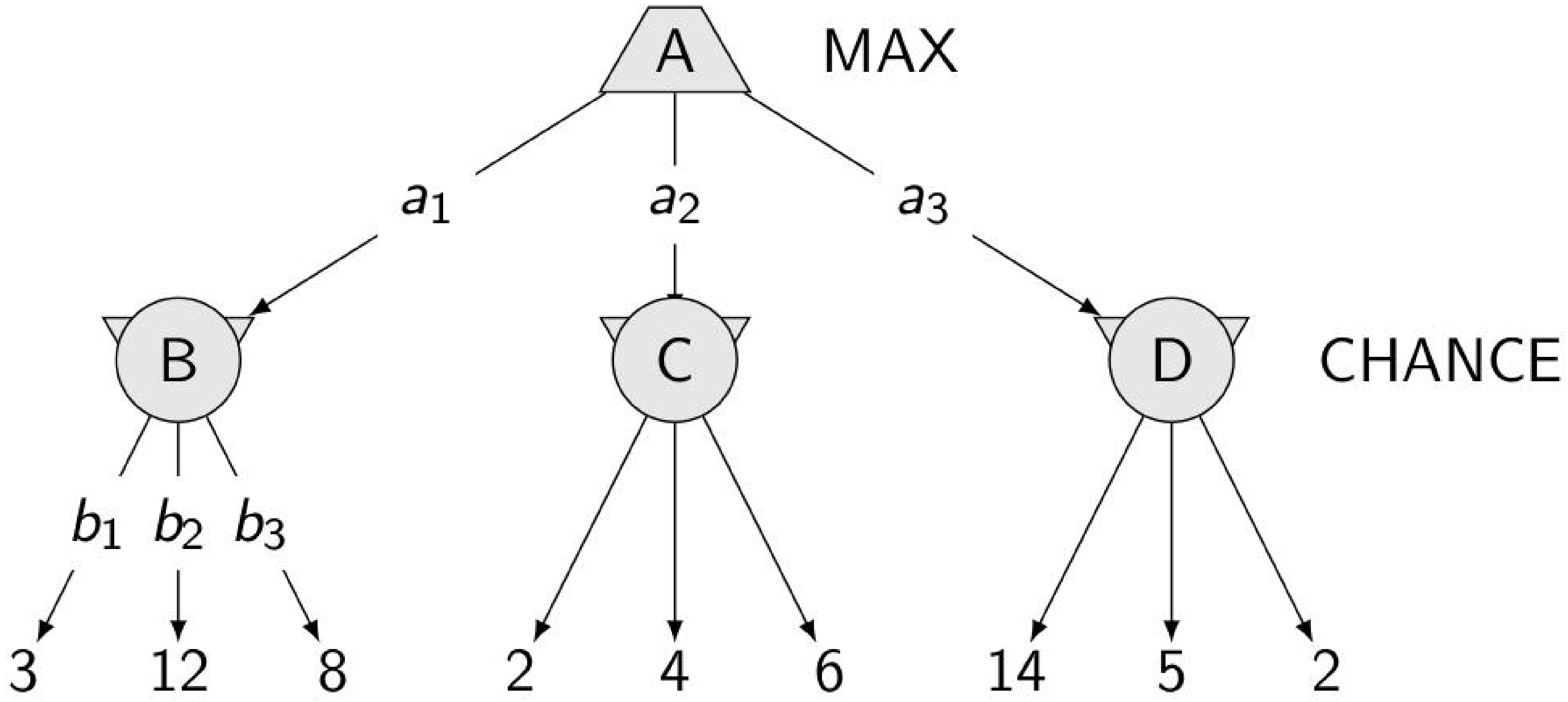
b_1, b_2, b_3 - probable branches, uncertain outcomes of a_1 action.

Deterministic opponent → stochastic environment



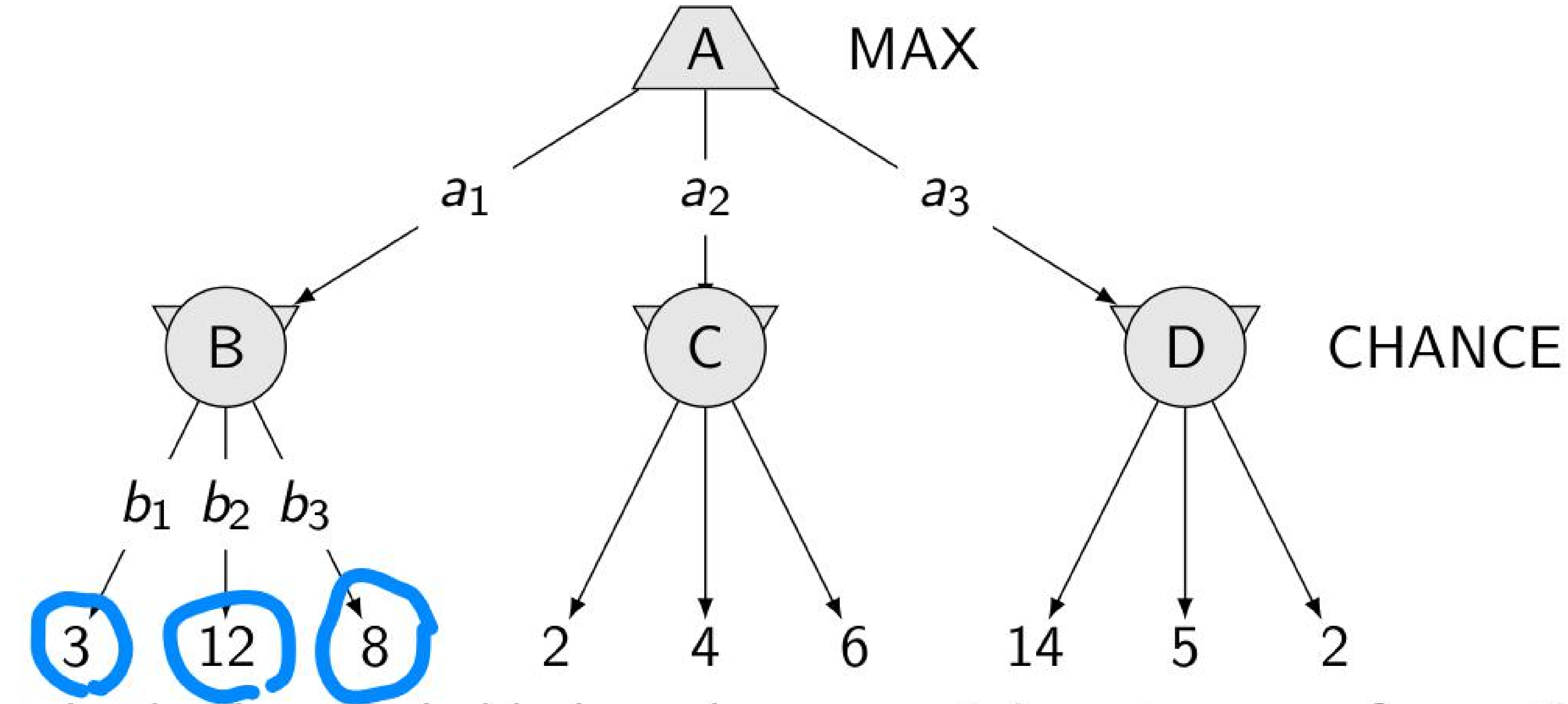
b_1, b_2, b_3 - probable branches, uncertain outcomes of a_1 action.

Deterministic opponent $\rightarrow$ stochastic environment



b_1, b_2, b_3 - probable branches, uncertain outcomes of a_1 action.

Deterministic opponent → stochastic environment



b_1, b_2, b_3 - probable branches, uncertain outcomes of a_1 action.

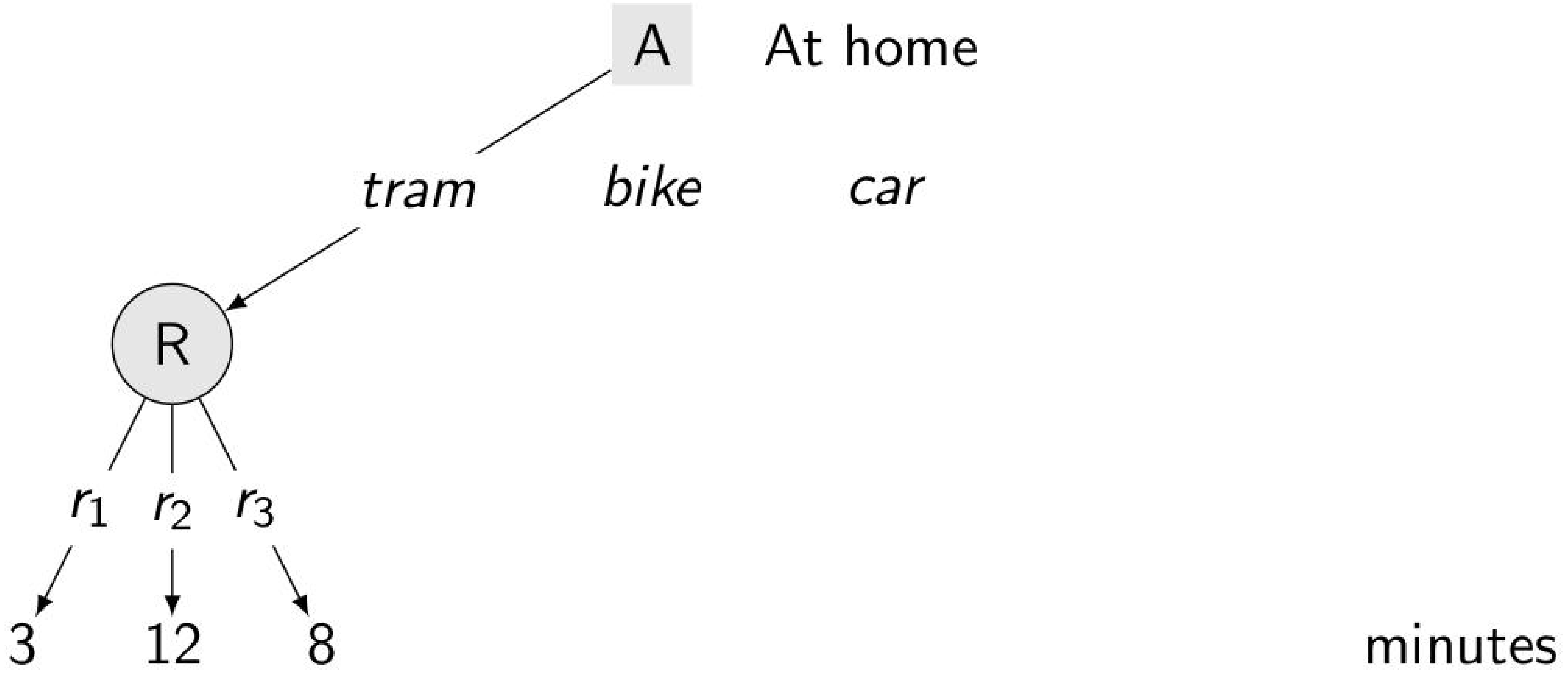
Why? Actions may fail, . . .

Video: Slipping robot. Vision for Robotics and Autonomous Systems, http://cyber.felk.cvut.cz/vras

Why? Actions may fail, . . .

Video: Slipping robot. Vision for Robotics and Autonomous Systems, http://cyber.felk.cvut.cz/vras

Why? Actions may fail, . . . , getting to work



Random variable: Situation on rails R

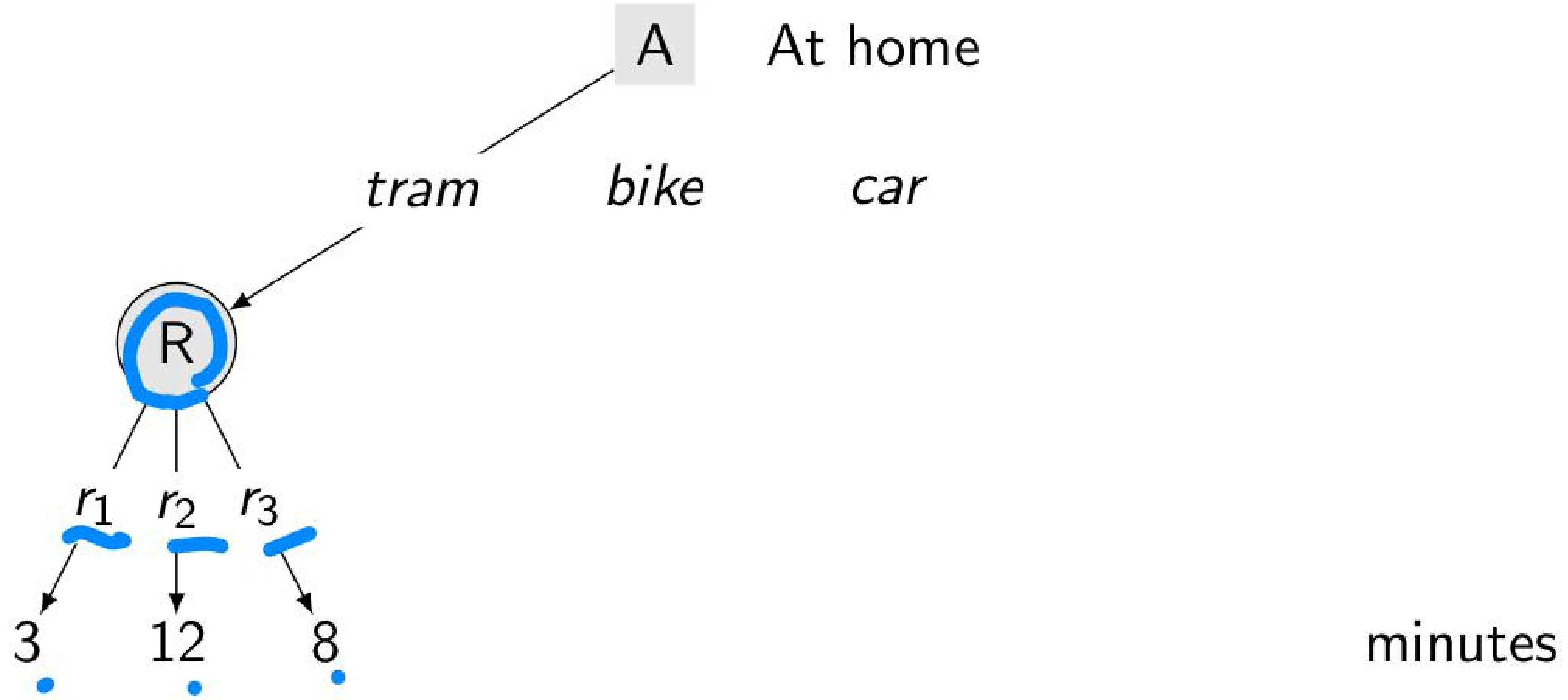
r_1 free rails

r_2 accident

r_3 congestion

MAX/MIN depends on what the $r_?$ options and terminal numbers mean. The goal may be to get to work as fast as possible.

Why? Actions may fail, . . . , getting to work



Random variable: Situation on rails R

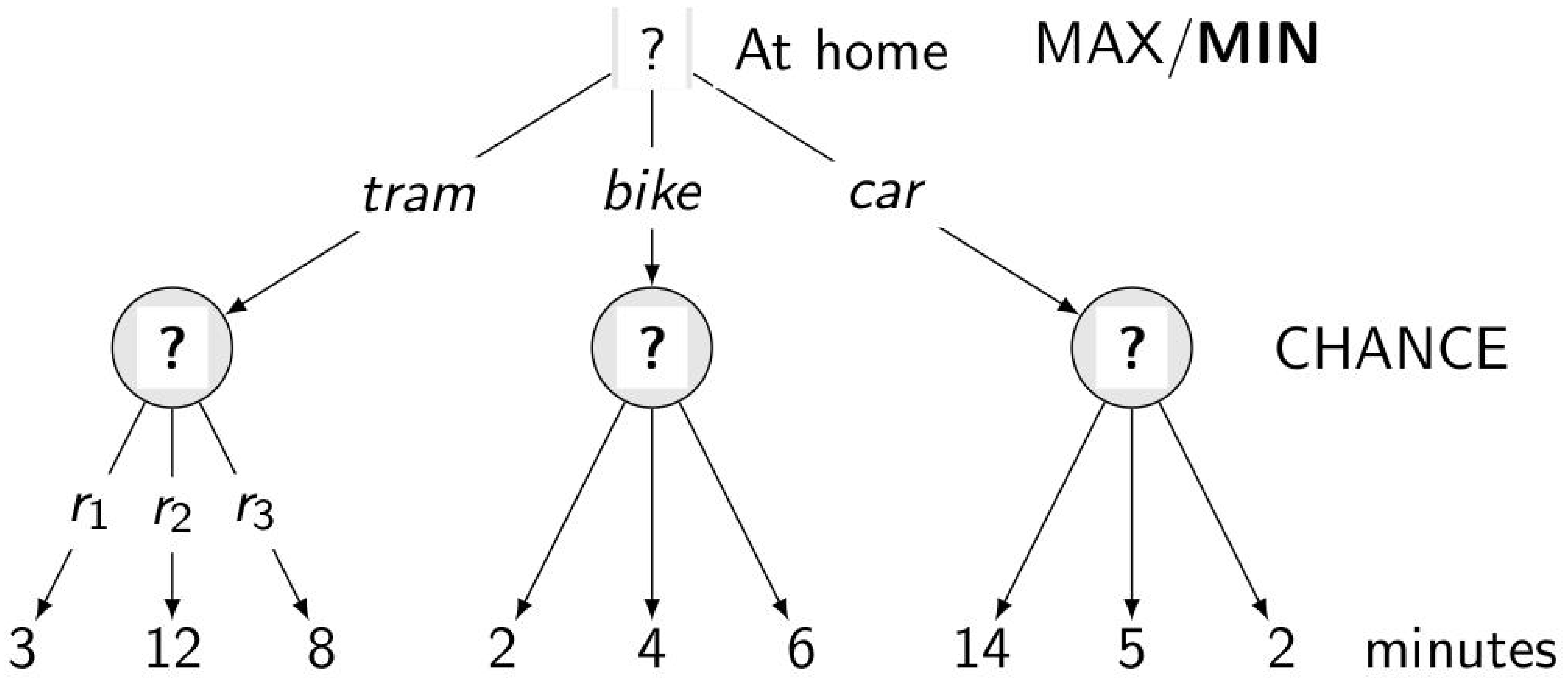
r_1 free rails

r_2 accident

r_3 congestion

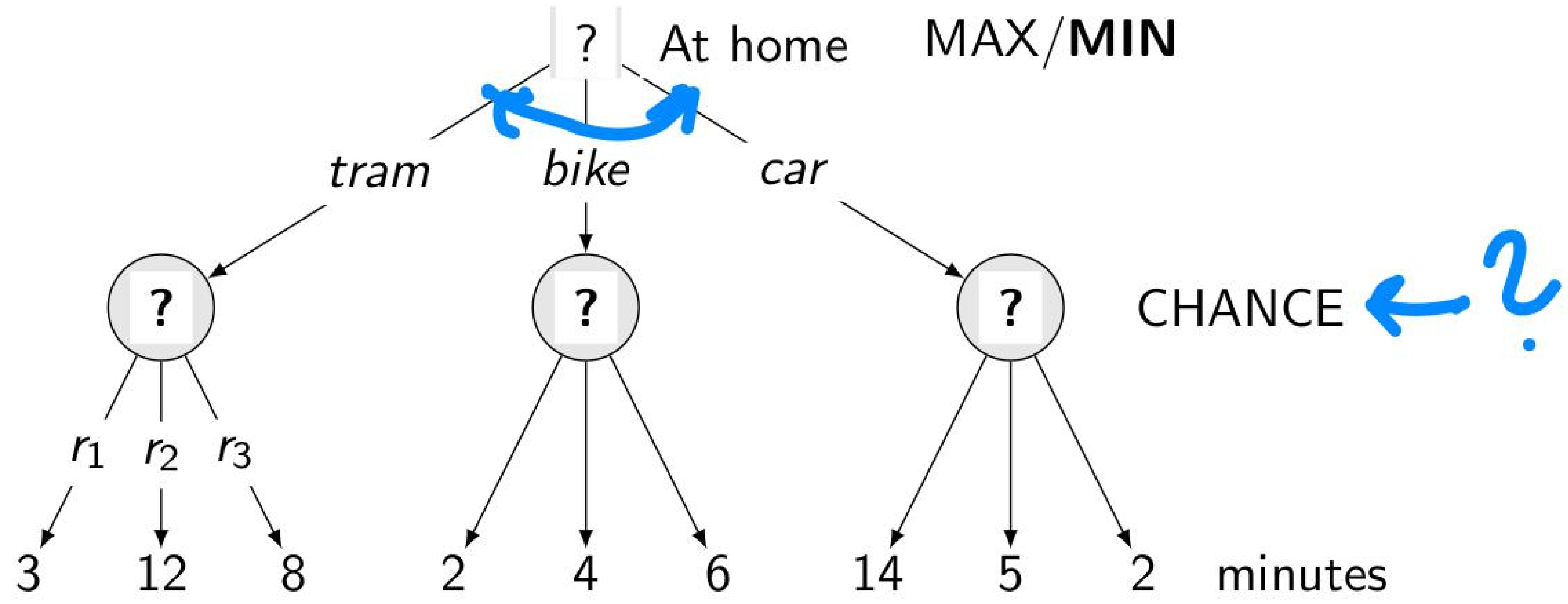
MAX/MIN depends on what the $r_?$ options and terminal numbers mean. The goal may be to get to work as fast as possible.

Chance nodes values



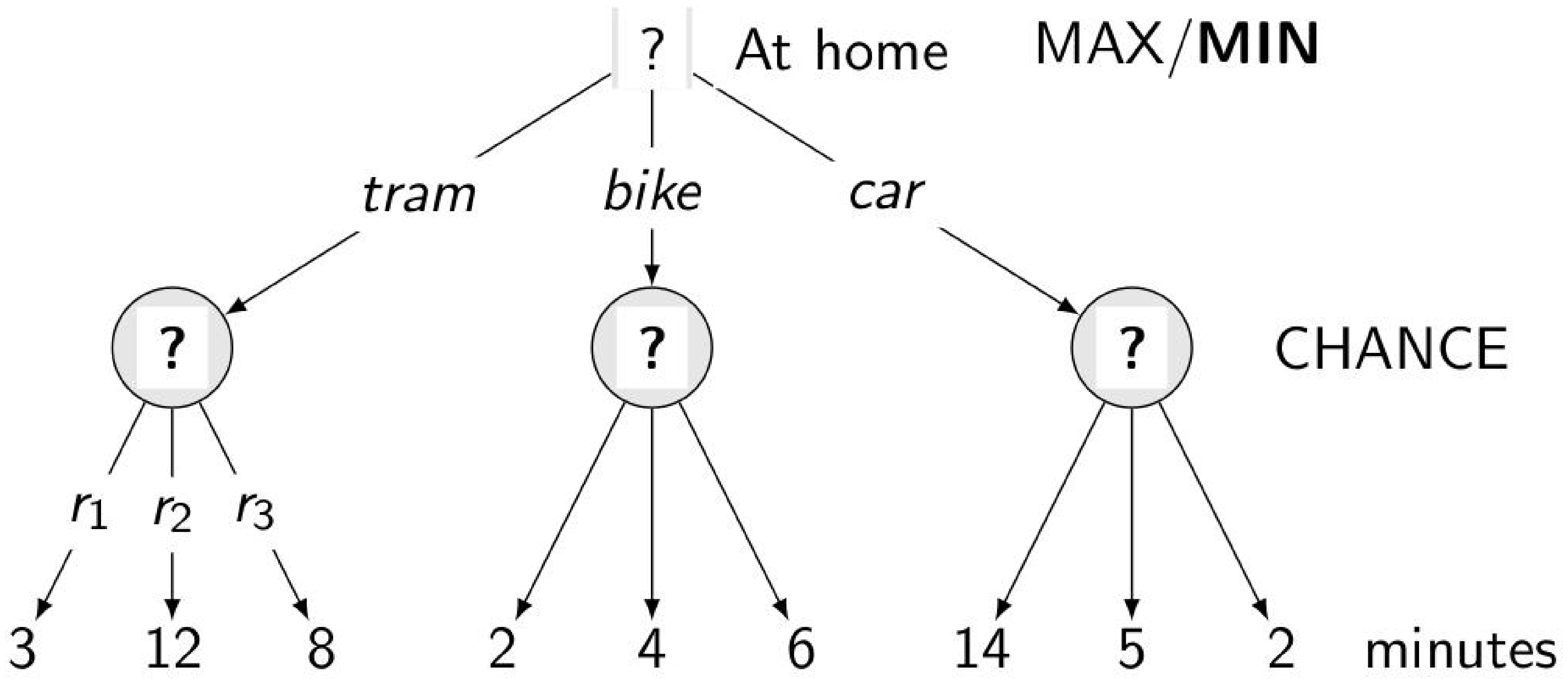
- Average case, not the worst case.
- Calculate expected utilities . . .
- i.e. take weighted average (expectation) of successors

Chance nodes values



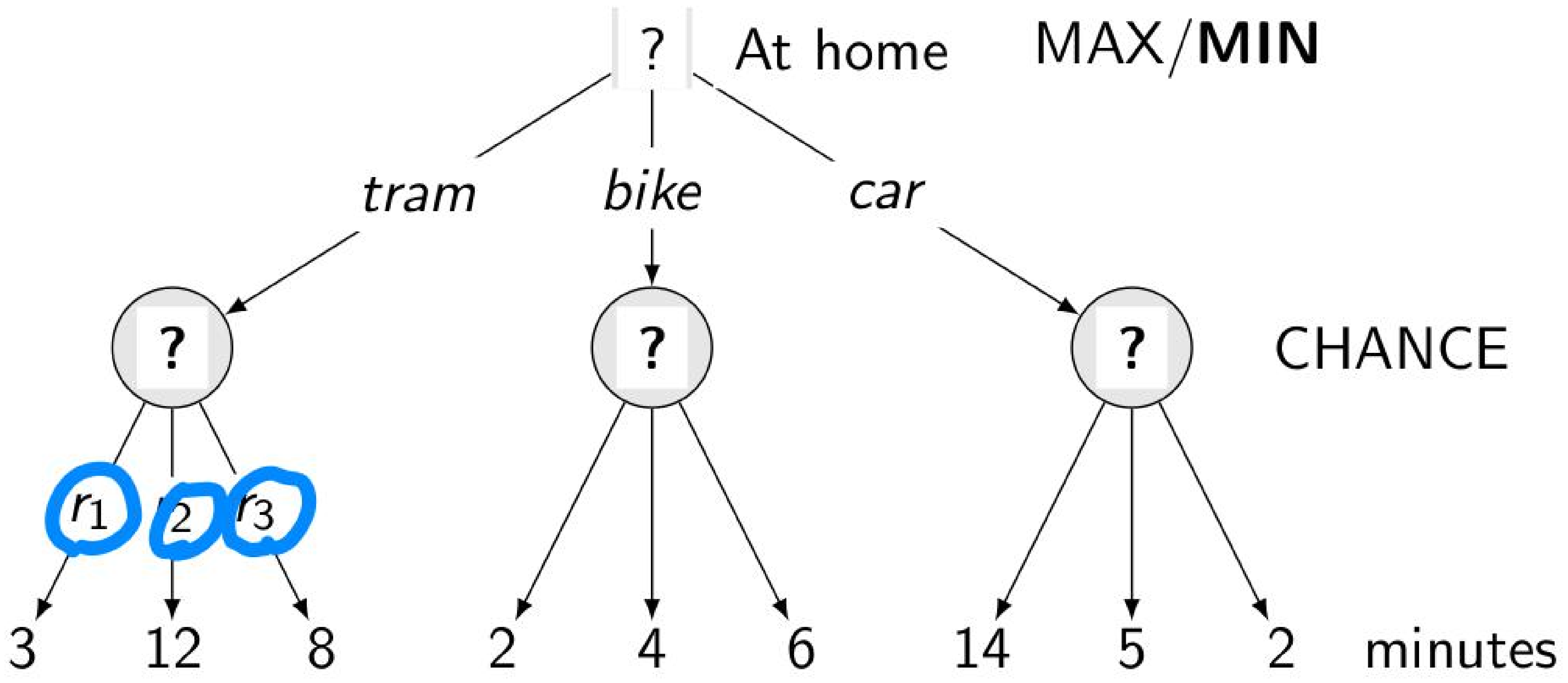
- Average case, not the worst case.
- Calculate expected utilities . . .
- i.e. take weighted average (expectation) of successors

Chance nodes values



- Average case, not the worst case.
- Calculate expected utilities ...
- i.e. take weighted average (expectation) of successors

Chance nodes values



- Average case, not the worst case.
- Calculate expected utilities ...
- i.e. take weighted average (expectation) of successors

Expectimax

function EXPECTIMAX(state) **return** a value
  if TERMINAL-TEST(state): **return** UTILITY(state)
  if state (next agent) is **MAX**: **return** MAX-VALUE(state)
  if state (next agent) is **CHANCE**: **return** EXP-VALUE(state)
end function

---

function MAX-VALUE(state) **return** value v
  $v \leftarrow -\infty$
  for a **in** ACTIONS(state) **do**
    $v \leftarrow$ **max**(v, EXPECTIMAX(RESULT(state,a)))
  end for
end function

---

function EXP-VALUE(state) **return** value v
  $v \leftarrow 0$
  for all $r \in$ random events **do**
    $v \leftarrow v + P(r)$ EXPECTIMAX(RESULT(state,r))
  end for
end function

Expectimax

function EXPECTIMAX(state) **return** a value
  if TERMINAL-TEST(state): **return** UTILITY(state)
  if state (next agent) is **MAX**: **return** MAX-VALUE(state)
  if state (next agent) is **CHANCE**: **return** EXP-VALUE(state)
end function

---

function MAX-VALUE(state) **return** value v
  $v \leftarrow -\infty$
  for a **in** ACTIONS(state) **do**
    $v \leftarrow$ **max**(v, EXPECTIMAX(RESULT(state,a)))
  end for
end function

---

function EXP-VALUE(state) **return** value v
  $v \leftarrow 0$
  for all $r \in$ random events **do**
    $v \leftarrow v + P(r)$ EXPECTIMAX(RESULT(state,r))
  end for
end function

Expectimax

function EXPECTIMAX(state) **return** a value
  if TERMINAL-TEST(state): **return** UTILITY(state)
  if state (next agent) is **MAX**: **return** MAX-VALUE(state)
  if state (next agent) is **CHANCE**: **return** EXP-VALUE(state)
end function

---

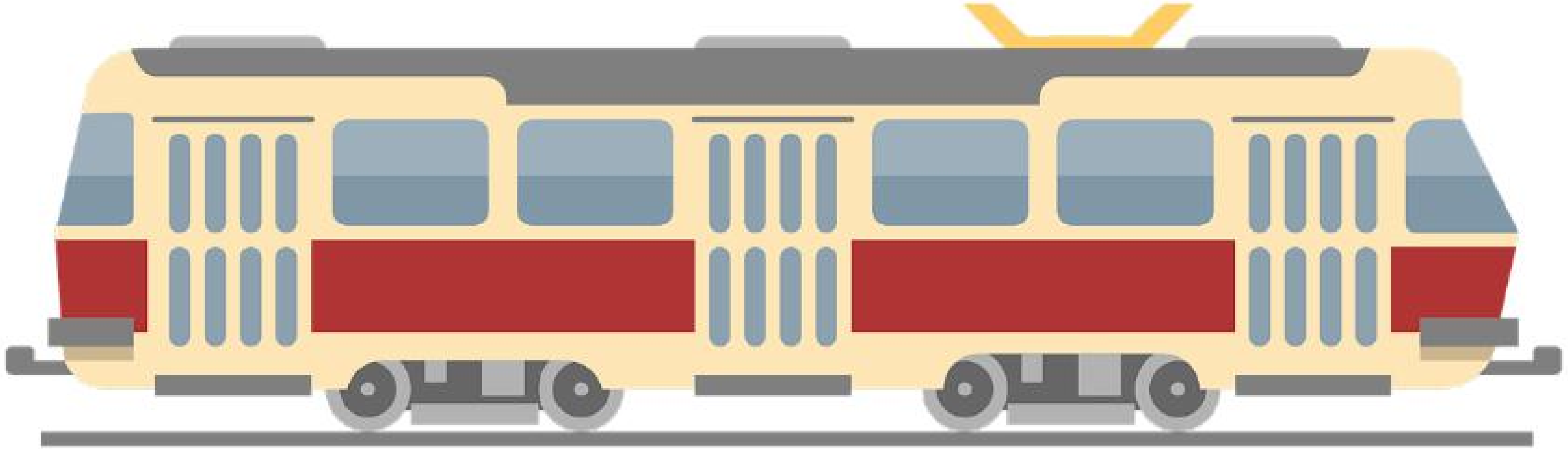
function MAX-VALUE(state) **return** value v
  $v \leftarrow -\infty$
  for a **in** ACTIONS(state) **do**
    $v \leftarrow$ **max**(v, EXPECTIMAX(RESULT(state,a)))
  end for
end function

---

function EXP-VALUE(state) **return** value v
  $v \leftarrow 0$
  for all $r \in$ random events **do**
    $v \leftarrow v + P(r)$ EXPECTIMAX(RESULT(state,r))
  end for
end function

Random variables, probability distribution, ...

- Random variable - an event with unknown outcome
- Probability distribution - assignment of weights to the outcomes



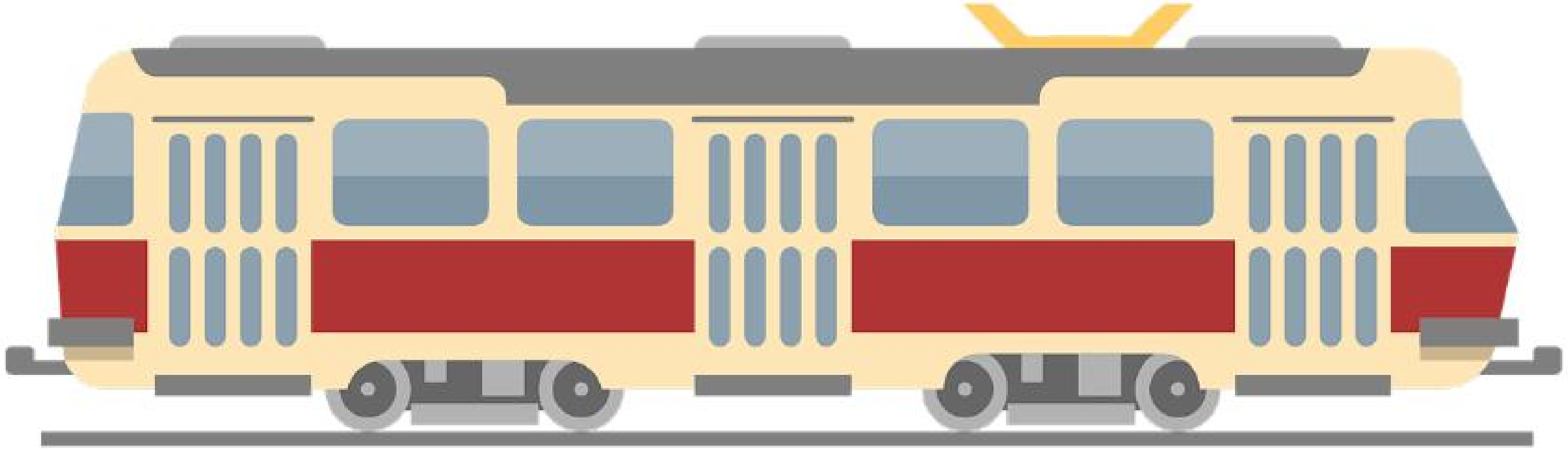
- Random variable: R - situation on rails
- Outcomes/events: $r \in \{\text{free rails, accindent, congestion}\}$
- Probability distribution: $P(R = \text{free rails}) = 0.3$, $P(R = \text{accident}) = 0.1$, $P(R = \text{congestion}) = 0.6$

Few reminders from laws of probability. Probabilities:

- always non-negative,
- sum over all possible outcomes is equal to 1.

Random variables, probability distribution, . . .

- Random variable - an event with unknown outcome
- Probability distribution - assignment of weights to the outcomes



- Random variable: R - situation on rails
- Outcomes/events: $r \in \{$free rails, accindent, congestion$\}$
- Probability distribution: $P(R = \text{free rails}) = 0.3$, $P(R = \text{accident}) = 0.1$, $P(R = \text{congestion}) = 0.6$

Few reminders from laws of probability. Probabilities:

- always non-negative,
- sum over all possible outcomes is equal to 1.

Random variables, probability distribution, . . .

- Random variable - an event with unknown outcome
- Probability distribution - assignment of weights to the outcomes

- Random variable: R - situation on rails
- Outcomes/events: $r \in \{\text{free rails, accindent, congestion}\}$
- Probability distribution: $P(R = \text{free rails}) = 0.3$, $P(R = \text{accident}) = 0.1$, $P(R = \text{congestion}) = 0.6$

Few reminders from laws of probability. Probabilities:

- always non-negative,
- sum over all possible outcomes is equal to 1.

Random variables, probability distribution, . . .

- **Random variable** - an event with unknown outcome
- **Probability distribution** - assignment of weights to the outcomes

- Random variable: R - situation on rails
- Outcomes/events: $r \in \{\text{free rails, accindent, congestion}\}$
- Probability distribution: $P(R = \text{free rails}) = 0.3$, $P(R = \text{accident}) = 0.1$, $P(R = \text{congestion}) = 0.6$

Few reminders from laws of probability. Probabilities:

- always non-negative,
- sum over all possible outcomes is equal to 1.

Expectations, . . .

How long does it take to go to work by tram?

- Depends on the random variable R - situation on rails with possible events r_1, r_2, r_3.
- What is the expectation of the time?

$$t = P(r_1)t_1 + P(r_2)t_2 + P(r_3)t_3$$

Weighted average.

Expectations, . . .

How long does it take to go to work by tram?

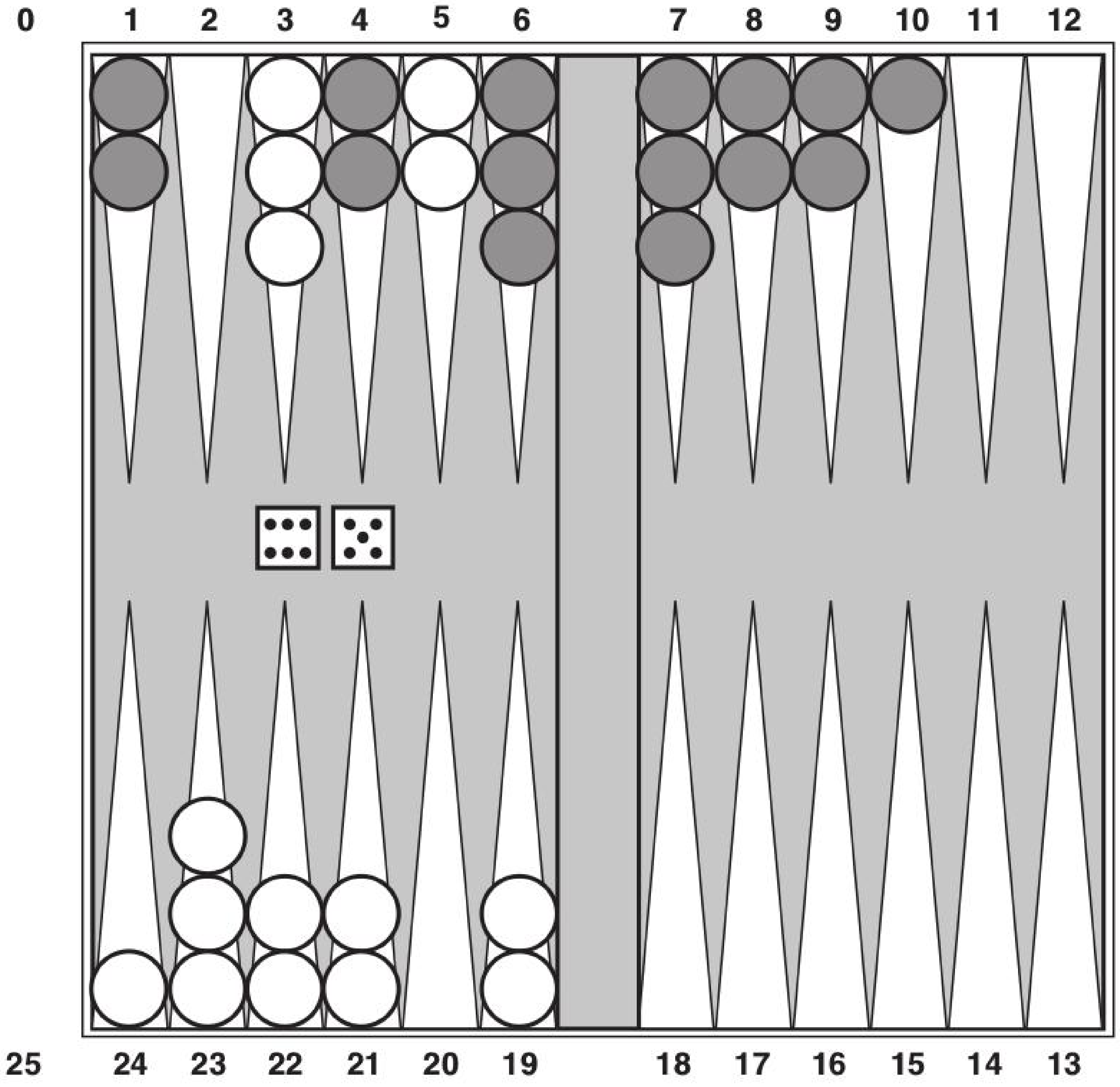
- Depends on the random variable R - situation on rails with possible events r_1, r_2, r_3.
- What is the expectation of the time?

R

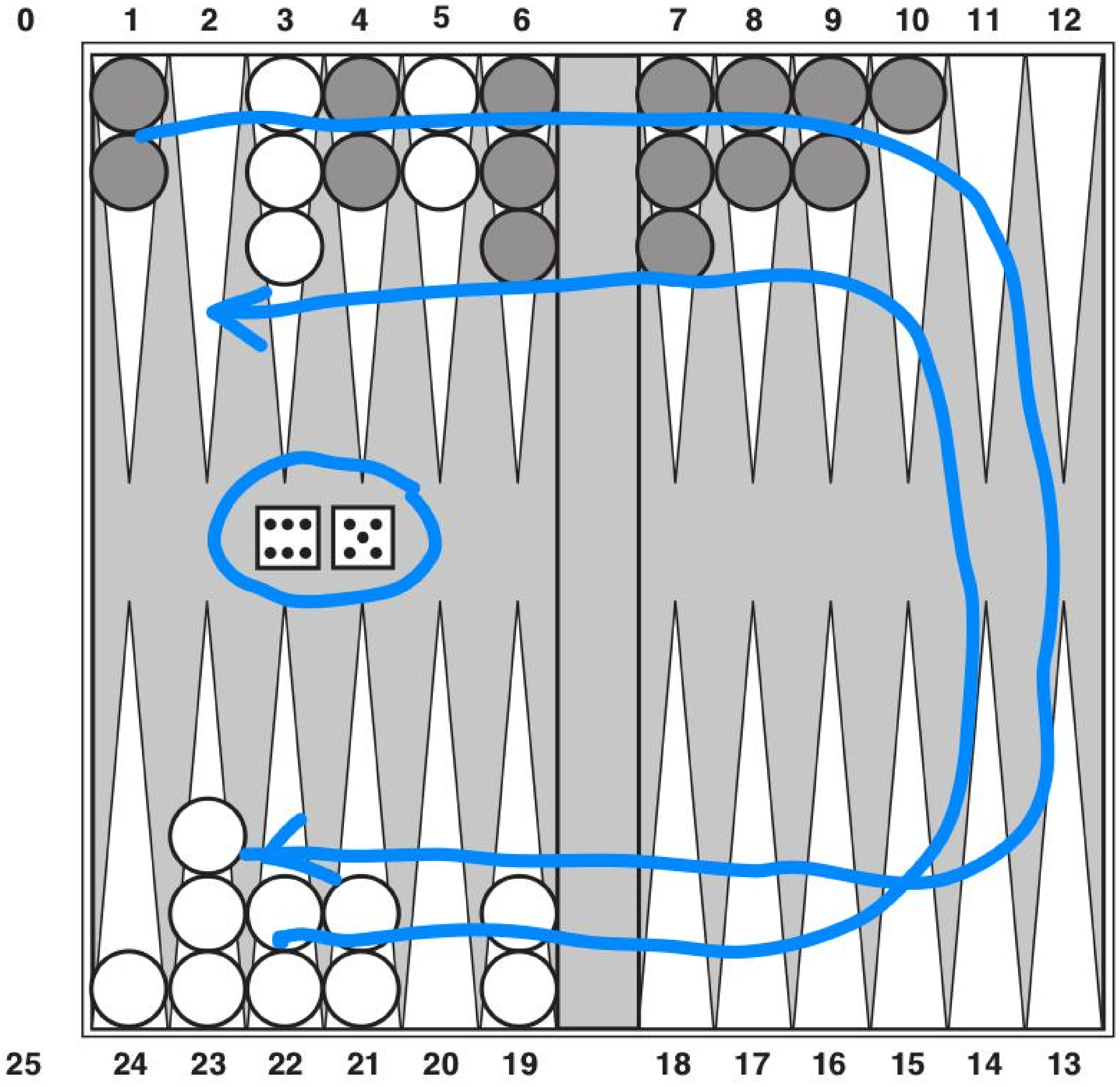
$$t = P(r_1)t_1 + P(r_2)t_2 + P(r_3)t_3$$

Weighted average.

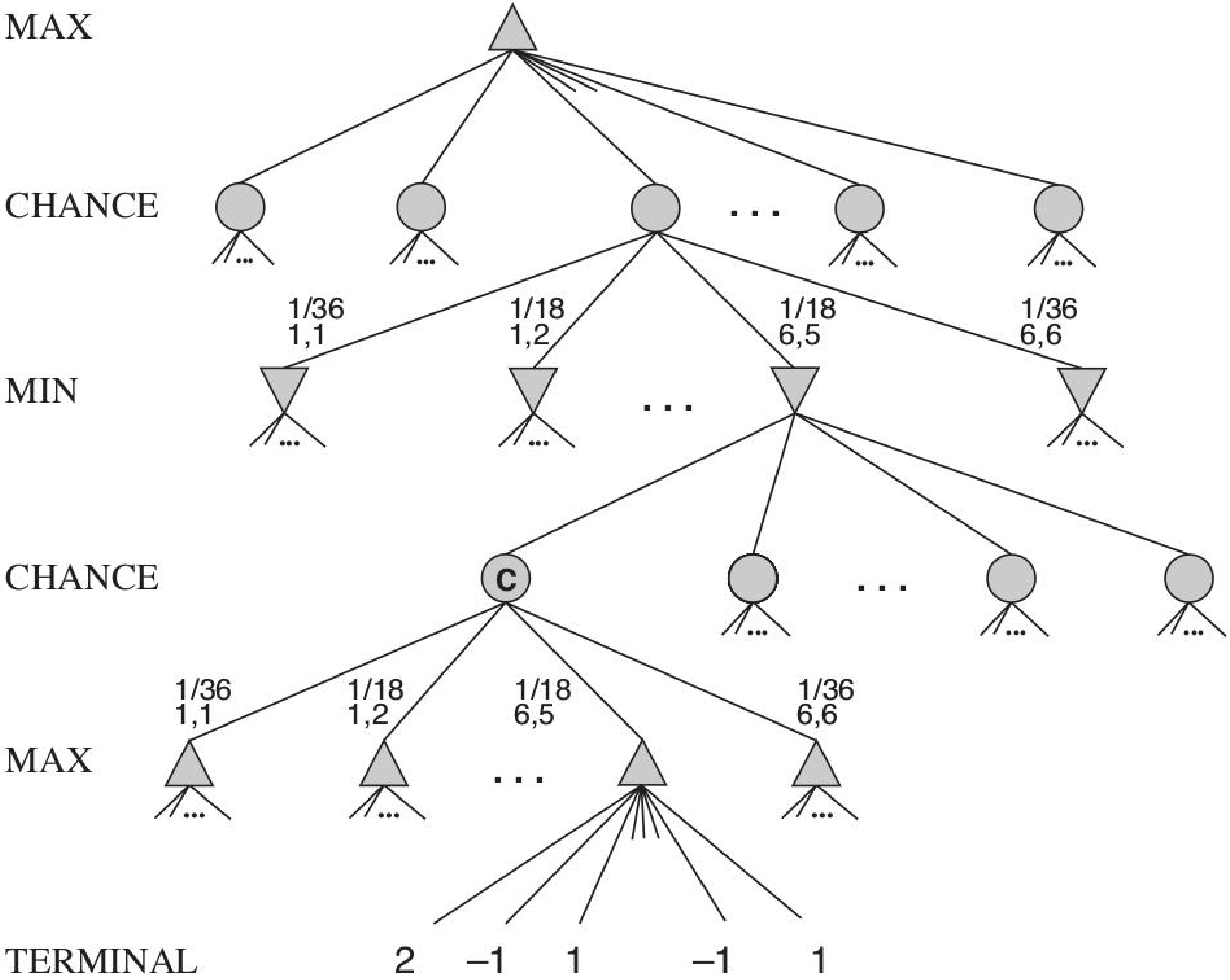
Games with chance **and** strategy



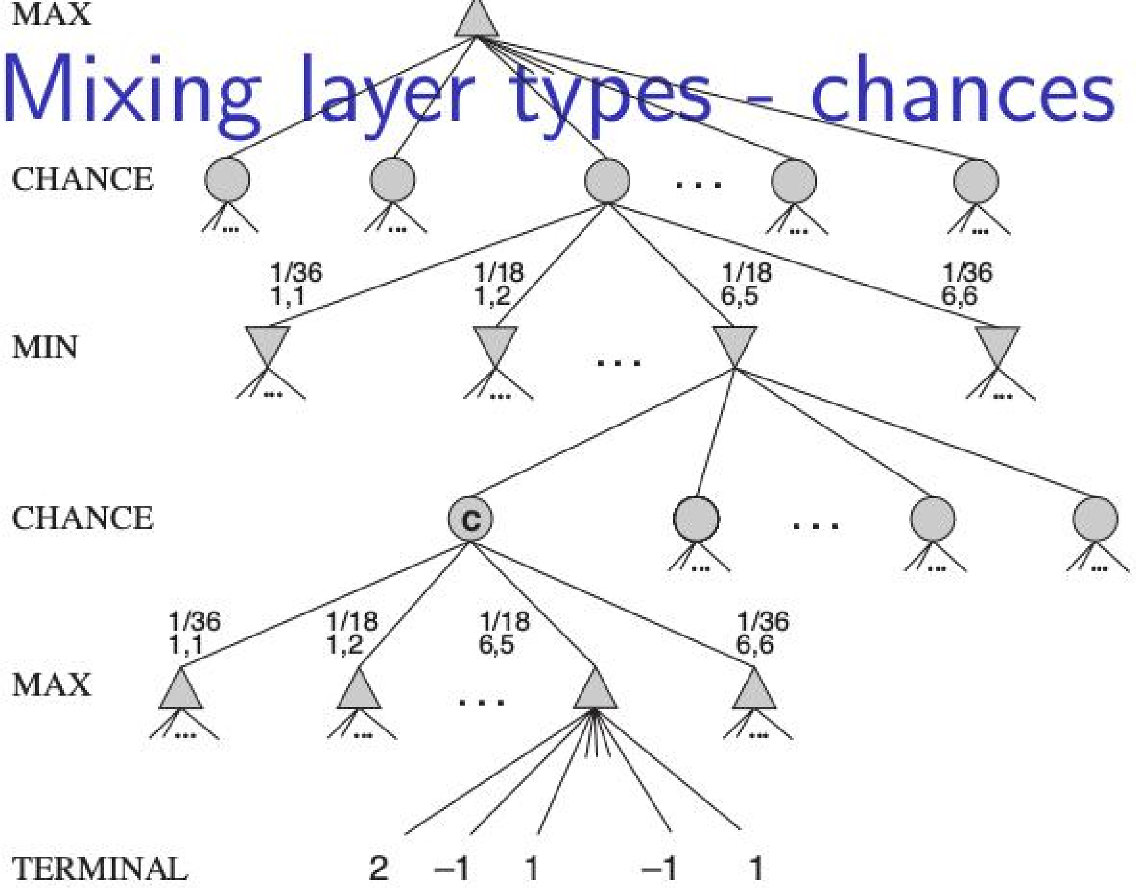
Games with chance **and** strategy

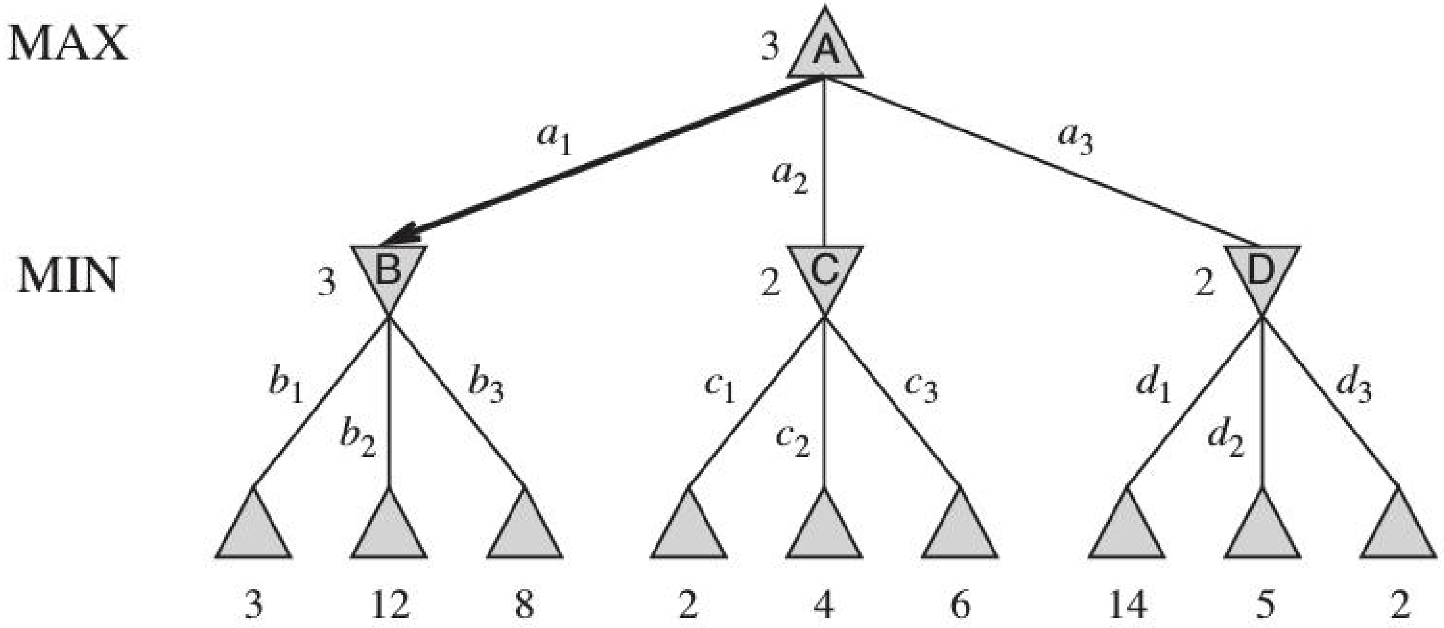


Mixing MAX, CHANCE, and MIN nodes



Mixing layer types - chances inserted

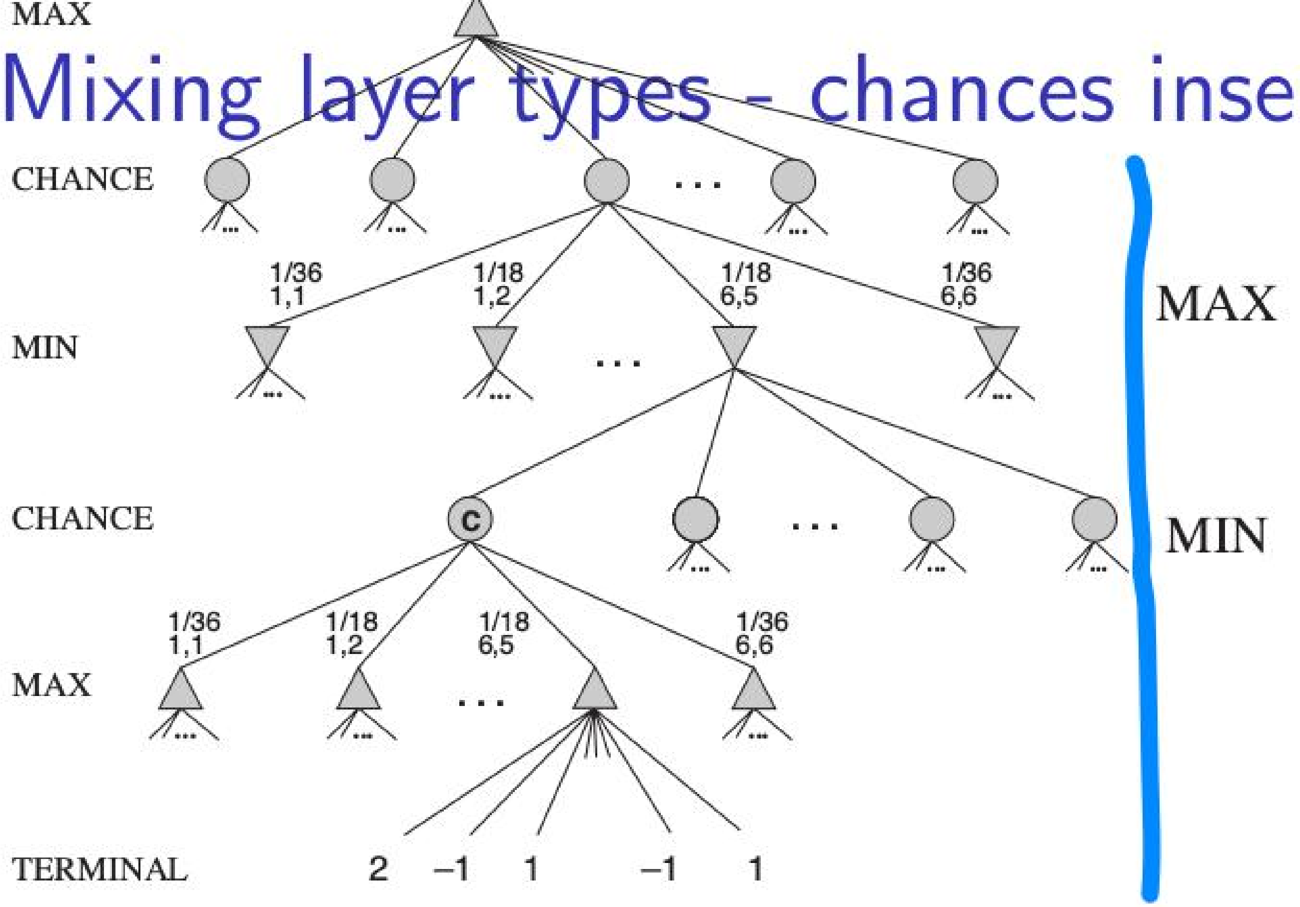


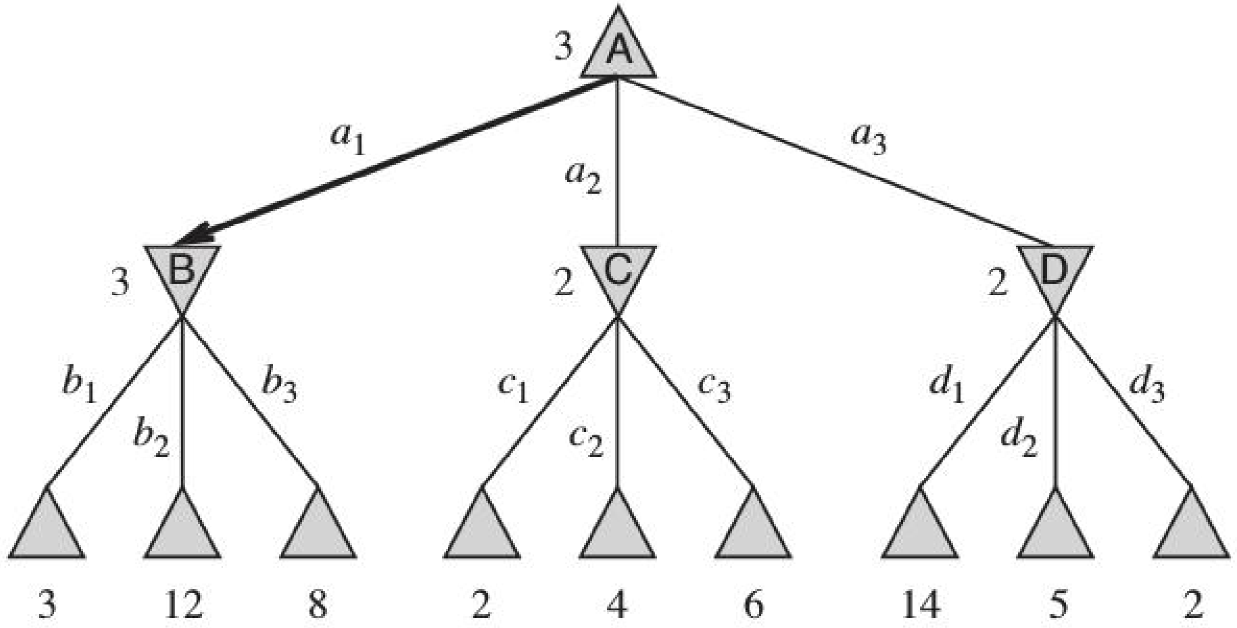


Extra random agent that moves after each MAX and MIN agent

$$\text{EXPECTIMINIMAX}(s) = \begin{cases} \text{UTILITY}(s) & \text{if} \quad \text{TERMINAL-TEST}(s) \\ \max_a \text{EXPECTIMINIMAX}(\text{RESULT}(s,a)) & \text{if} \quad \text{PLAYER}(s) = \text{MAX} \\ \min_a \text{EXPECTIMINIMAX}(\text{RESULT}(s,a)) & \text{if} \quad \text{PLAYER}(s) = \text{MIN} \\ \sum_r P(r)\text{EXPECTIMINIMAX}(\text{RESULT}(s,r)) & \text{if} \quad \text{PLAYER}(s) = \text{CHANCE} \end{cases}$$

Mixing layer types - chances inserted

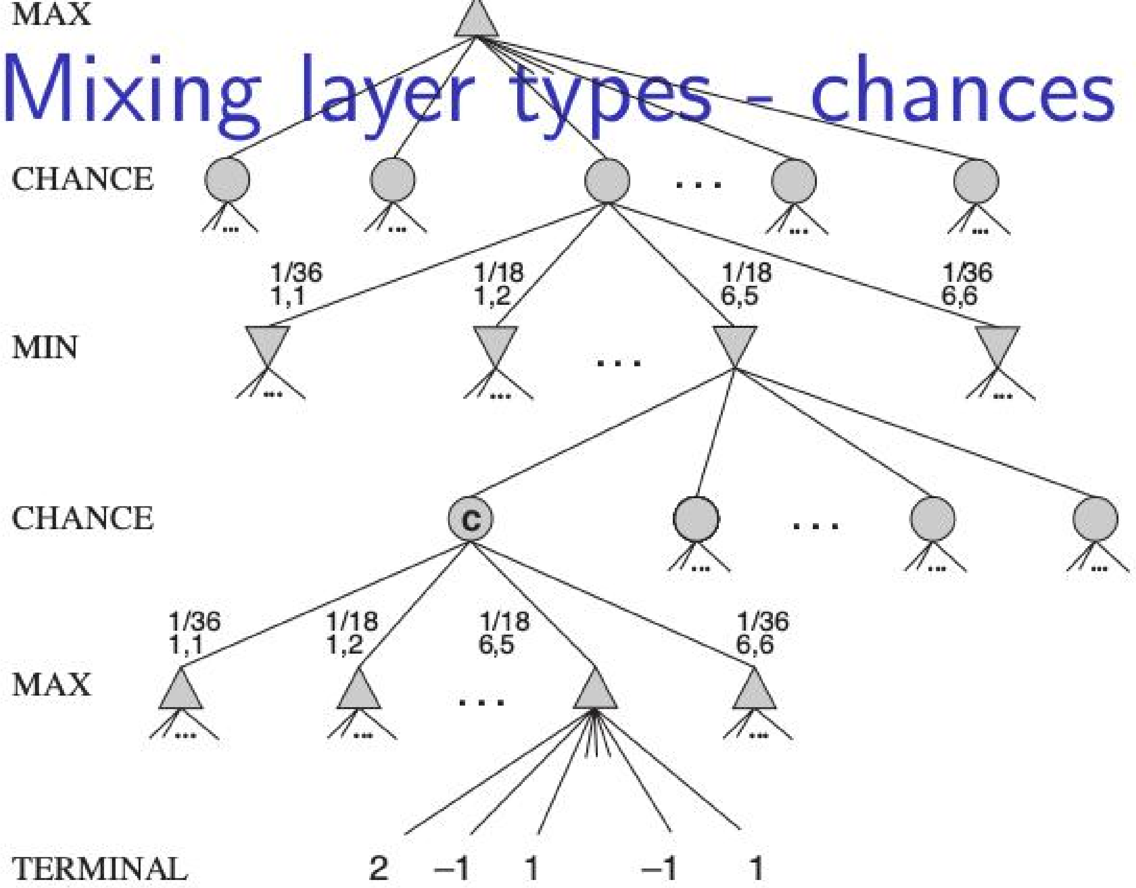


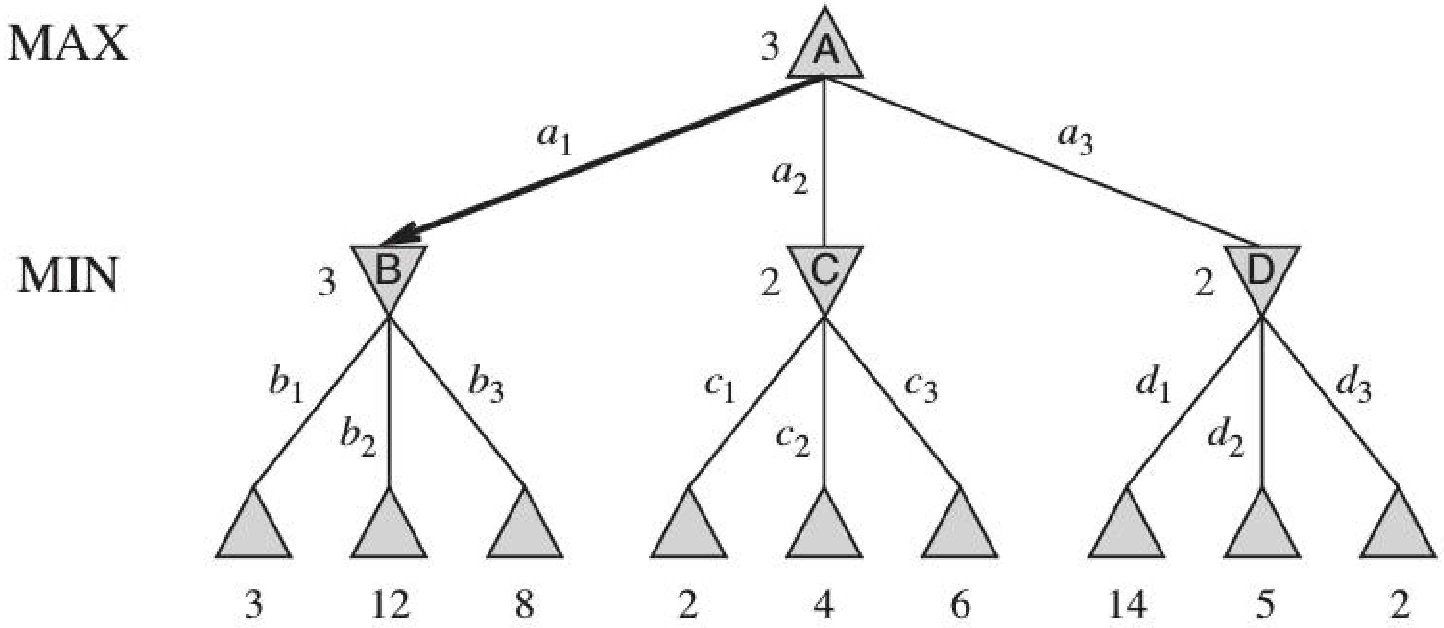


Extra random agent that moves after each MAX and MIN agent

$$\text{EXPECTIMINIMAX}(s) = \begin{cases} \text{UTILITY}(s) & \text{if} \quad \text{TERMINAL-TEST}(s) \\ \max_a \text{EXPECTIMINIMAX}(\text{RESULT}(s,a)) & \text{if} \quad \text{PLAYER}(s) = \text{MAX} \\ \min_a \text{EXPECTIMINIMAX}(\text{RESULT}(s,a)) & \text{if} \quad \text{PLAYER}(s) = \text{MIN} \\ \sum_r P(r)\text{EXPECTIMINIMAX}(\text{RESULT}(s,r)) & \text{if} \quad \text{PLAYER}(s) = \text{CHANCE} \end{cases}$$

Mixing layer types - chances inserted

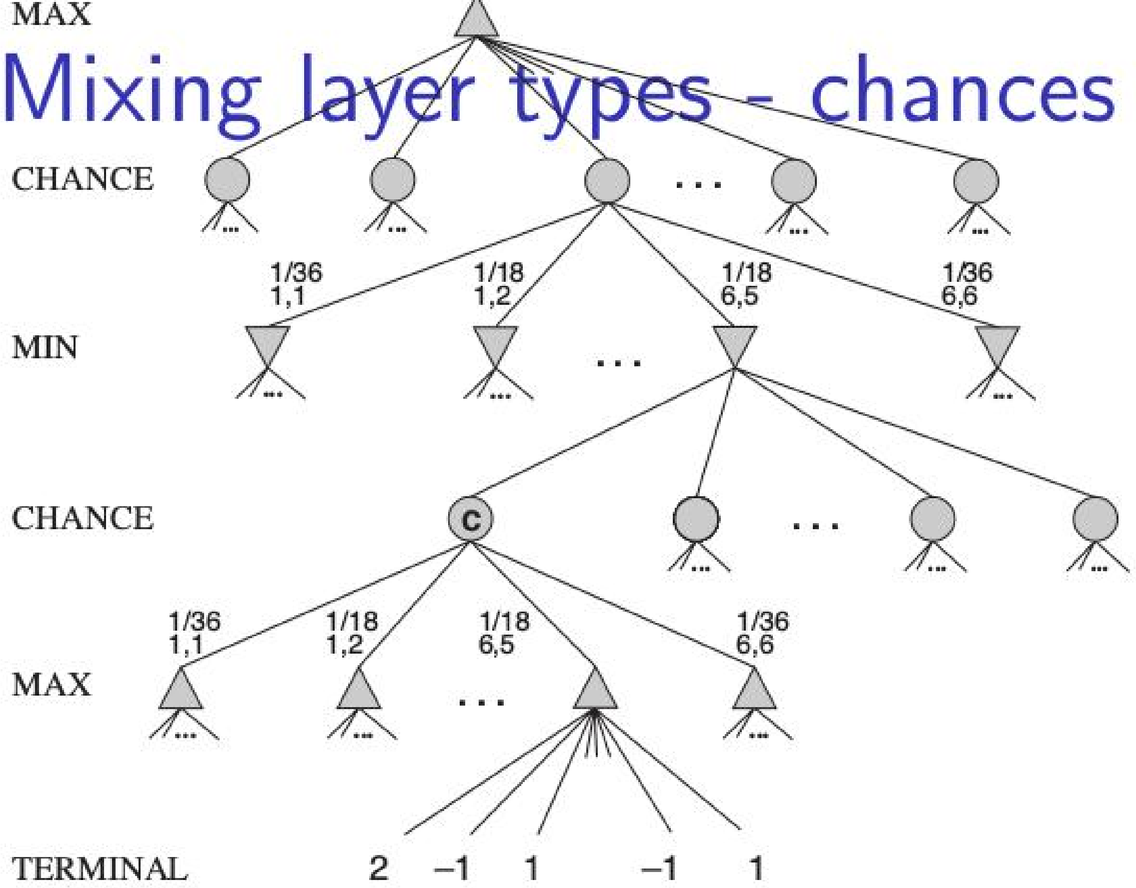


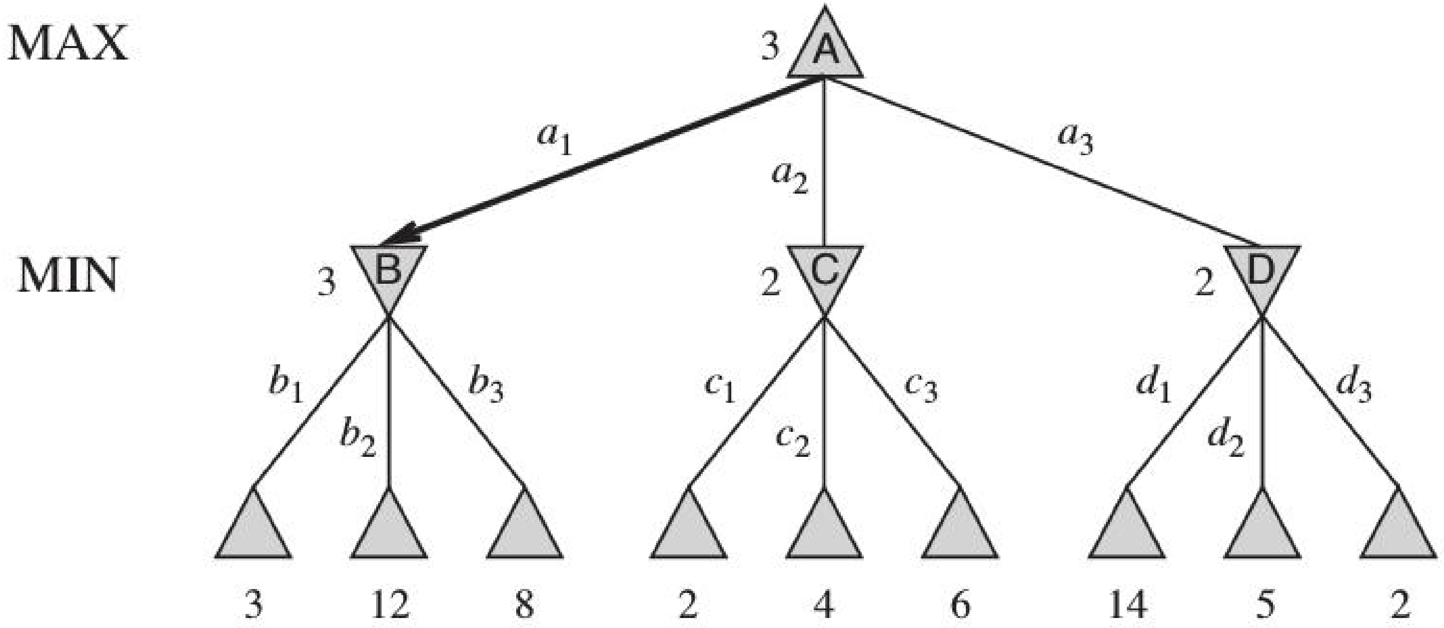


Extra random agent that moves after each MAX and MIN agent

$$\text{EXPECTIMINIMAX}(s) = \begin{cases} \text{UTILITY}(s) & \text{if} \quad \text{TERMINAL-TEST}(s) \\ \max_a \text{EXPECTIMINIMAX}(\text{RESULT}(s, a)) & \text{if} \quad \text{PLAYER}(s) = \text{MAX} \\ \min_a \text{EXPECTIMINIMAX}(\text{RESULT}(s, a)) & \text{if} \quad \text{PLAYER}(s) = \text{MIN} \\ \sum_r P(r) \text{EXPECTIMINIMAX}(\text{RESULT}(s, r)) & \text{if} \quad \text{PLAYER}(s) = \text{CHANCE} \end{cases}$$

Mixing layer types - chances inserted

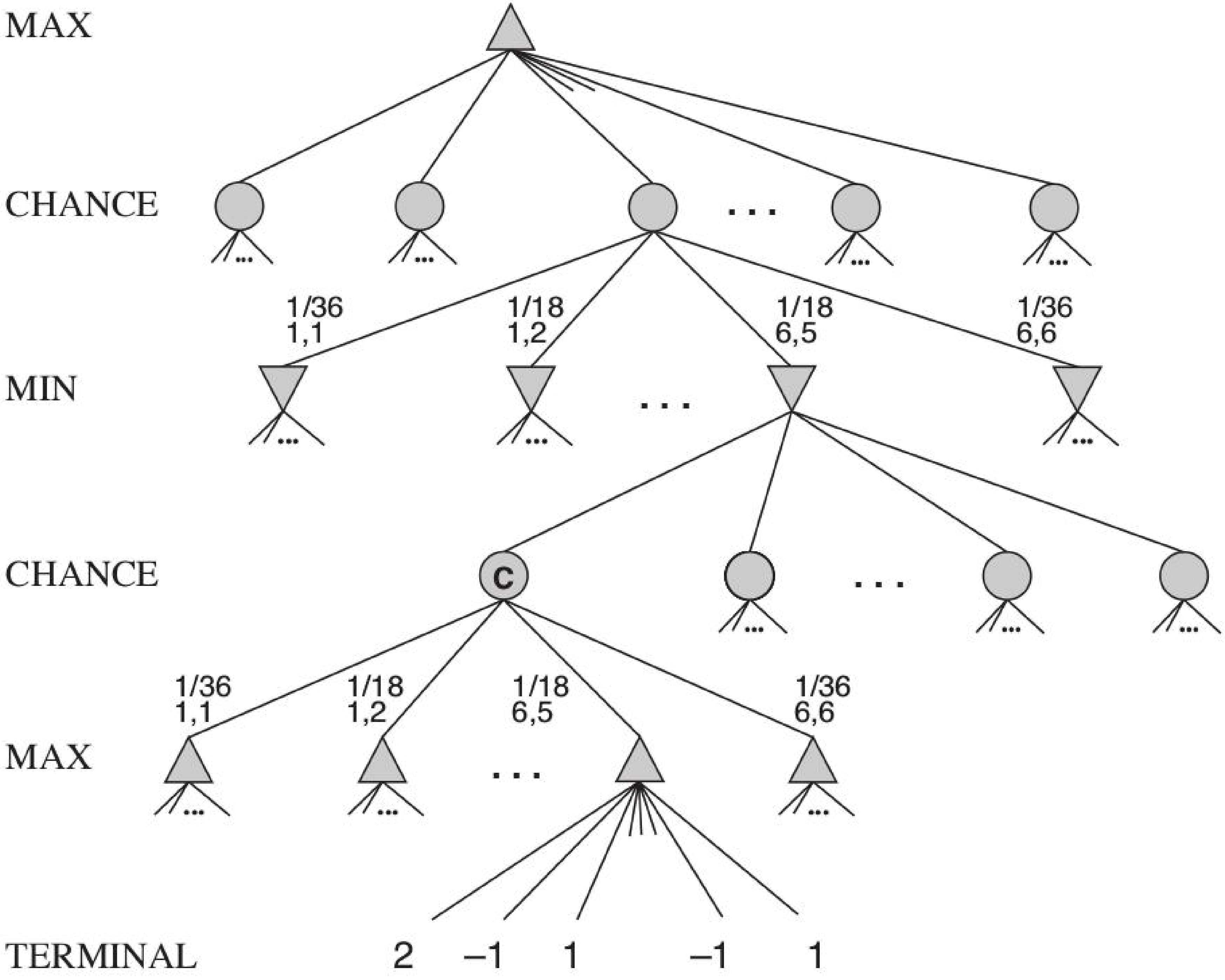




Extra random agent that moves after each MAX and MIN agent

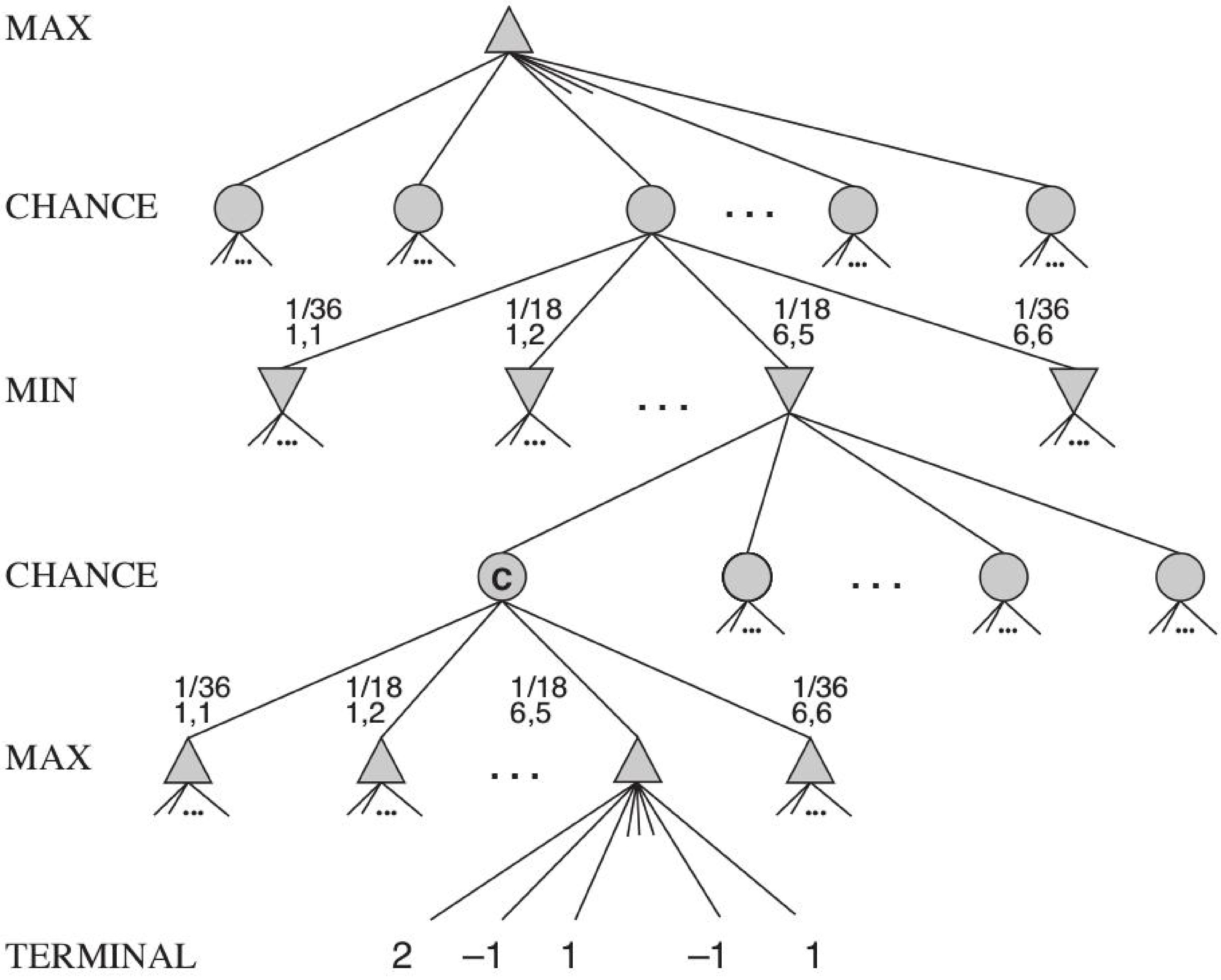
$$
\text{EXPECTIMINIMAX}(s) =
\begin{cases}
\text{UTILITY}(s) & \text{if TERMINAL-TEST}(s) \\
\max_a \text{EXPECTIMINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MAX} \\
\min_a \text{EXPECTIMINIMAX}(\text{RESULT}(s,a)) & \text{if PLAYER}(s) = \text{MIN} \\
\sum_r P(r)\,\text{EXPECTIMINIMAX}(\text{RESULT}(s,r)) & \text{if PLAYER}(s) = \text{CHANCE}
\end{cases}
$$

Mixing chances into min/max tree, how big?



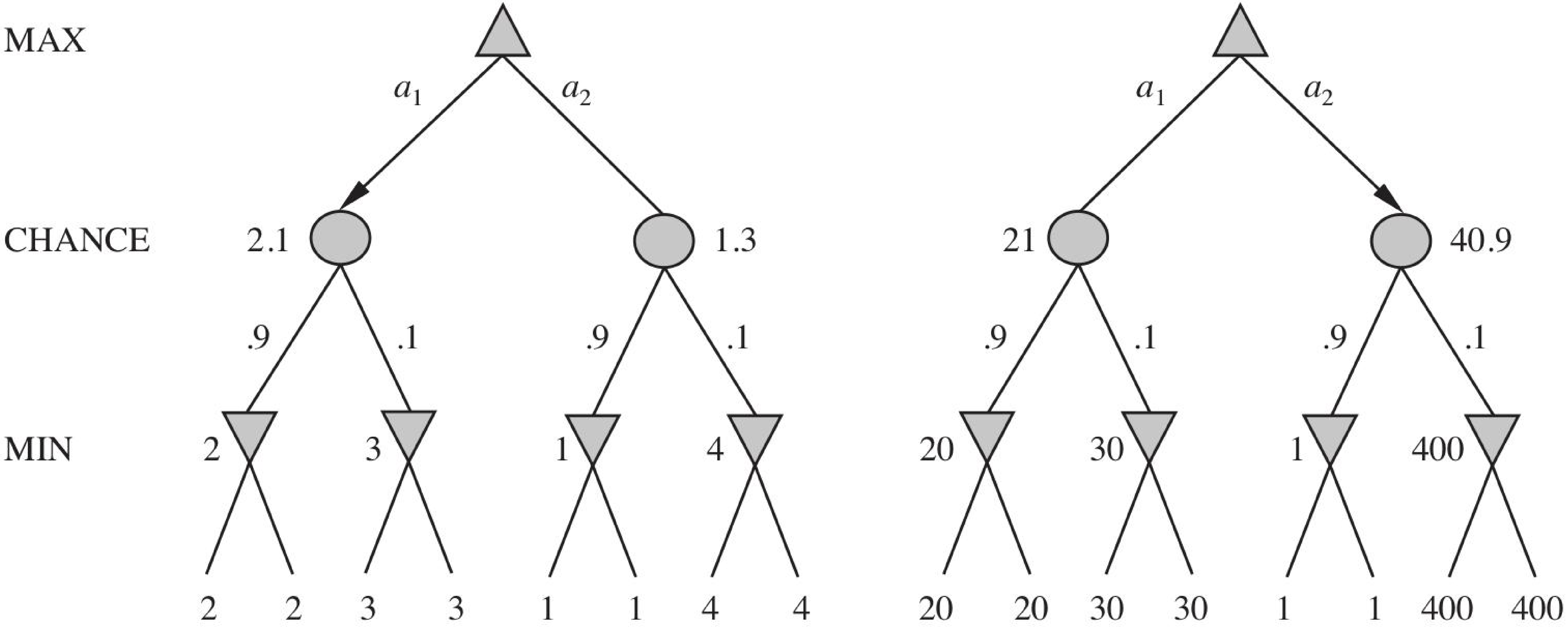
- b branching factor
- m maximum depth
- n number of distinct rolls

Mixing chances into min/max tree, how big?



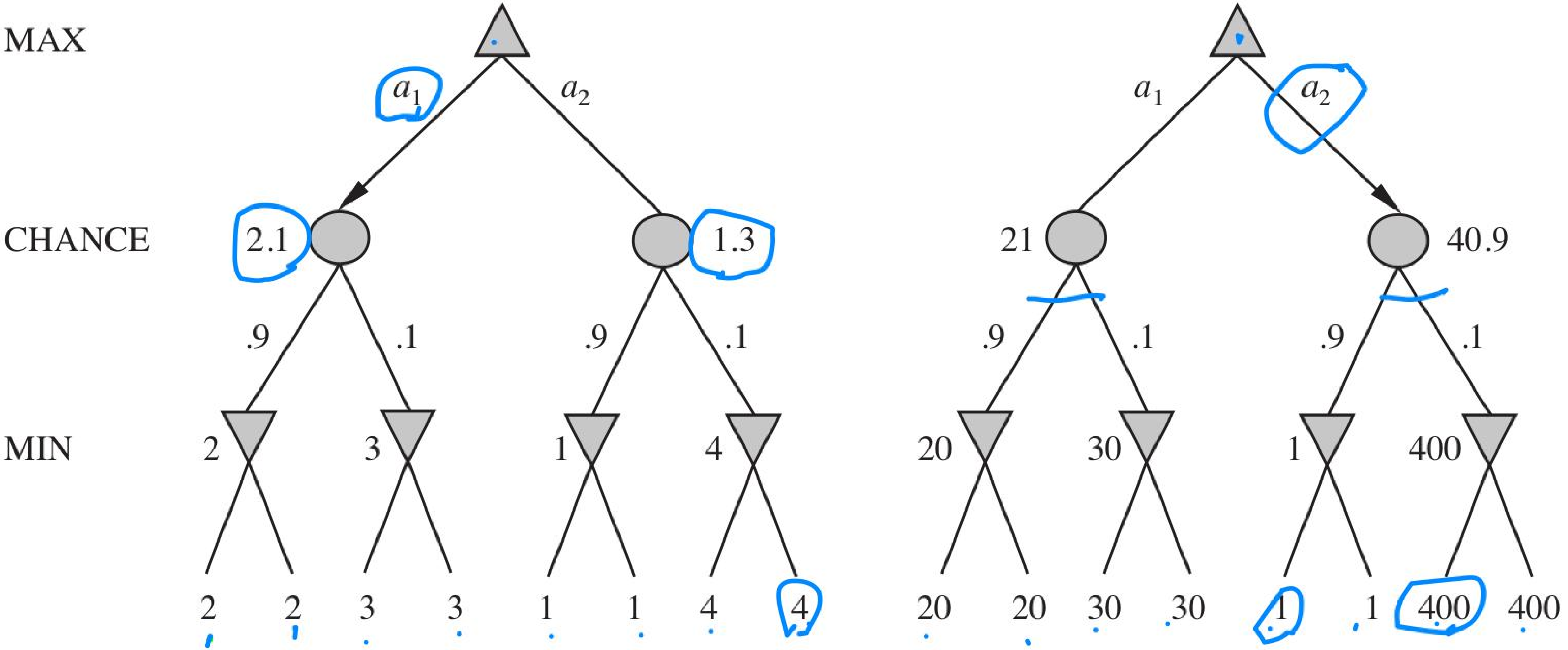
- b branching factor ~20
- m maximum depth
- n number of distinct rolls 21

Evaluation function



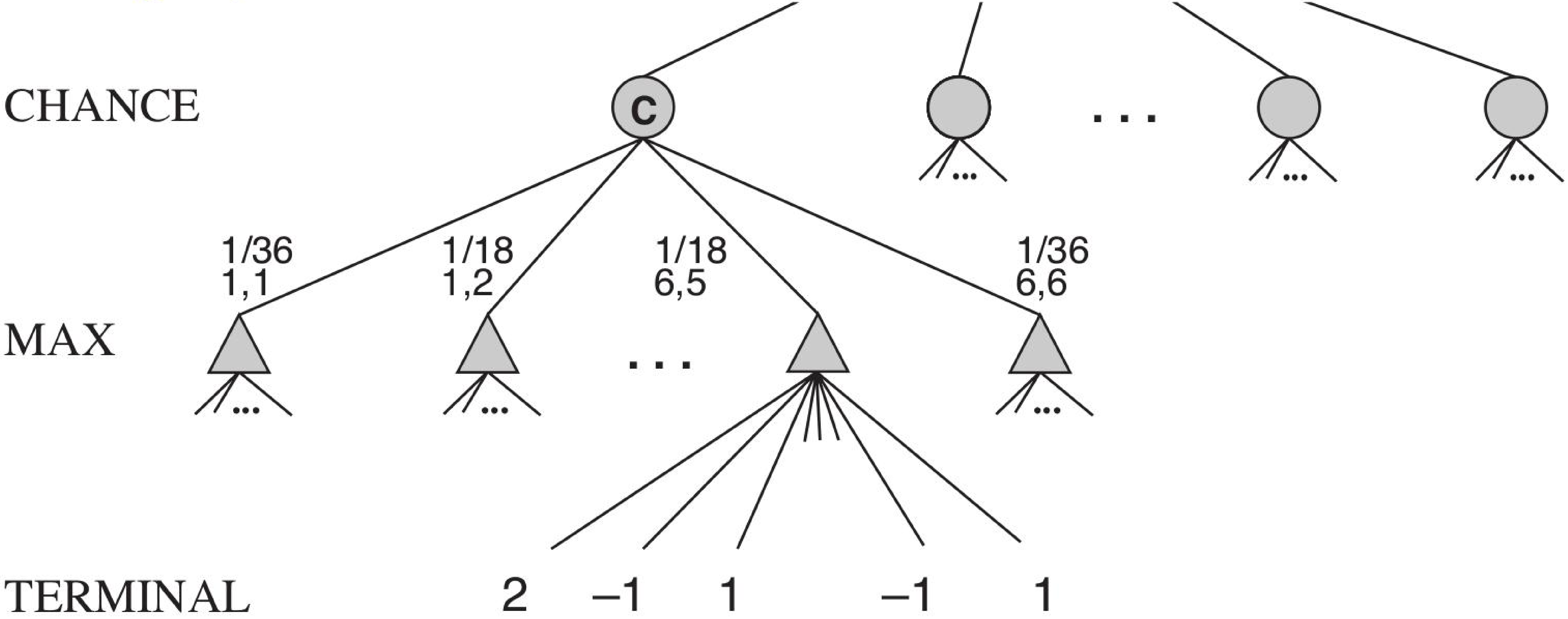
- Left: a_1 is the best. Right: a_2 is the best. Ordering of the (terminal) leaves is the same.
- Scale matters! Not only ordering.
- Can we prune the tree? (α, β like?)

Evaluation function



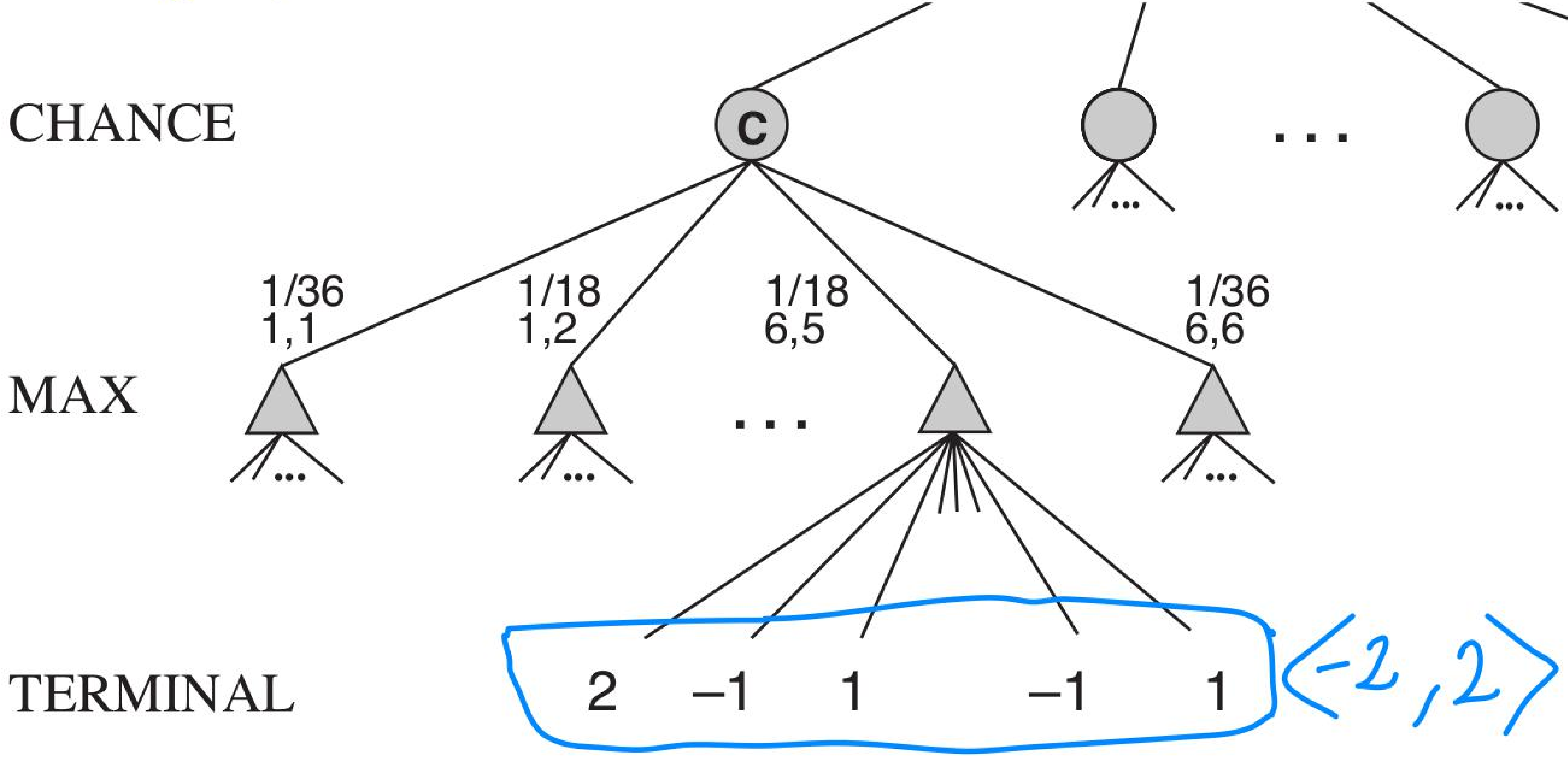
- Left: a_1 is the best. Right: a_2 is the best. Ordering of the (terminal) leaves is the same.
- Scale matters! Not only ordering.
- Can we prune the tree? (α, β like?)

Pruning expectiminimax tree



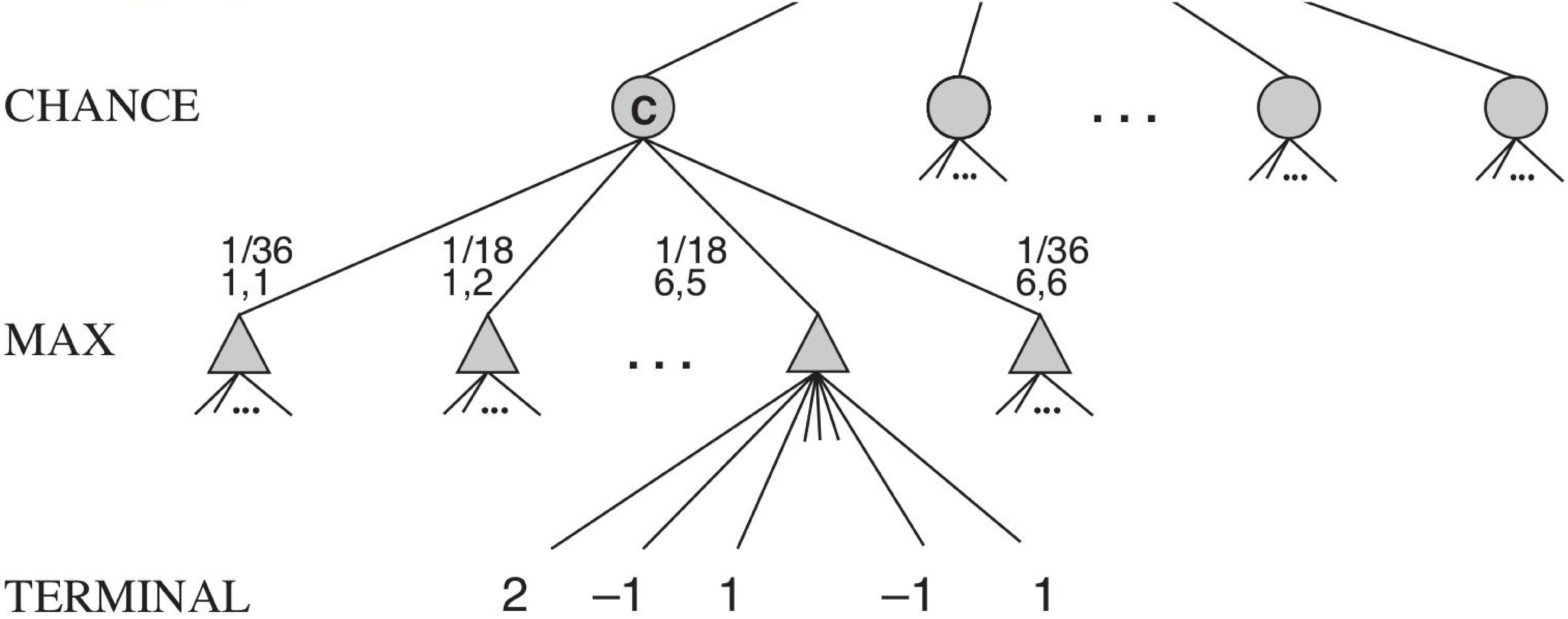
- Bounds on terminal utilities needed. Terminal values from -2 to 2.
- Monte Carlo simulation for evaluation a position (state).

Pruning expectiminimax tree



- Bounds on terminal utilities needed. Terminal values from −2 to 2.
- Monte Carlo simulation for evaluation a position (state).

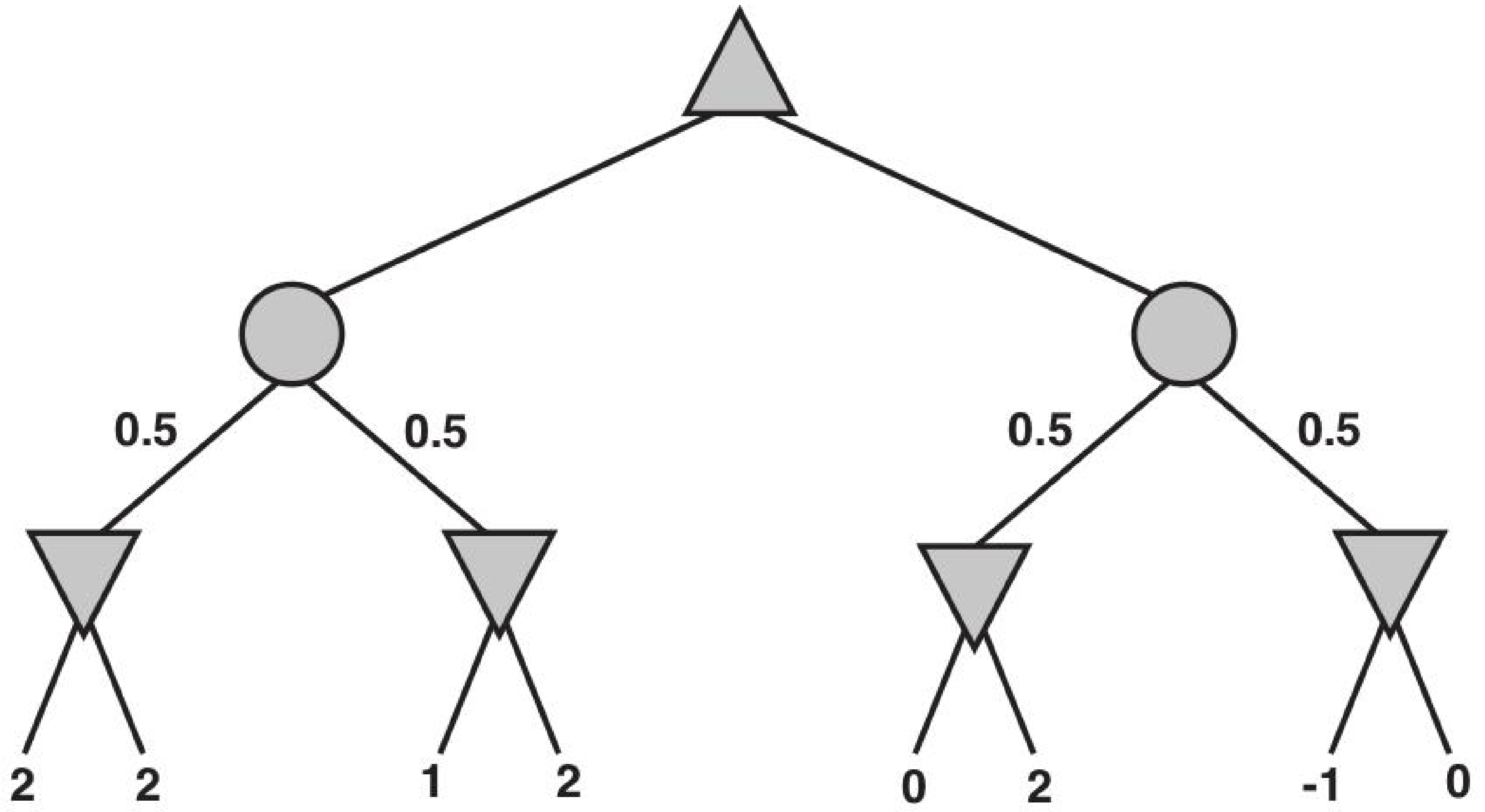
Pruning expectiminimax tree



- Bounds on terminal utilities needed. Terminal values from -2 to 2.
- Monte Carlo simulation for evaluation a position (state).

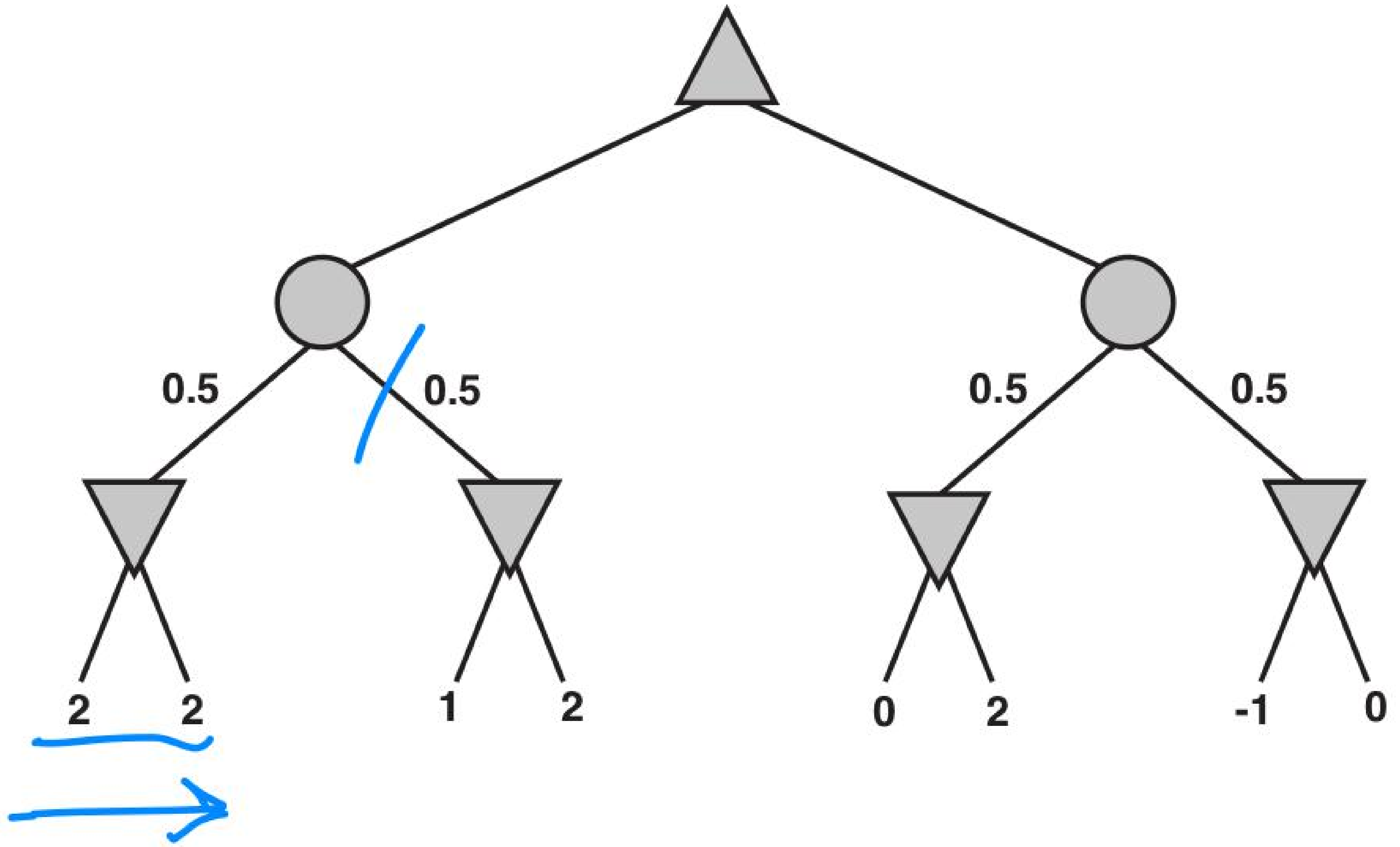
Where to prune the Expectimax tree

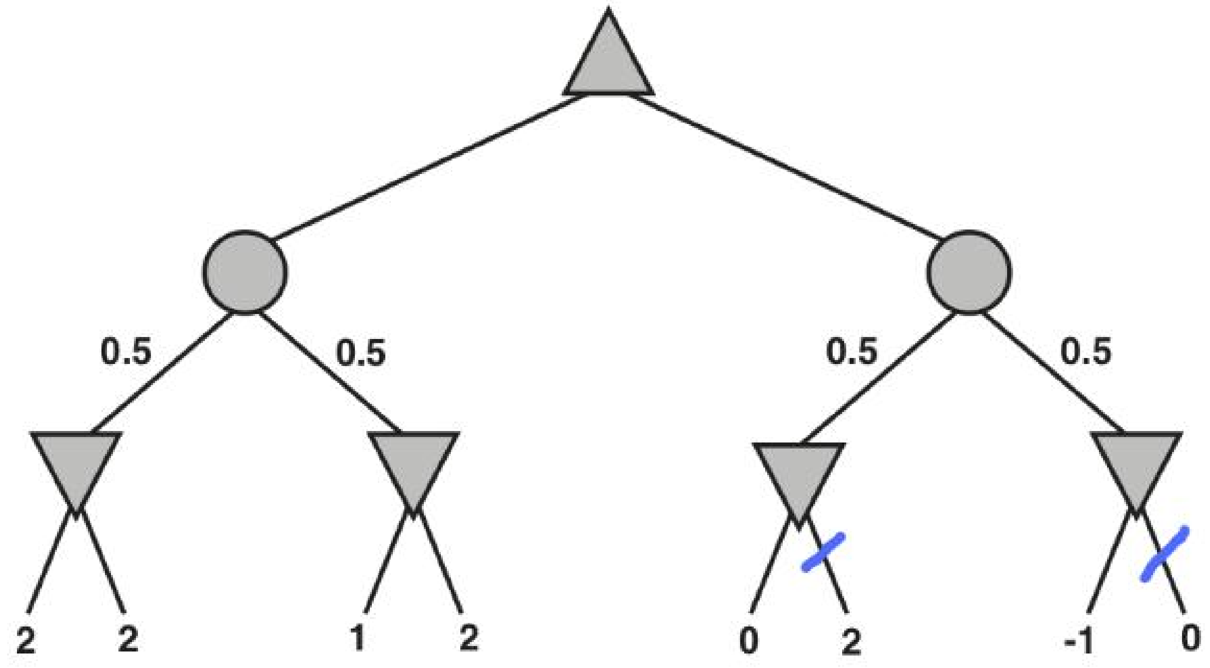
- Assume terminal nodes bounded to -2 to 2, inclusive
- Going from left to right.
- Which branches can be pruned out?



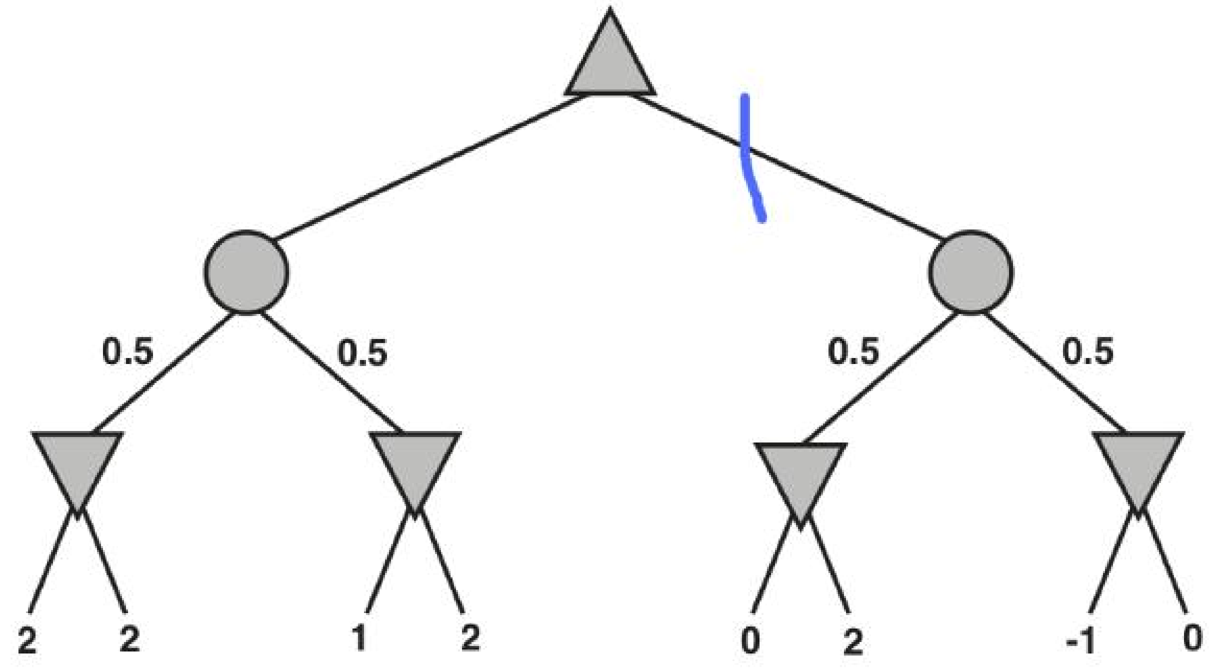
Where to prune the Expectimax tree

- Assume terminal nodes bounded to -2 to 2, inclusive
- Going from left to right.
- Which branches can be pruned out?

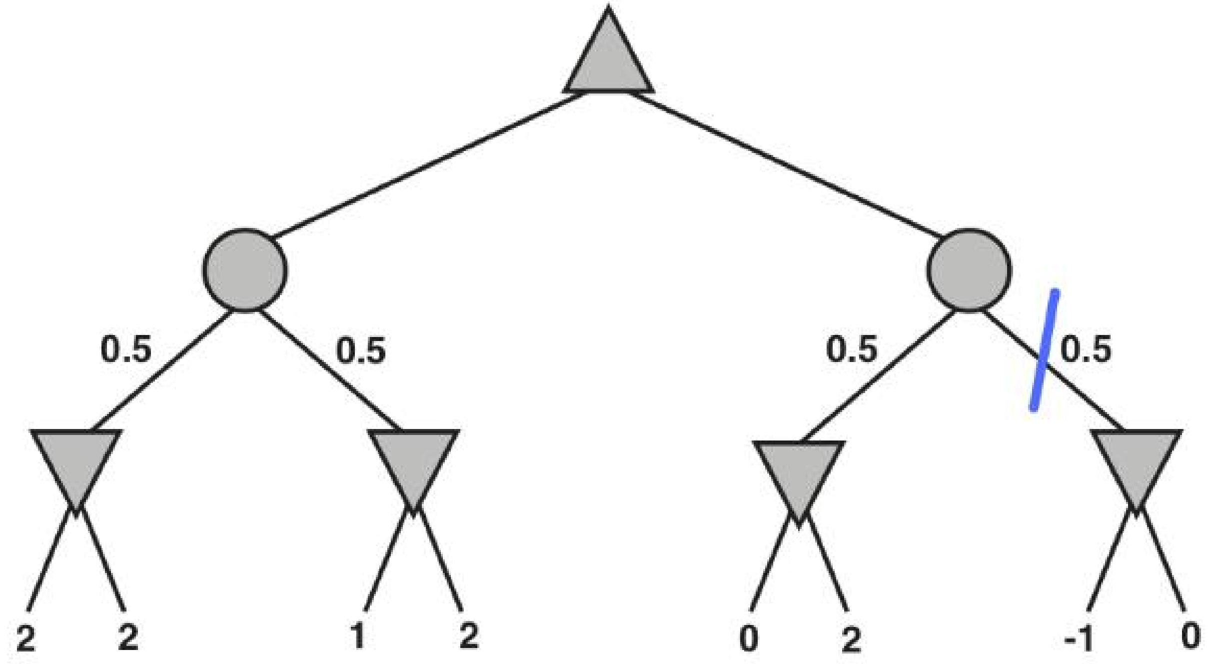



0.5
0.5
0.5
0.5
2
2
1
2
0
2
-1
0

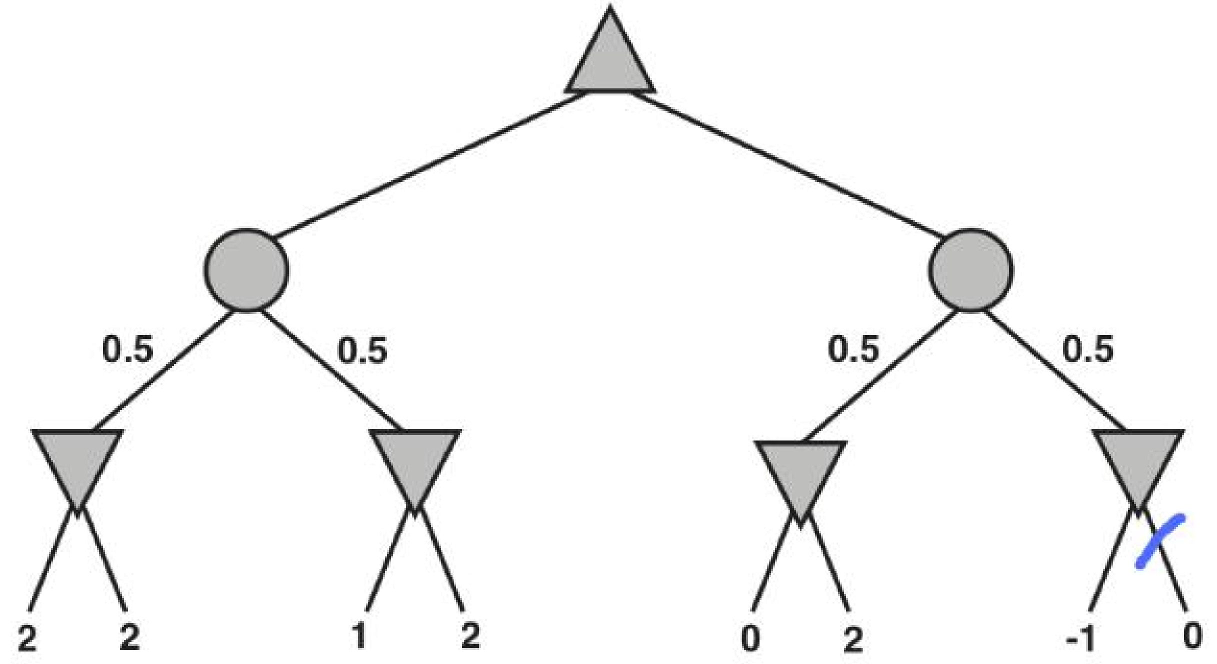
A


0.5
0.5
0.5
0.5
2
2
1
2
0
2
-1
0

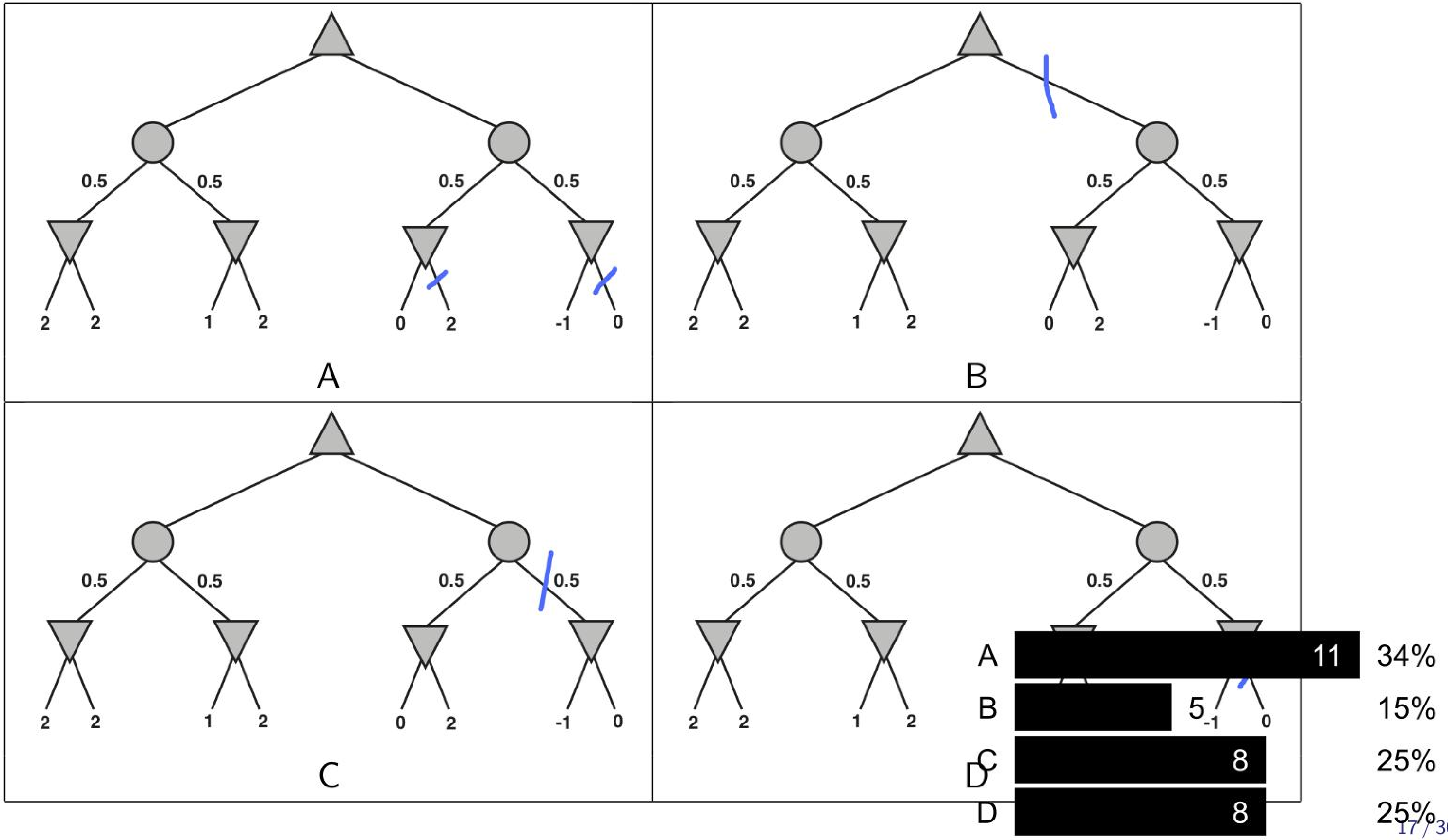
B

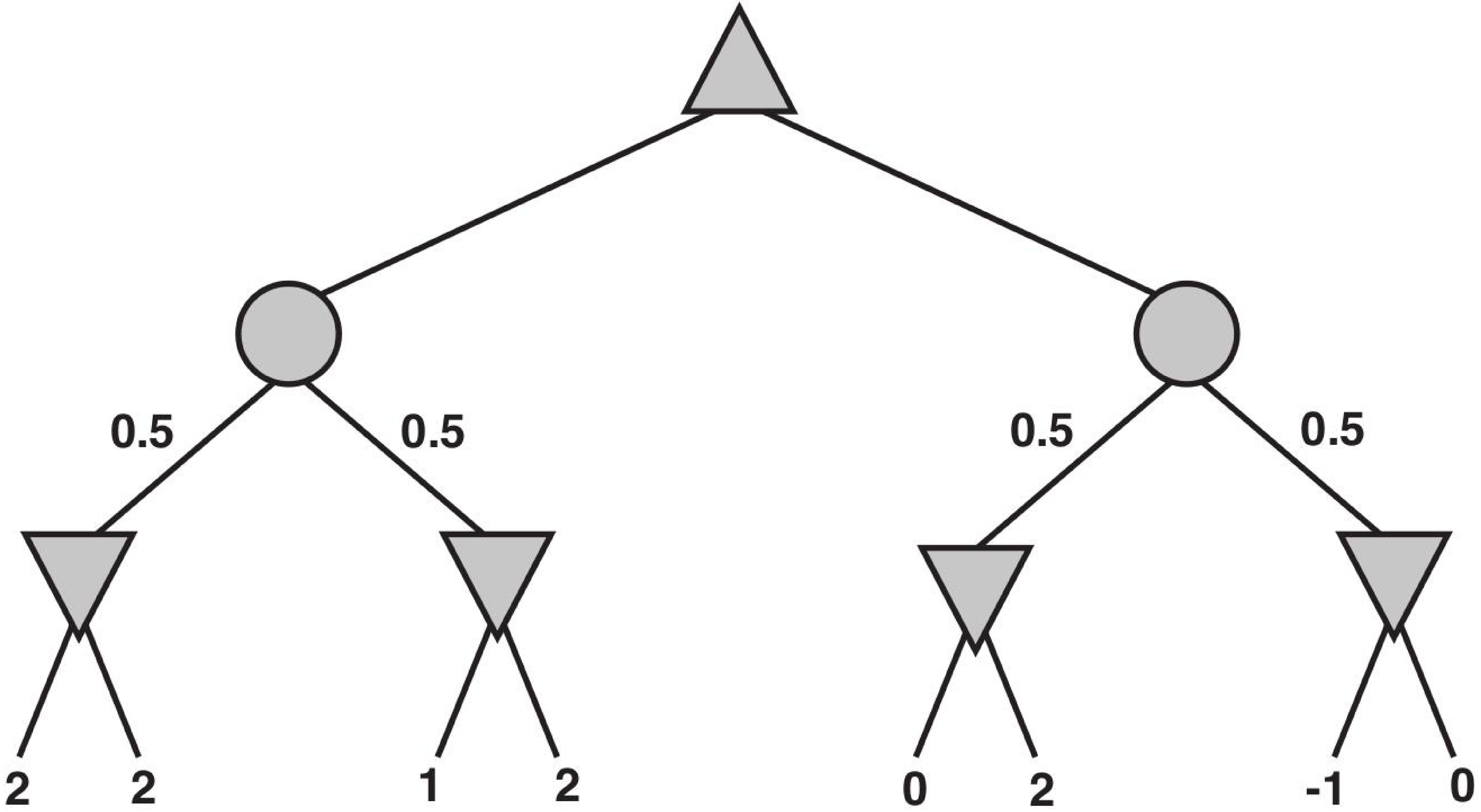

0.5
0.5
0.5
0.5
2
2
1
2
0
2
-1
0

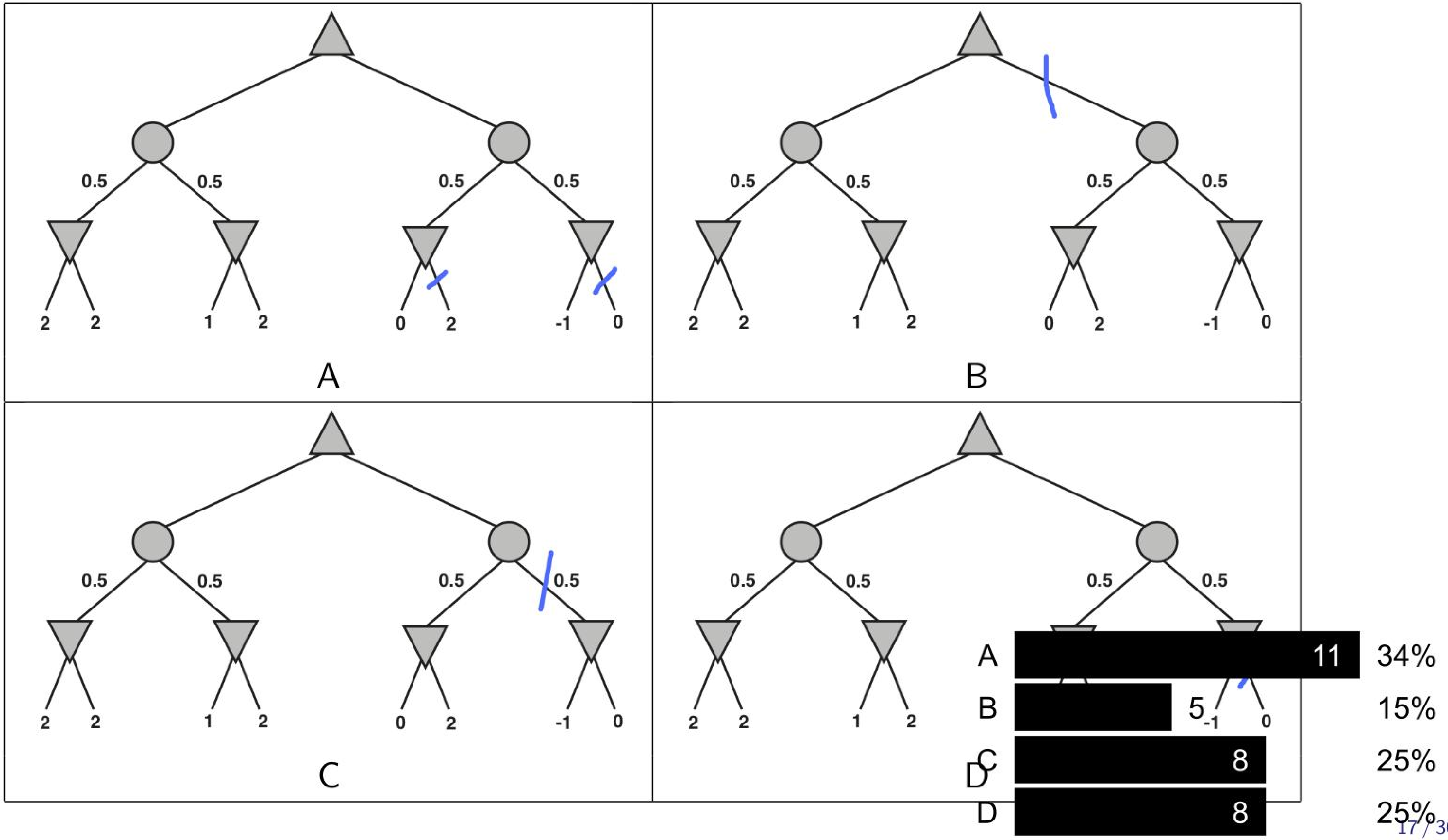
C

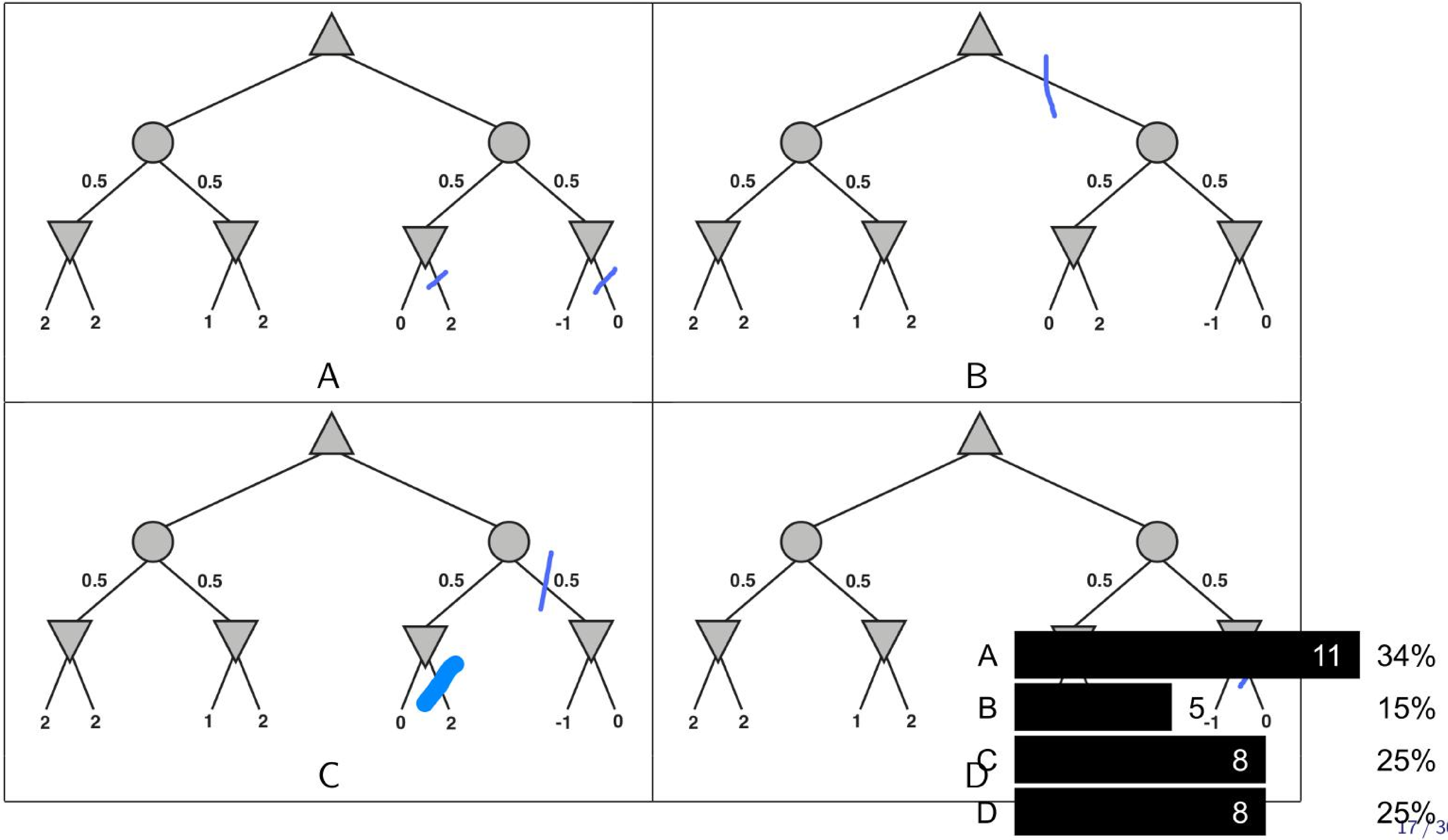

0.5
0.5
0.5
0.5
2
2
1
2
0
2
-1
0

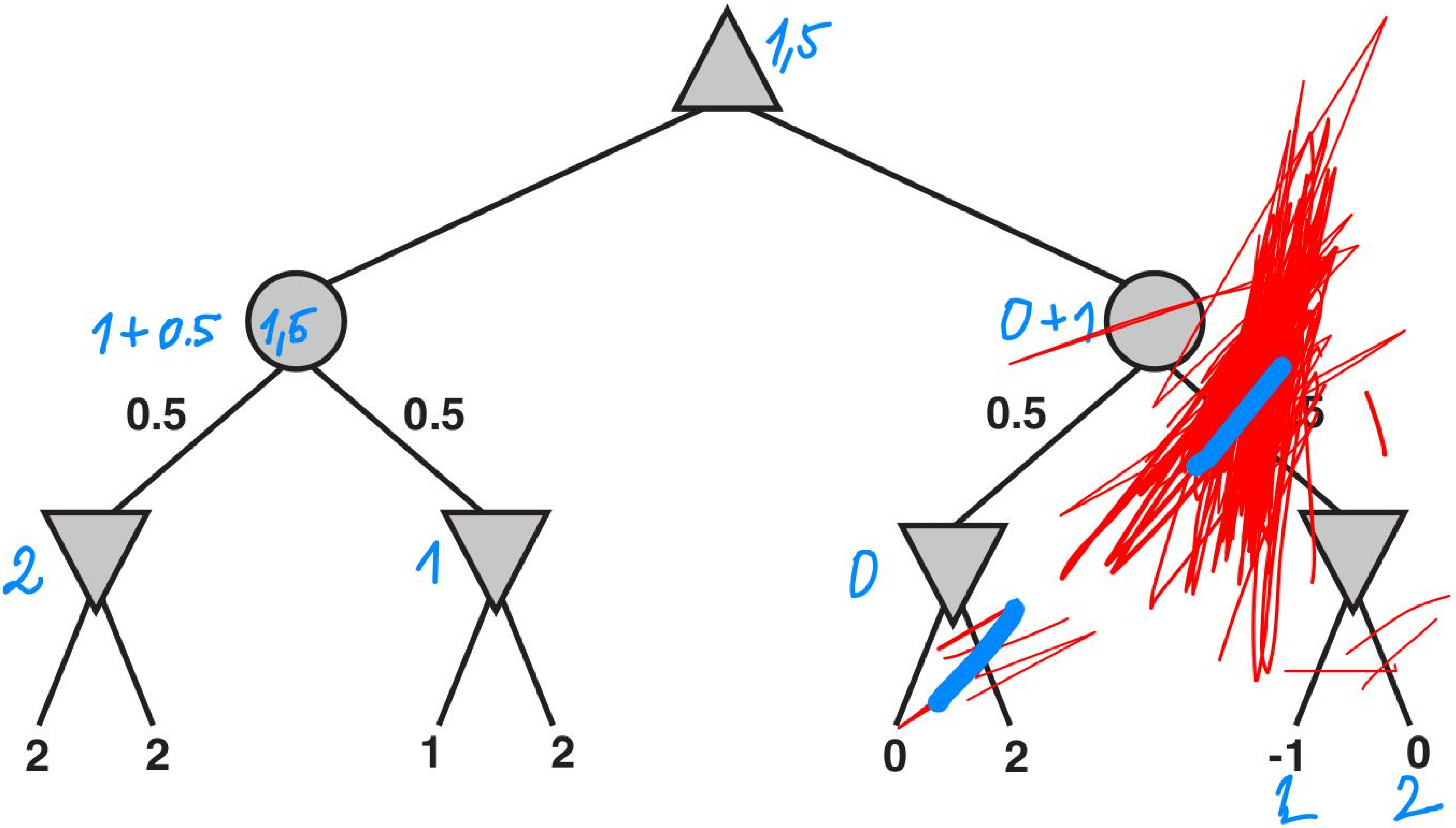
D

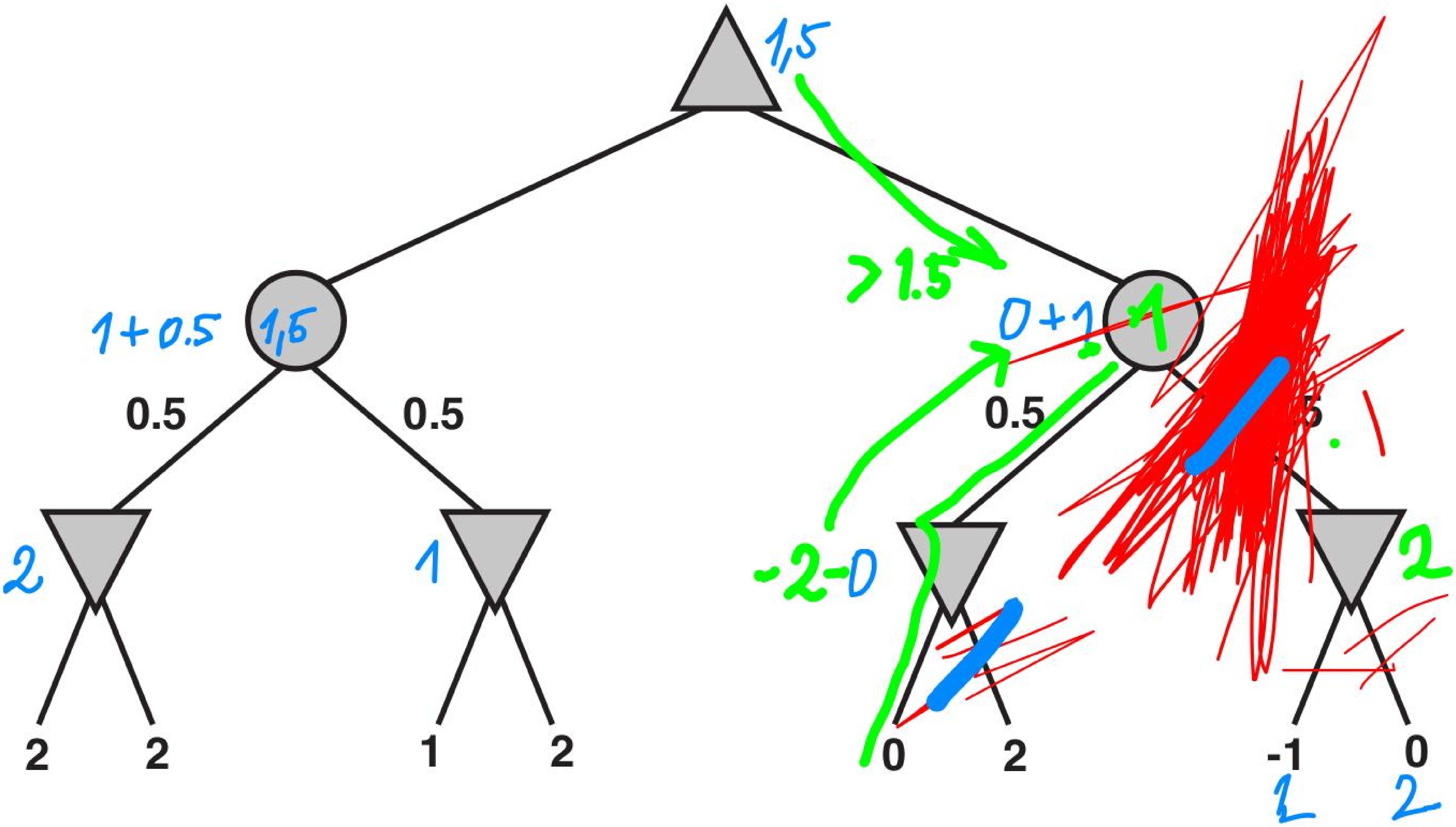

0.5
A
B
C
D
A 11 34%
B 5 15%
C 8 25%
D 8 25%

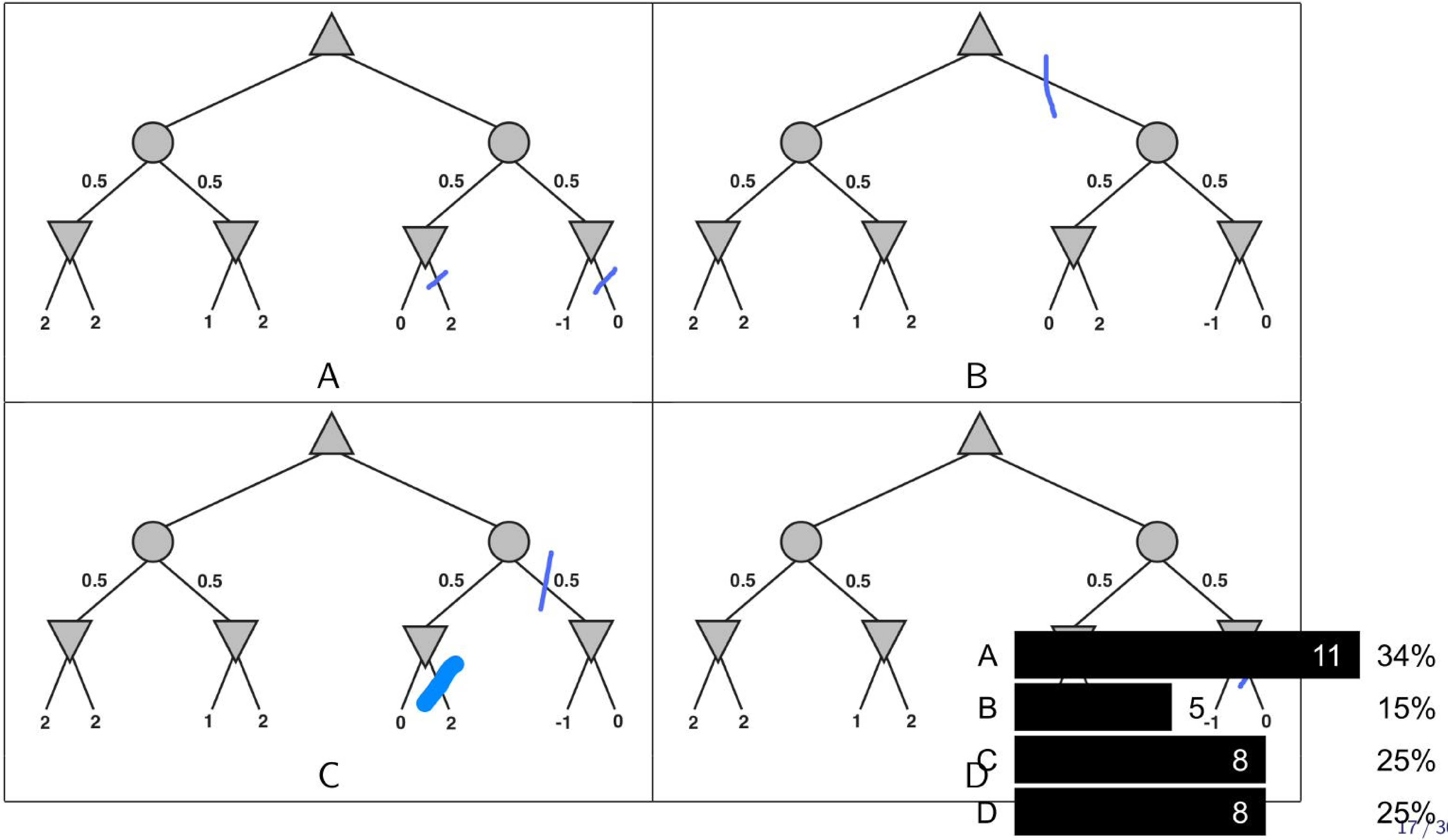

0.5
0.5
0.5
0.5
2
2
1
2
0
2
-1
0

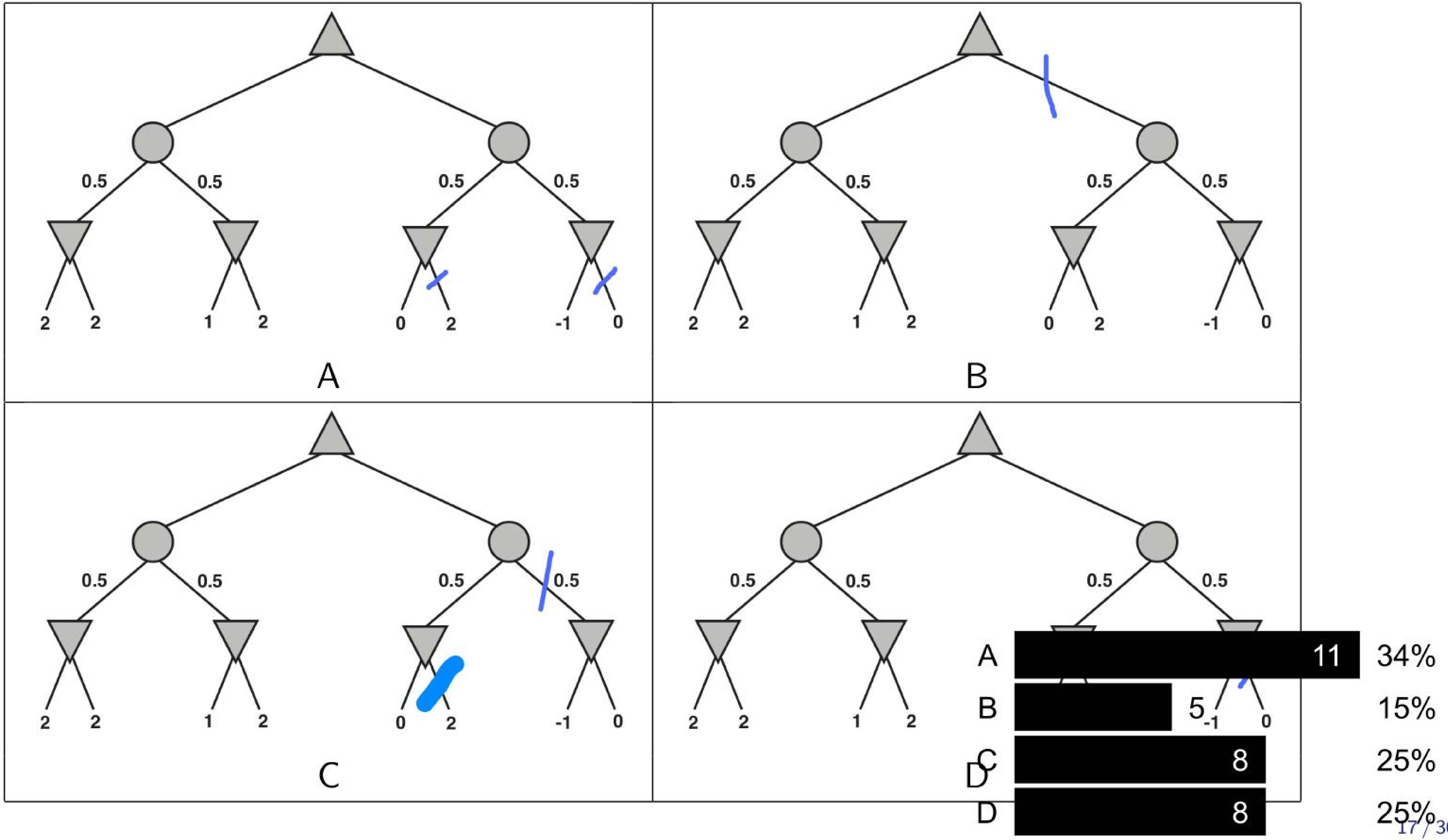

A
B
C
D
0.5
2 2 1 2 0 2 -1 0
A 11 34%
B 5 15%
C 8 25%
D 8 25%


0.5 0.5 0.5 0.5
2 2 1 2 0 2 -1 0
A
B
C
D
A 11 34%
B 5 15%
C 8 25%
D 8 25%


1,5
1+0.5
1,5
0+1
0.5
0.5
0.5
2
1
0
2
2
1
2
0
2
-1
0
1
2

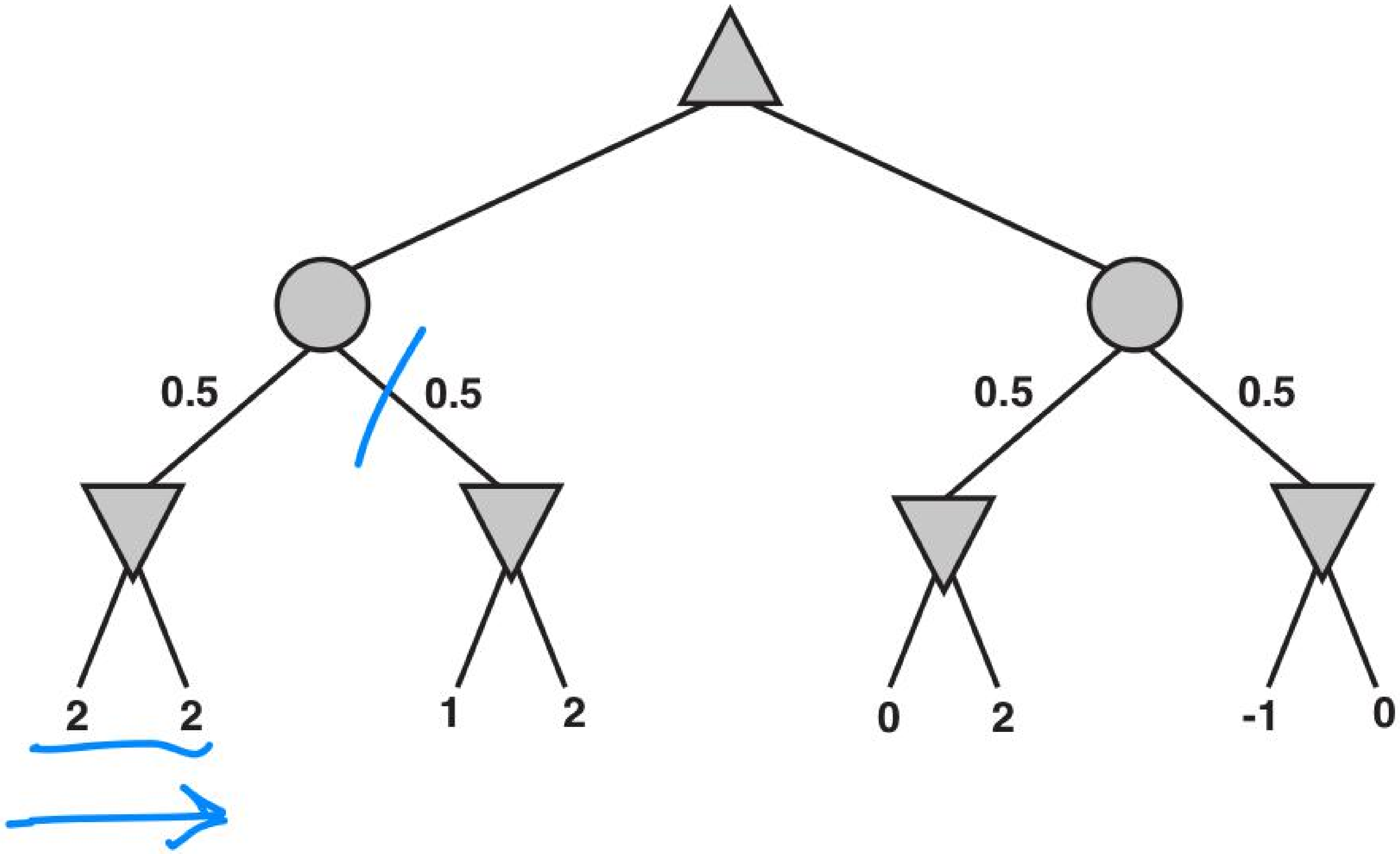

0.5
0.5
0.5
2
2
1
2
0
2
-1
0


0.5
2 2 1 2 0 2 -1 0
A
B
C
D
A 11 34%
B 5 15%
C 8 25%
D 8 25%


0.5 0.5 0.5 0.5
2 2 1 2 0 2 -1 0
A
0.5 0.5 0.5 0.5
2 2 1 2 0 2 -1 0
B
0.5 0.5 0.5 0.5
2 2 1 2 0 2 -1 0
C
0.5 0.5 0.5 0.5
2 2 1 2 -1 0
D
A 11 34%
B 5 15%
C 8 25%
D 8 25%

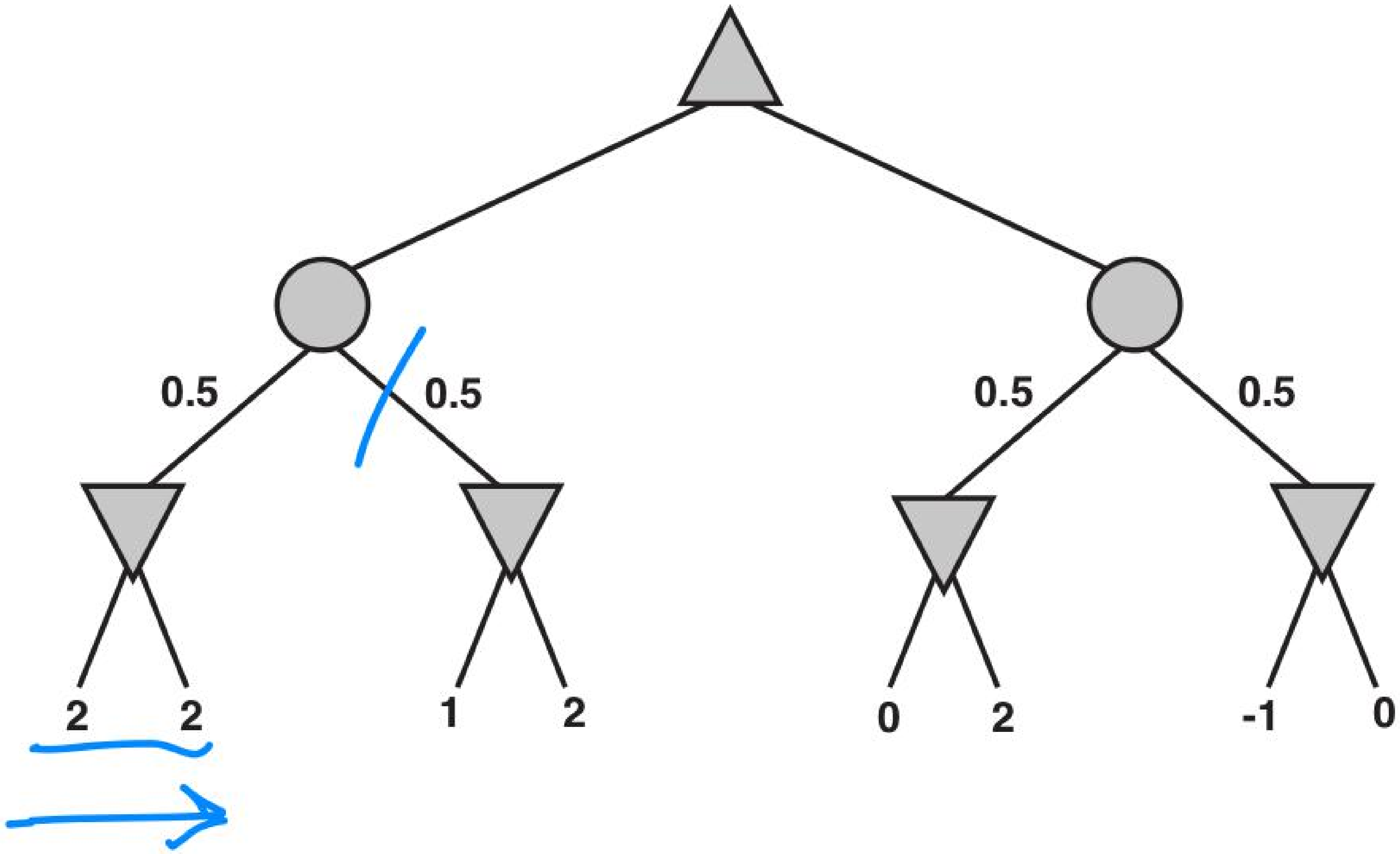
Where to prune the Expectimax tree

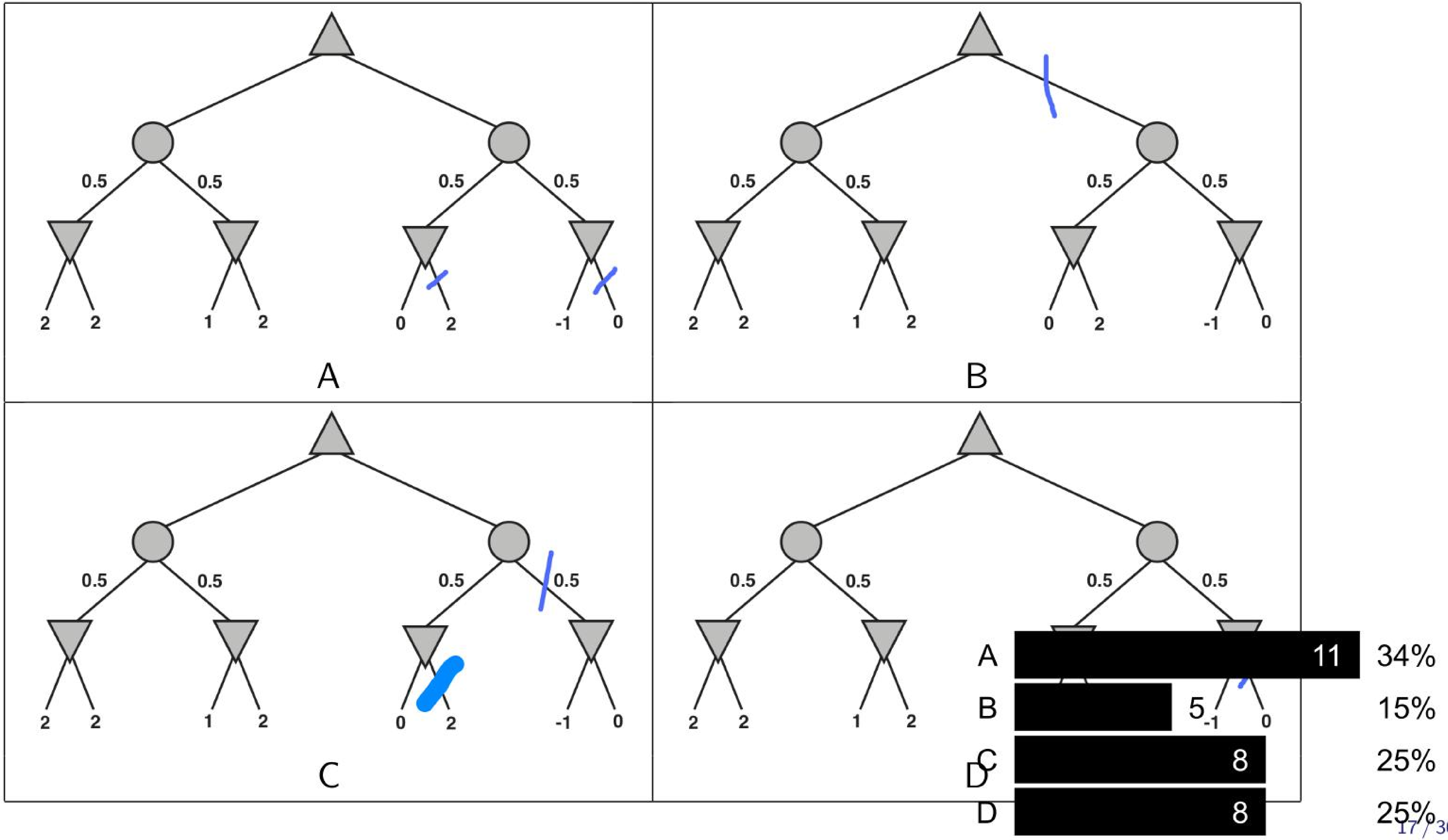
- Assume terminal nodes bounded to −2 to 2, inclusive
- Going from left to right.
- Which branches can be pruned out?



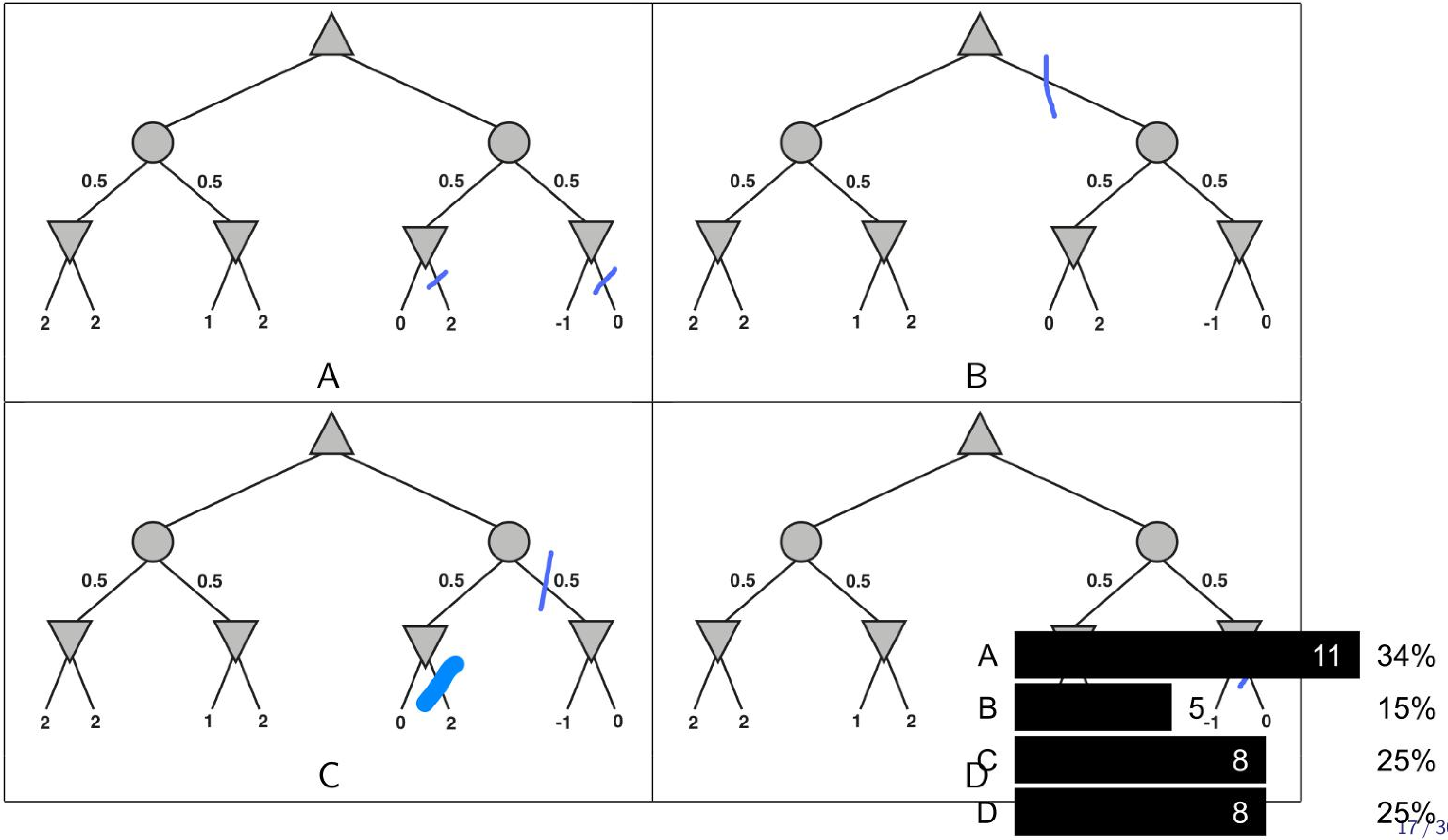
Where to prune the Expectimax tree

- Assume terminal nodes bounded to -2 to 2, inclusive
- Going from left to right.
- Which branches can be pruned out?

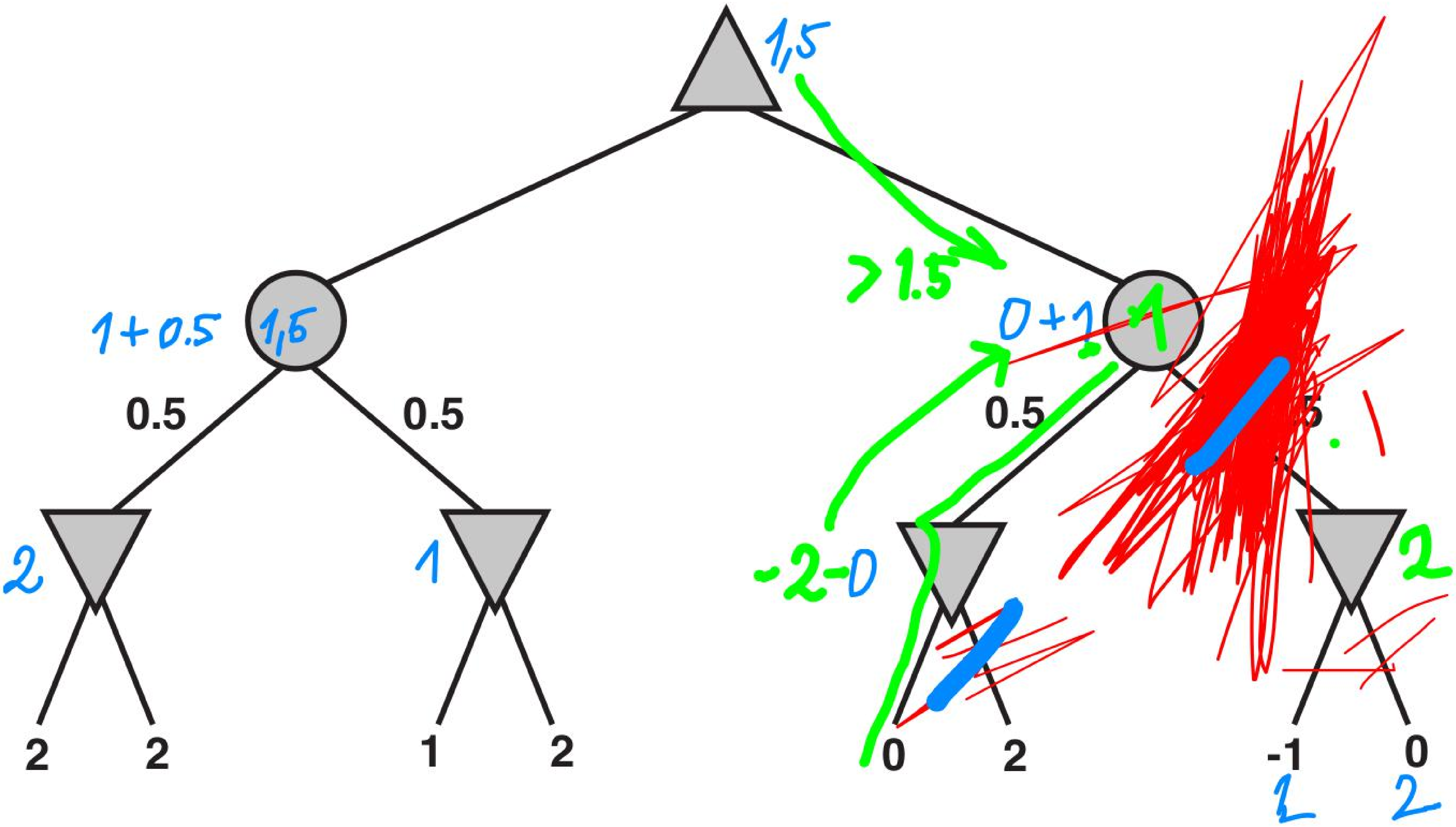


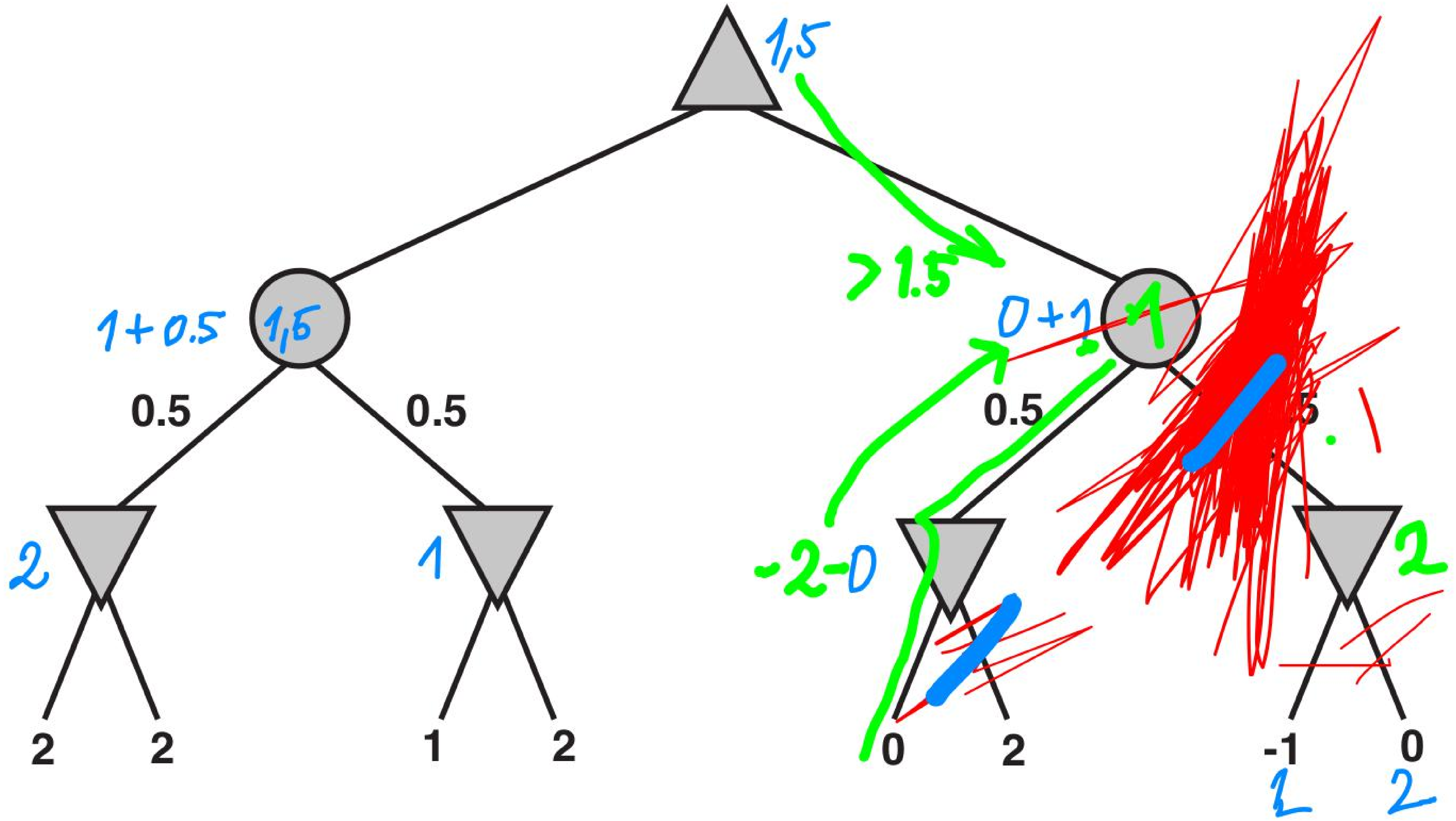


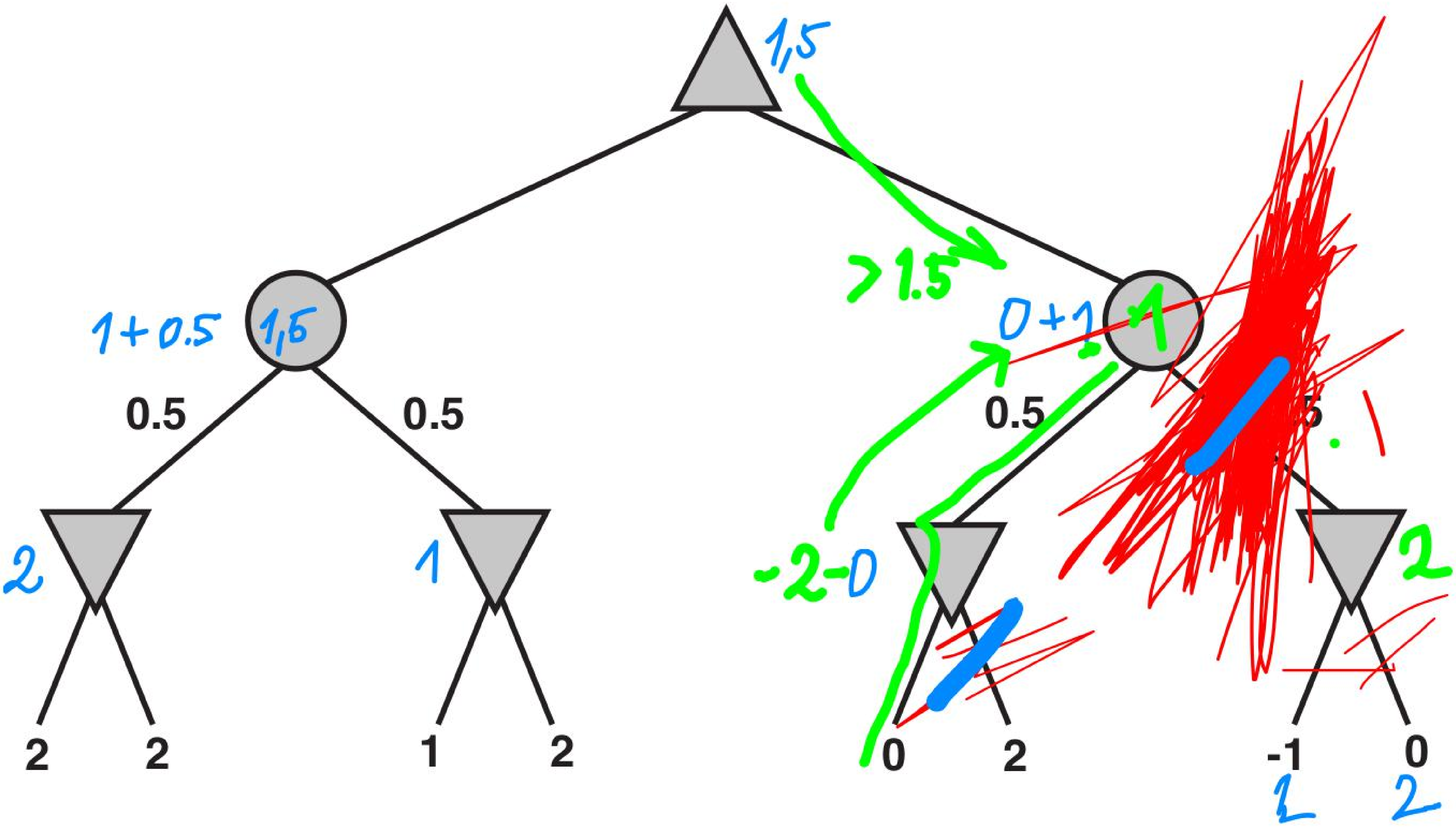
0.5
2 2 1 2 0 2 -1 0
A
B
C
D
A 11 34%
B 5 15%
C 8 25%
D 8 25%

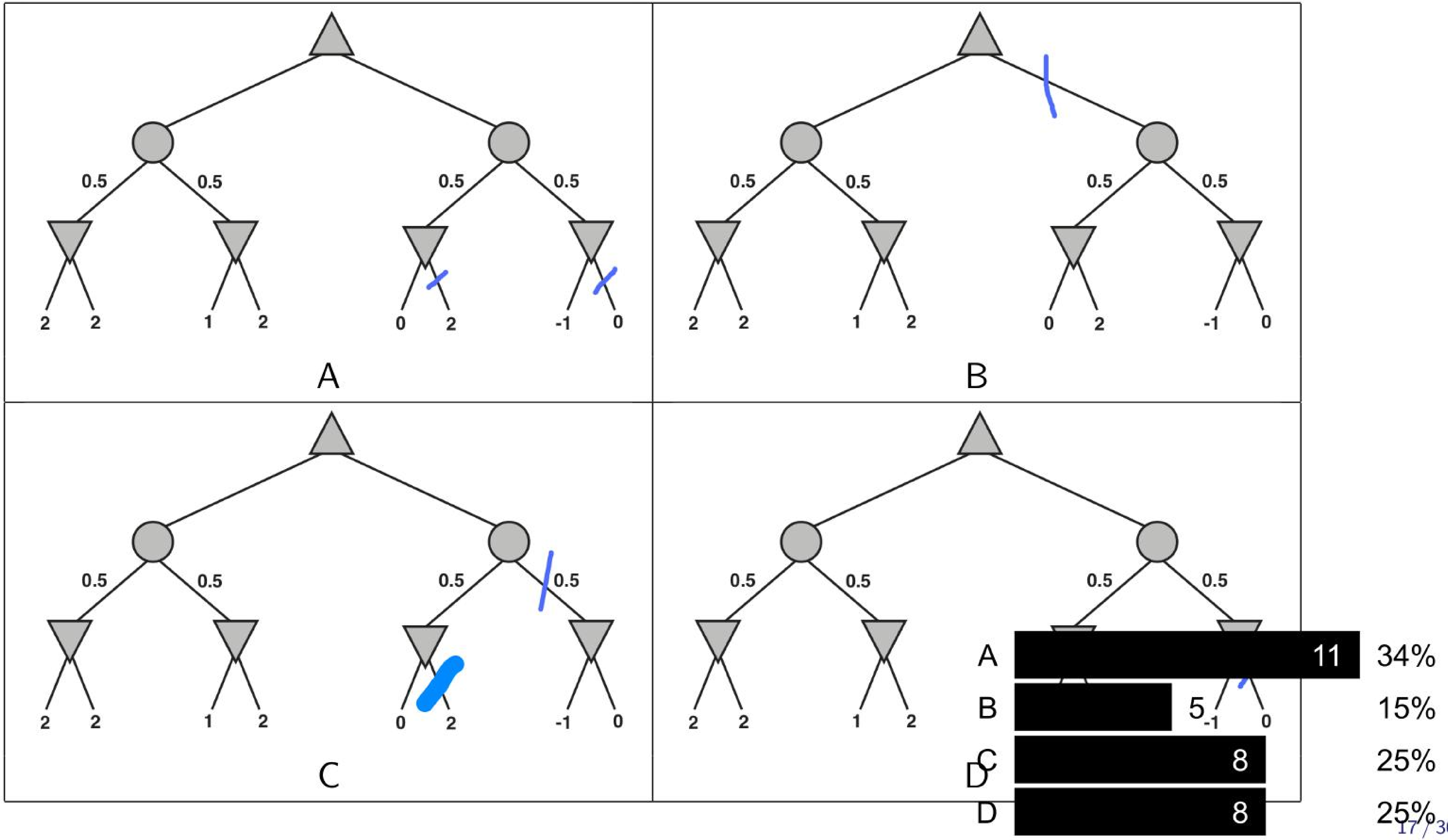


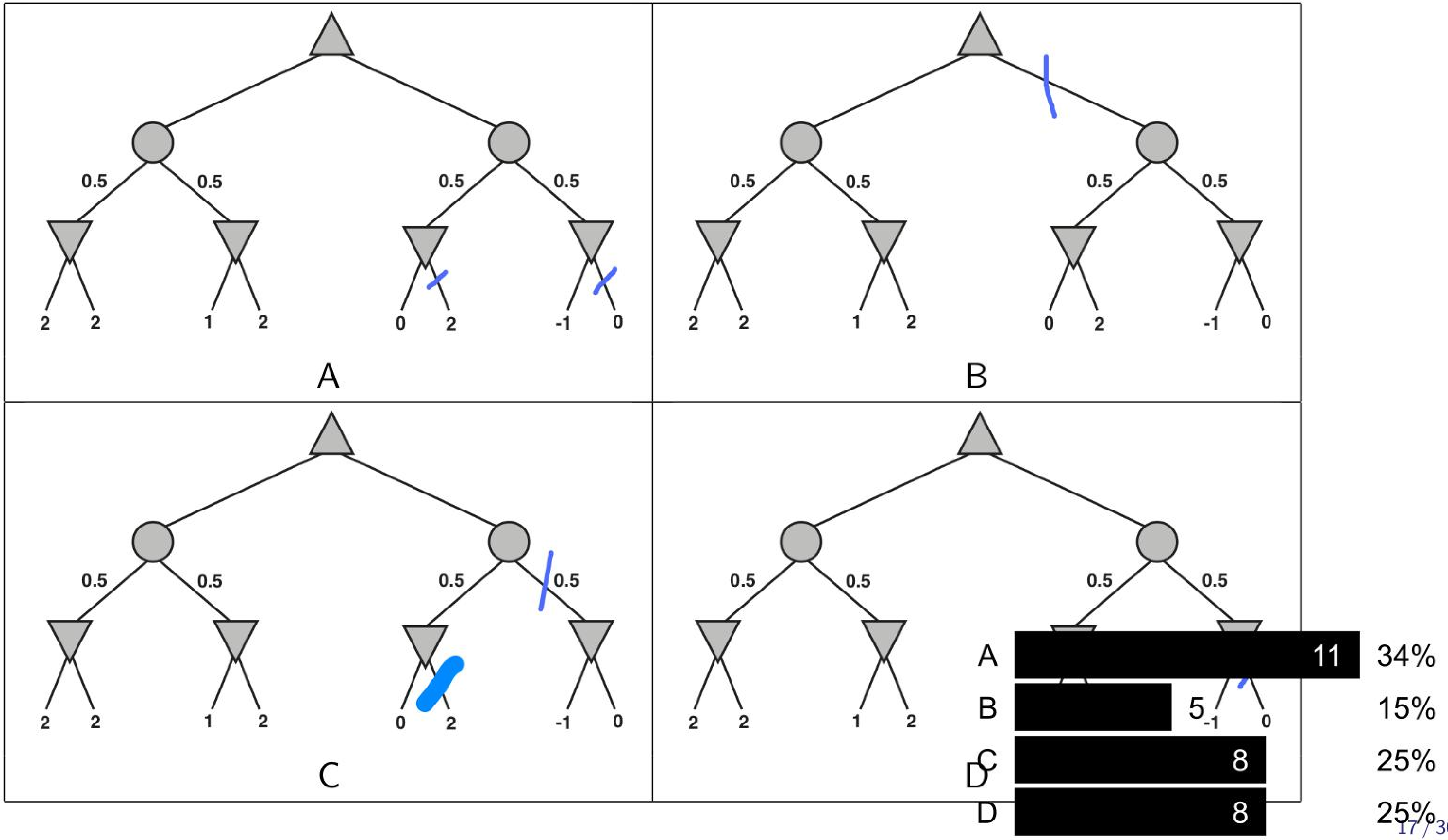
0.5 0.5 0.5 0.5
2 2 1 2 0 2 -1 0
A
B
C
D
A 11 34%
B 5 15%
C 8 25%
D 8 25%

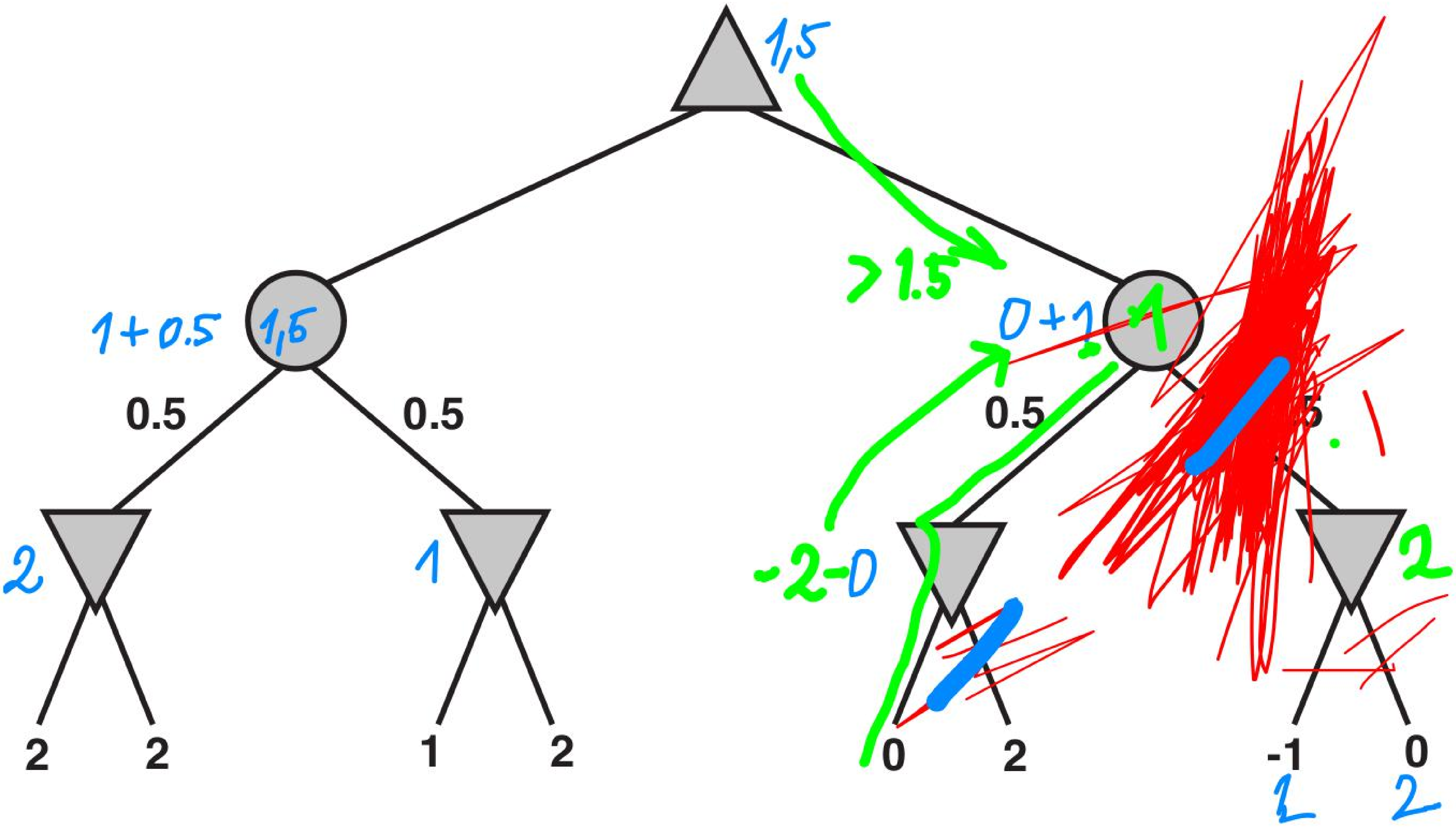

0.5
0.5
0.5
2
2
1
2
0
2
-1
0


1,5
>1.5
1+0.5
1,5
0+1
1
0.5
0.5
0.5
2
1
-2-0
2
2
2
1
2
0
2
-1
0
1
2

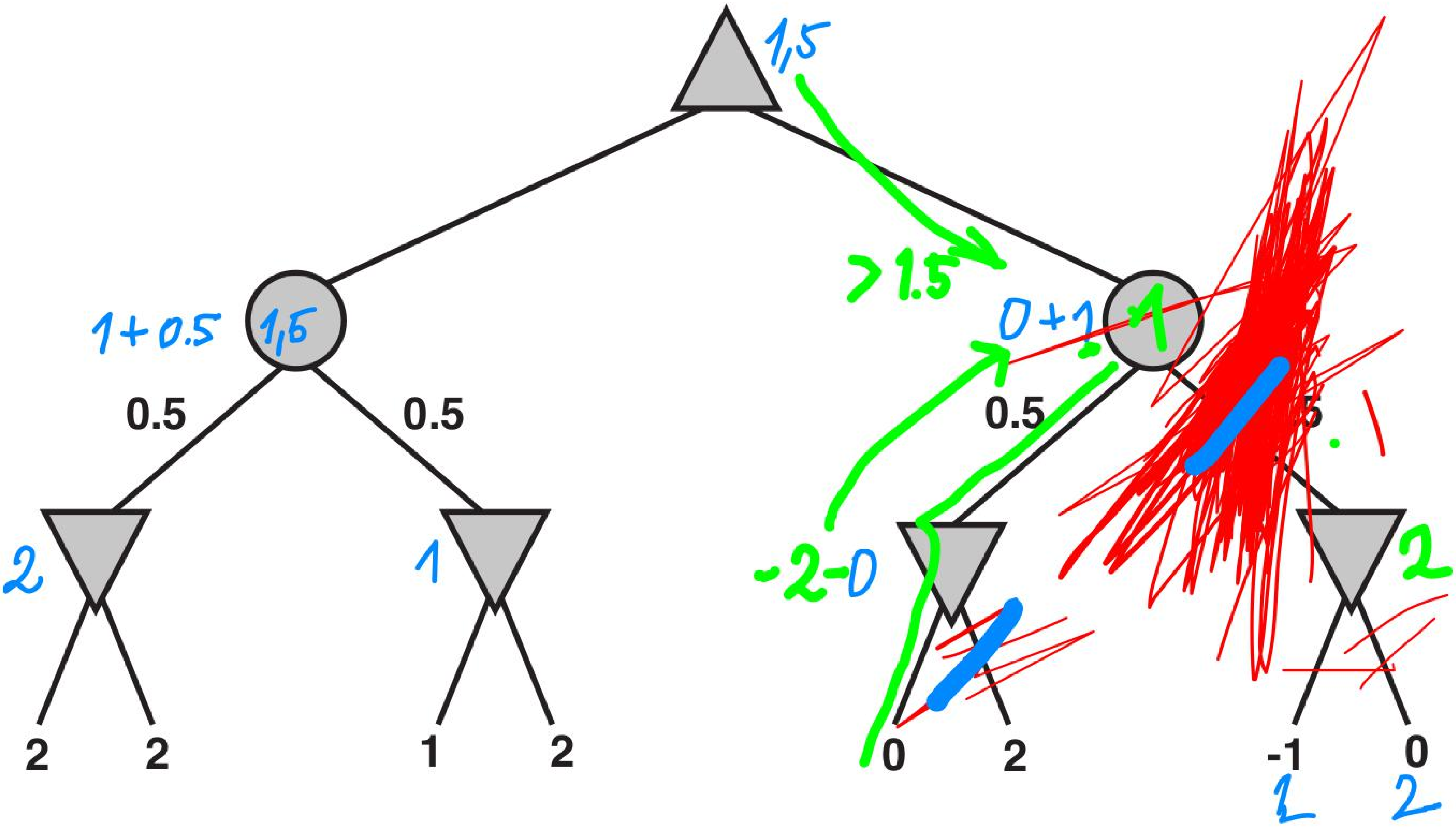

0.5
0.5
0.5
.5
2
2
1
2
0
2
-1
0


0.5 0.5 0.5 0.5
2 2 1 2 0 2 -1 0
A
0.5 0.5 0.5 0.5
2 2 1 2 0 2 -1 0
B
0.5 0.5 0.5 0.5
2 2 1 2 0 2 -1 0
C
0.5 0.5 0.5 0.5
2 2 1 2 -1 0
D
A 11 34%
B 5 15%
C 8 25%
D 8 25%


A
B
C
D
0.5
2 2 1 2 0 2 -1 0
A 11 34%
B 5 15%
C 8 25%
D 8 25%



0.5
0.5
0.5
2
2
1
2
0
2
-1
0


1,5
1+0.5
1,5
>1.5
0+1
1
0.5
0.5
0.5
2
1
-2-0
2
2
2
1
2
0
2
-1
0
1
2

Uncertain, partially observable environment

- Currents state s may be unknown, observations $\mathbf{e}$
- Uncertain outcome, random variable $\text{RESULT}(a)$
- Probability of outcome s' given $\mathbf{e}$ is

$$P(\text{RESULT}(a) = s' | a, \mathbf{e})$$

- Utility function $U(s)$ corresponds to agent preferences.
- Expected utility of an action a given $\mathbf{e}$:

$$EU(a|\mathbf{e}) = \sum_{s'} P(\text{RESULT}(a) = s' | a, \mathbf{e}) U(s')$$

Amatrice, Italy, 2016.

Uncertain, partially observable environment

- Currents state s may be unknown, observations $\mathbf{e}$
- Uncertain outcome, random variable $\text{RESULT}(a)$
- Probability of outcome s' given $\mathbf{e}$ is

$$P(\text{RESULT}(a) = s' | a, \mathbf{e})$$

- Utility function $U(s)$ corresponds to agent preferences.
- Expected utility of an action a given $\mathbf{e}$:

$$EU(a|\mathbf{e}) = \sum_{s'} P(\text{RESULT}(a) = s' | a, \mathbf{e}) U(s')$$

Amatrice, Italy, 2016.

Rational agent

Agent's expected utility of an action a given $\mathbf{e}$:

$$EU(a|\mathbf{e}) = \sum_{s'} P(\text{RESULT}(a) = s'|a, \mathbf{e})U(s')$$

What should a rational agent do?
Is it then all solved? Do we know all what we need?

- $P(\text{RESULT}(a) = s'|a, \mathbf{e})$
- $U(s')$

Rational agent

Agent's expected utility of an action a given $\mathbf{e}$:

$$EU(a|\mathbf{e}) = \sum_{s'} P(\text{RESULT}(a) = s'|a, \mathbf{e})U(s')$$

What should a rational agent do?

Is it then all solved? Do we know all what we need?

- $P(\text{RESULT}(a) = s'|a, \mathbf{e})$
- $U(s')$

Rational agent

Agent's expected utility of an action a given $\mathbf{e}$:

$$EU(a|\mathbf{e}) = \sum_{s'} P(\text{RESULT}(a) = s'|a, \mathbf{e})U(s')$$

What should a rational agent do?

Is it then all solved? Do we know all what we need?

- $P(\text{RESULT}(a) = s'|a, \mathbf{e})$
- $U(s')$

Rational agent

Agent's expected utility of an action a given $\mathbf{e}$:

$$EU(a|\mathbf{e}) = \sum_{s'} P(\text{RESULT}(a) = s'|a, \mathbf{e})U(s')$$

What should a rational agent do?
Is it then all solved? Do we know all what we need?

- $P(\text{RESULT}(a) = s'|a, \mathbf{e})$
- $U(s')$

Rational agent

Agent's expected utility of an action a given $\mathbf{e}$:

$$EU(a|\mathbf{e}) = \sum_{s'} P(\text{RESULT}(a) = s'|a, \mathbf{e})U(s')$$

What should a rational agent do?
Is it then all solved? Do we know all what we need?

- $P(\text{RESULT}(a) = s'|a, \mathbf{e})$
- $U(s')$