Artificial Intelligence – CM7

Multi-agent systems – Adversarial games

Arthur Bit-Monnot

INSA Toulouse – 4IR

Multiagent environments

One decision maker

When there are:

- a number of **actors**
- but only one **decision maker**

Actors simply do what they are told (*benevolent agent assumption*)

- e.g. a fleet of robots in a factory

Properties:

- not “truly” multi-agent
- problems arise from distributed execution of actions by several actors
  - concurrency, synchronization, …

Multiple decision makers

Several agents, where each:

- has preferences,
- chooses and executes its own plans

Multiple decision makers

Several agents, where each:

- has preferences,
- chooses and executes its own plans

Two main possibilities:

- agents have a **common goal**
  - e.g. workers in a company
  - challenge: coordination
- agents have **personal preferences**, pursued to the best of their abilities
  - e.g., players in games, car drivers, …

Game theory

Game theory: theory of strategic decision making

- strategic: a *player* takes into account what other *players* might do

Game theory

Game theory: theory of strategic decision making

- strategic: a *player* takes into account what other *players* might do
- **not** (only) for games!
  - auctioning oil drilling rights
  - product development and pricing
  - national defense

Opposed to *decision theory*: theoretical foundation for single agent AI

Game theory in AI

1. **Agent design**
   - exploit *game theory* to analyze possible decisions and compute expected utility
   - assumption: other agents act rationally (i.e. according to game theory)
2. **Mechanism design**
   - design the **rules** of the game
   - so that the **collective good** of all agents is maximized

Making collective decisions

Example: Group assignment

A cooperative game-theoretic situation:

- 65 students
- must be grouped into 5 subgroups
- each student with its own preferences regarding composition

Example: Group assignment

A cooperative game-theoretic situation:

- 65 students
- must be grouped into 5 subgroups
- each student with its own preferences regarding composition

Rings a bell?

Mechanism design (applied to group assignment)

A mechanism consists of:

1. **language for describing the allowed strategy**
   - formulation 0 to 3 vows

Mechanism design (applied to group assignment)

A mechanism consists of:

1. **language for describing the allowed strategy**
   - formulation 0 to 3 vows
2. **a distinguished agent that collects the strategy choice from the agents in the game**
   - me and my google form

Mechanism design (applied to group assignment)

A mechanism consists of:

1. **language for describing the allowed strategy**
   - formulation 0 to 3 vows
2. **a distinguished agent that collects the strategy choice from the agents in the game**
   - me and my google form
3. **an outcome rule, *known to all agents* used to determine the payoffs of each agent**
   - assignment maximizing the *global utility* (implementation detail: using the CPSat solver)

Utility in group assignment

Key challenge in collective decisions: capturing the agents' preferences

Here: approximated by a small number of vows

Utility in group assignment

Key challenge in collective decisions: capturing the agents' preferences

Here: approximated by a small number of vows

- **keeps complexity low**
  - easy to express for agents
  - limit computational complexity of the combinatorial problem
- **normalizes individual utility (in [0, 3])**
  - avoids *utility monsters*
  - Bob expressing a utility of 1000 for being with Alice

Example: Group assignment

Assignment 1

Individual utility (# vows fulfilled)
- Alice: 3
- Bob: 2
- Chloe: 0

Assignment 2

Individual utility (# vows fulfilled)
- Alice: 1
- Bob: 2
- Chloe: 1

Example: Group assignment

Assignment 1

Individual utility (# vows fulfilled)

- Alice: 3
- Bob: 2
- Chloe: 0

Social welfare:

- utilitarian (sum): 5
- egalitarian (min): 0

Assignment 2

Individual utility (# vows fulfilled)

- Alice: 1
- Bob: 2
- Chloe: 1

Social welfare:

- utilitarian (sum): 4
- egalitarian (min): 1

Social welfare

Measure of **social welfare** by aggregating individual utility:

Must balance between:

- total utility (sum)
- spreading among agents (min, Gini index, ...)

Common mechanisms

- **Auctions** (fr: enchères)
  - bid value correlates with agent utility
  - attribution to highest bid (proxy for best utility)
- **Utility alignment**
  - incentivizes agent to play for common good
- **Voting**
  - let agents express preferences
  - algorithm to choose based on preferences

Utility alignment (stock options ex.)

Intel's new CEO gets pay package valued at about $69 million[1]

Fortune – 15 march 2025

- $1 million base salary
- $2 performance bonus (cash)
- rest in stock options and equity, based on performance

[1] https://fortune.com/2025/03/15/intel-new-ceo-lip-bu-tan-pay-package-stock-options-bonus-69-million/

Utility alignment (stock options ex.)

Intel's new CEO gets pay package valued at about $69 million[1]

Fortune – 15 march 2025

- $1 million base salary
- $2 performance bonus (cash)
- rest in stock options and equity, based on performance

⇒ The compensation (decided by shareholders) gives Mr Tan great incentives to raise the stock's value 🚀

[1] https://fortune.com/2025/03/15/intel-new-ceo-lip-bu-tan-pay-package-stock-options-bonus-69-million/

Utility alignment

Process of defining the environment so that the agent's own utility **aligns with the one of designer's**.

E.g., incentives to work towards a company's objectives:

- legal (work contract, breaking it would have undesirable consequences)
- social (peer recognition, …)
- economical (bonuses, raise perspectives)

Utility alignment in policy making

Airplane

- price: 70 €
- eCO_2: 0.365 t

Train

- price: 110 €
- eCO_2: 0.002 t

[1] Social Cost of Carbon: https://en.wikipedia.org/wiki/Social_cost_of_carbon

Utility alignment in policy making

Airplane	Train
• price: 70 €	• price: 110 €
• eCO_2: 0.365 t	• eCO_2: 0.002 t

Global cost of emitting 1t eCO_2 is estimated to 1000€ shared among all earthling (8 billions)[1]

- personal cost / t: $\frac{1000}{8 \times 10^9} = 1.3 \times 10^{-7}$ €

[1] Social Cost of Carbon: https://en.wikipedia.org/wiki/Social_cost_of_carbon

Utility alignment in policy making

Airplane

- price: 70 €
- eCO_2: 0.365 t
  - self eCO_2 cost: ≈ 0€
  - shared eCO_2 cost: 365 €

Train

- price: 110 €
- eCO_2: 0.002 t
  - self eCO_2 cost: ≈ 0€
  - shared eCO_2 cost: 2 €

Global cost of emitting 1t eCO_2 is estimated to 1000€ shared among all earthling (8 billions)[1]

- personal cost / t: $\frac{1000}{8 \times 10^9} = 1.3 \times 10^{-7}$ €
- my personal interest is to take the airplane (regardless of others)
- yet, total utility increase by 323€ if I take the train

[1] Social Cost of Carbon: https://en.wikipedia.org/wiki/Social_cost_of_carbon

Utility alignment (tragedy of the commons)

Tragedy of the commons

If nobody has to pay for a shared resource, it may be exploited in a way that leads to a lower utility for all agents. ♣

Solution: charge agents for common resource usage (externalities)

- Align agent's preferences to that maximizing his utility maximizes common good

E.g. a carbon price of 200€ would increase the cost of the plane ticket by 73€

Did the designers of the fifth Republic
had a background in social choice[1]?

[1] https://en.wikipedia.org/wiki/Social_choice_theory

Mechanism designs: Vote

Did the designers of the fifth Republic had a background in social choice[1]?

No.
Two-round majority vote lacks almost all desirable properties for a voting system.

[1] https://en.wikipedia.org/wiki/Social_choice_theory

Agent design

Decision whan facing other agents

Others agents may be considered as:

- an **economy**: when there are many agents whose impact can be considered as aggregated
  - e.g. increase in demand causes prices to rises
- part of the **environment**: when their strategies are independent of our own strategy
- **explicitly modeled**: in adversarial environments
  - adversarial game-tree search

Two-players zero-sum game

Most studied games (chess, go, …) are:

- two players
- deterministic
- turn-taking
- perfect information (= fully observable)
- zero-sum

Zero-sum game

What is good for a player is equaly bad for the other. Any gain for one is an equivalent for the other.

♣

Utility in zero-sum games

- my own utility function `SelfUtility(s)` that I try to maximize.
- advsary's utility function is `AdvUtility(s) = - SelfUtility(s)`
  - equivalently, he should to **minimize** my own utility function.

A tiny zero-sum game

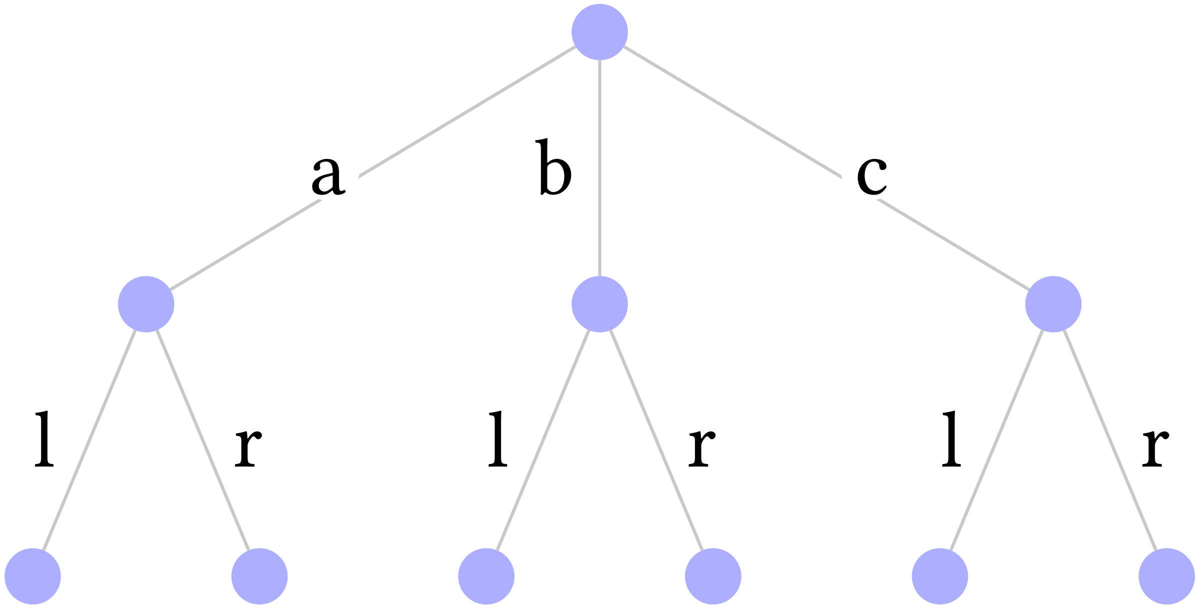
Consider a turn-taking game where:

- I start
- I have three available actions {a, b, c}
- my *adversary* has two actions {l, r}

Minimax game tree

Self's turn:

Adv's turn:



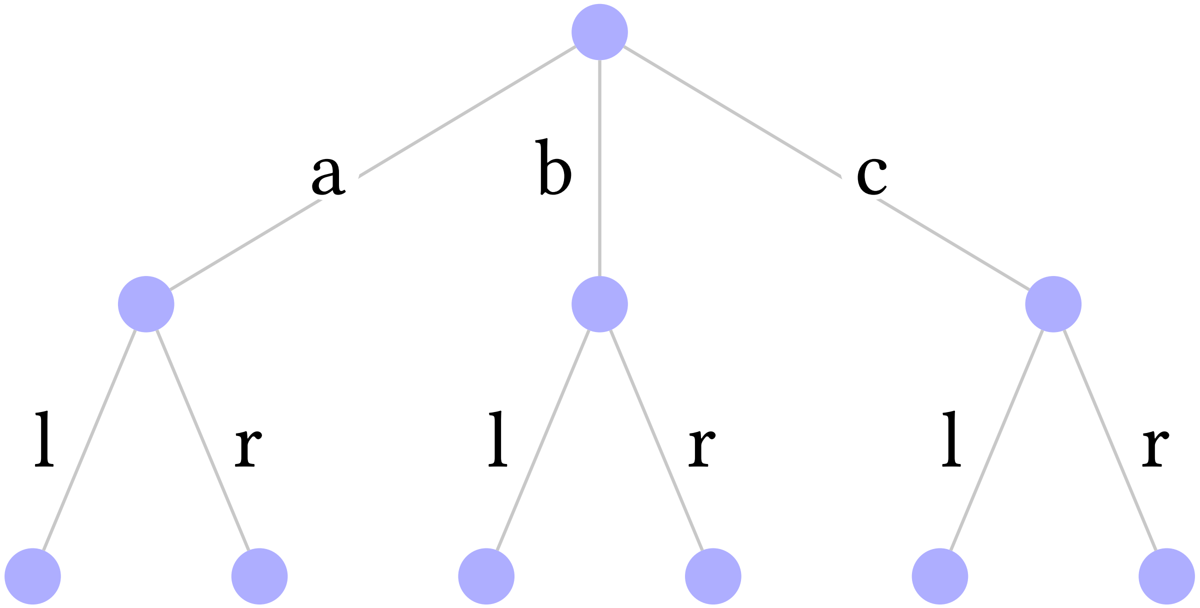
Minimax game tree

Self's turn:

Adv's turn:

Self’s evaluation

(terminal / heuristic / rec call)



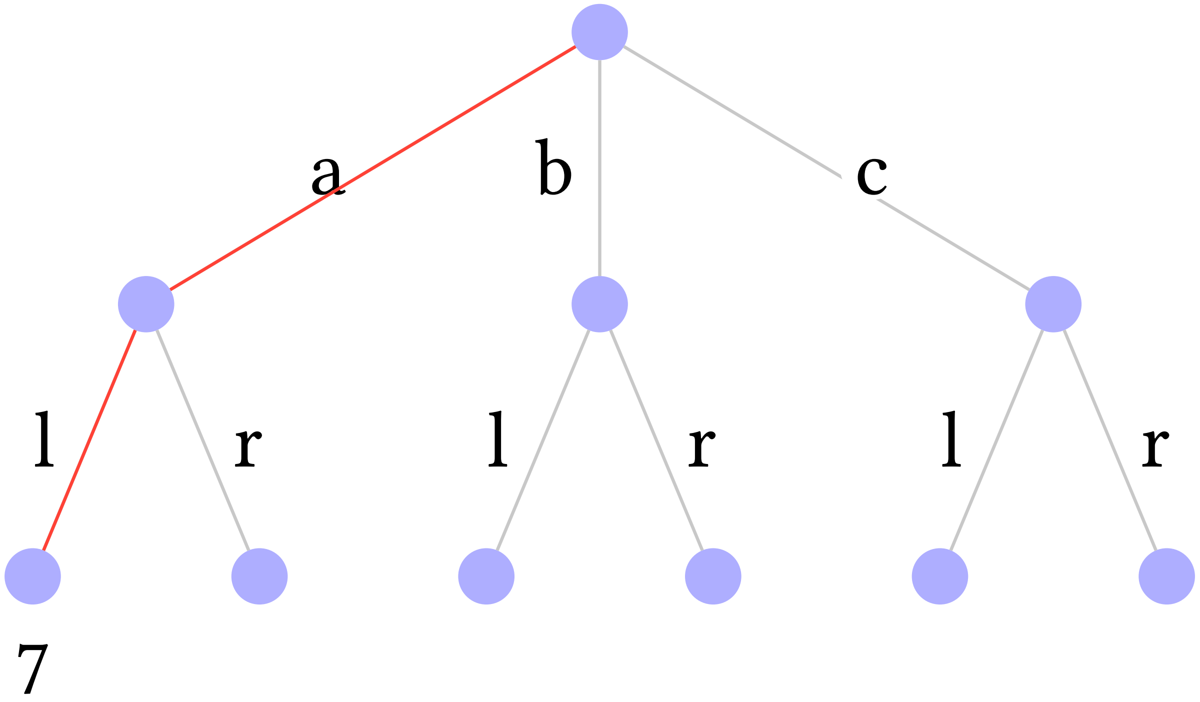
Minimax game tree

Self's turn:

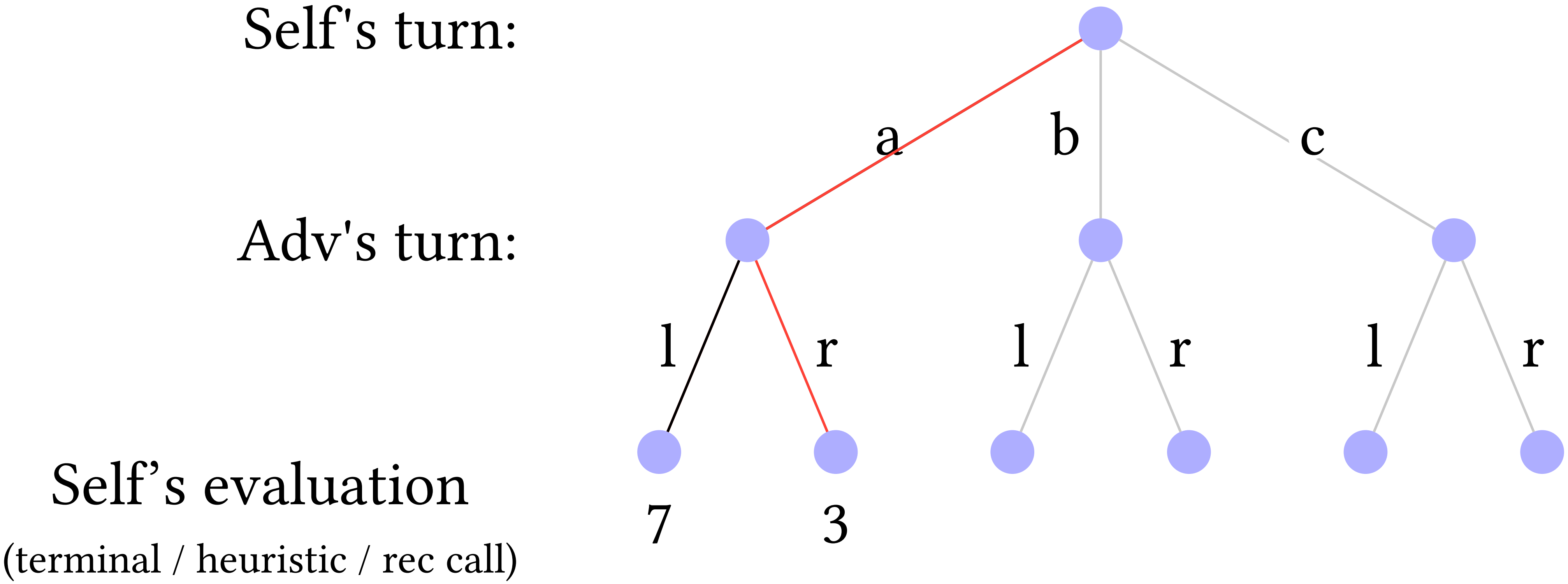
Adv's turn:

Self's evaluation

(terminal / heuristic / rec call)



Minimax game tree



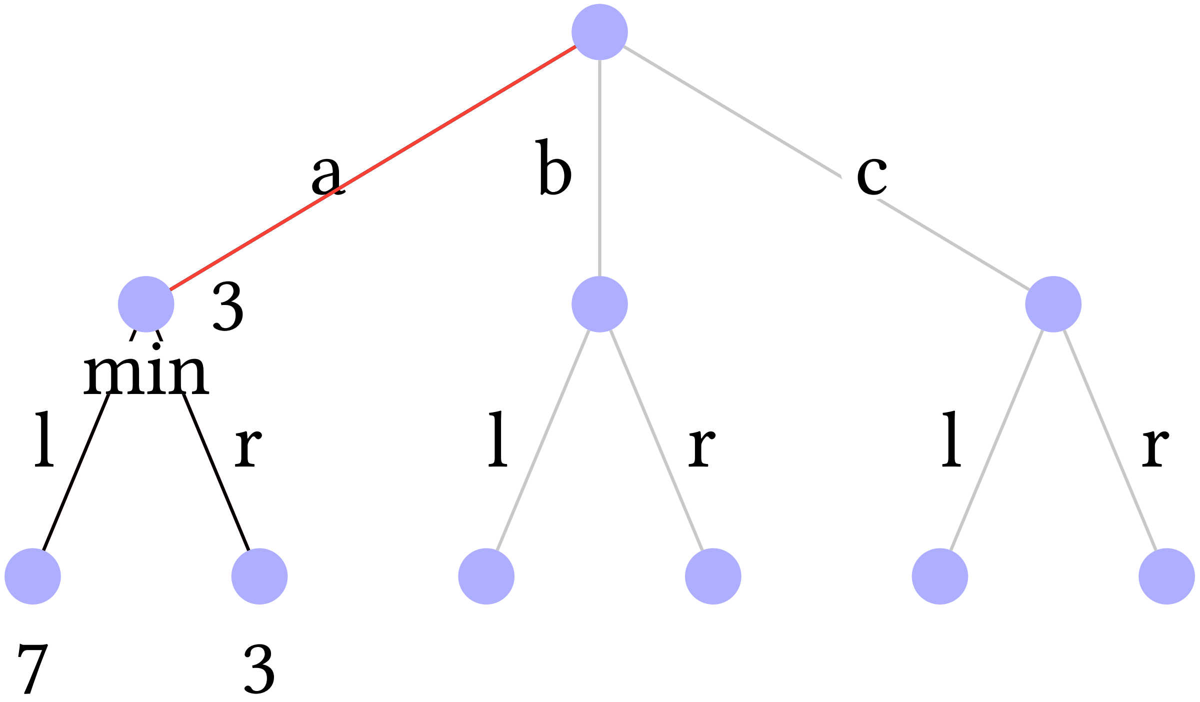
Minimax game tree

Self's turn:

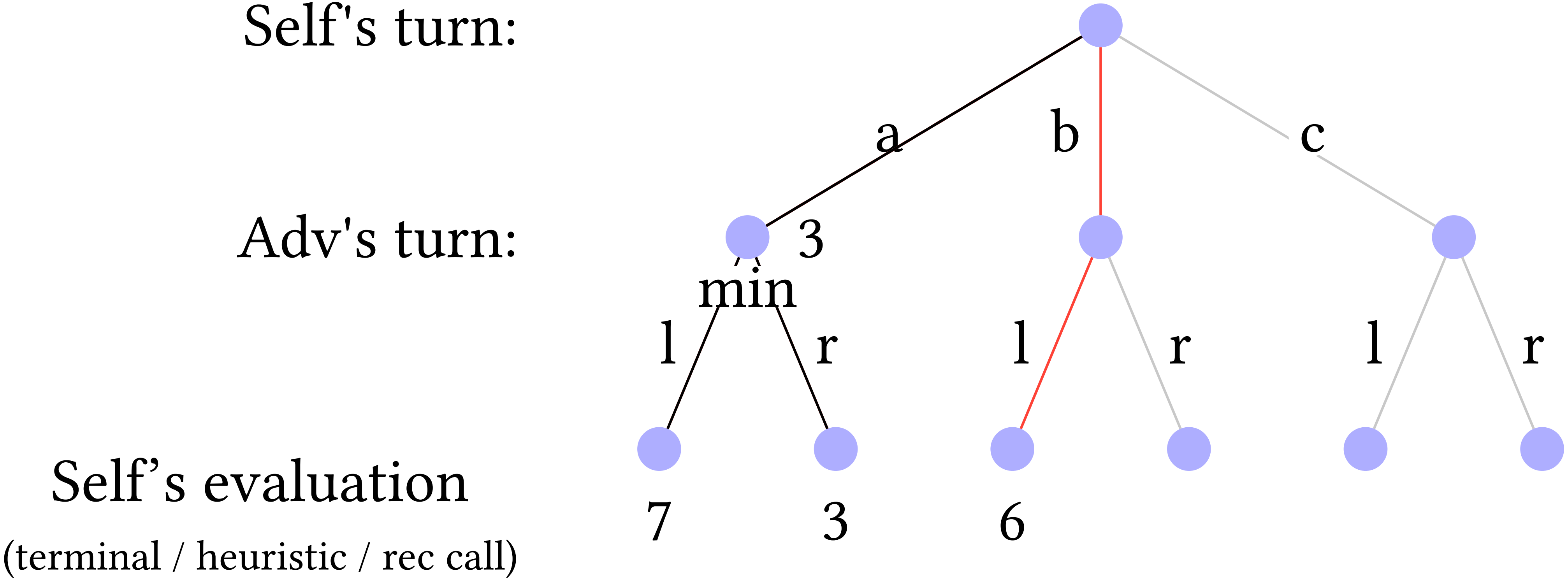
Adv's turn:

Self's evaluation

(terminal / heuristic / rec call)



Minimax game tree



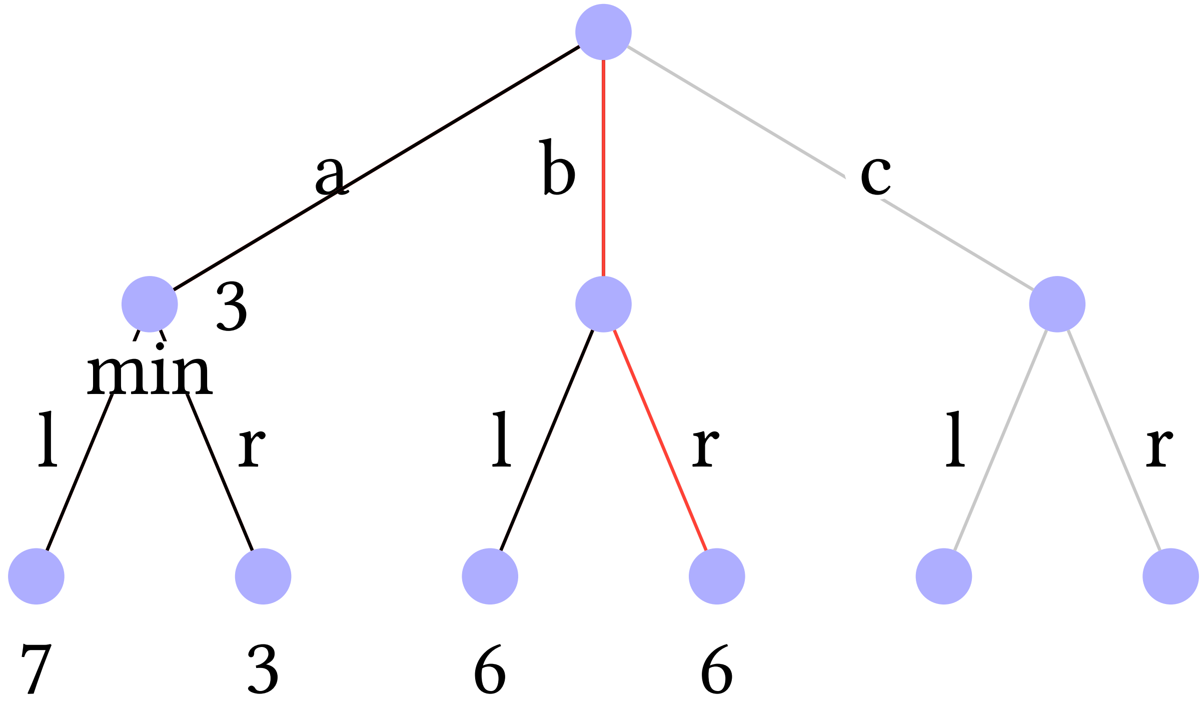
Minimax game tree

Self's turn:

Adv's turn:

Self's evaluation

(terminal / heuristic / rec call)



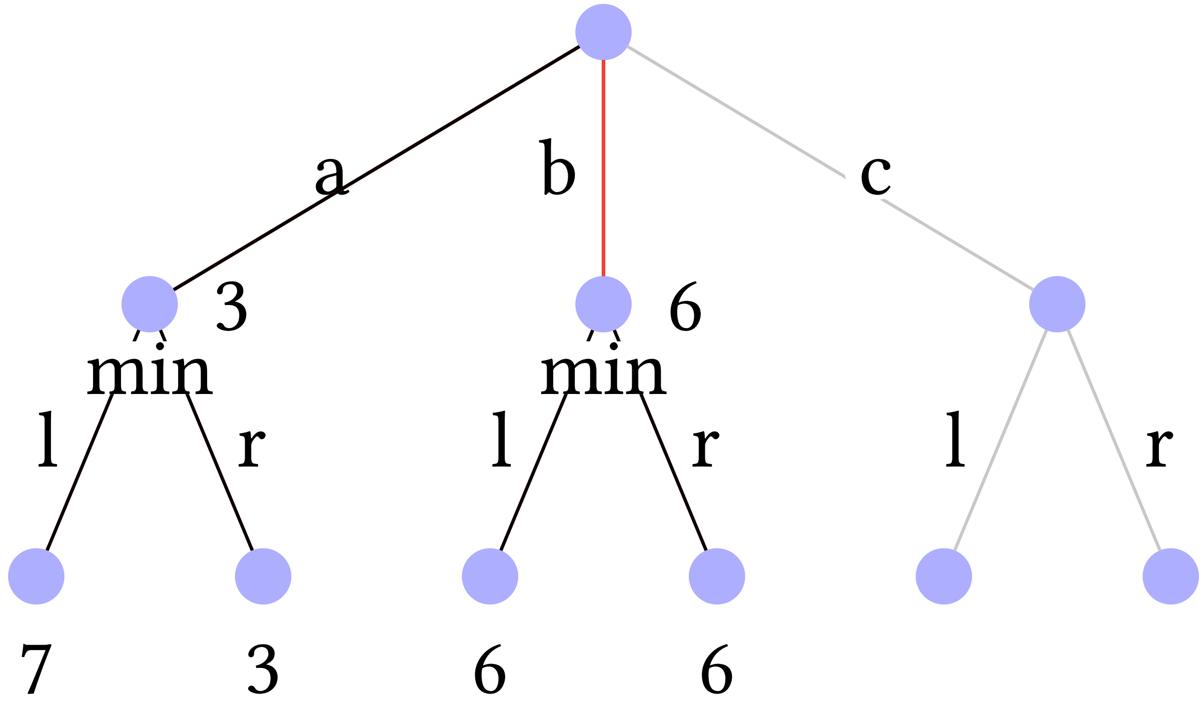
MINIMAX game tree

Self's turn:

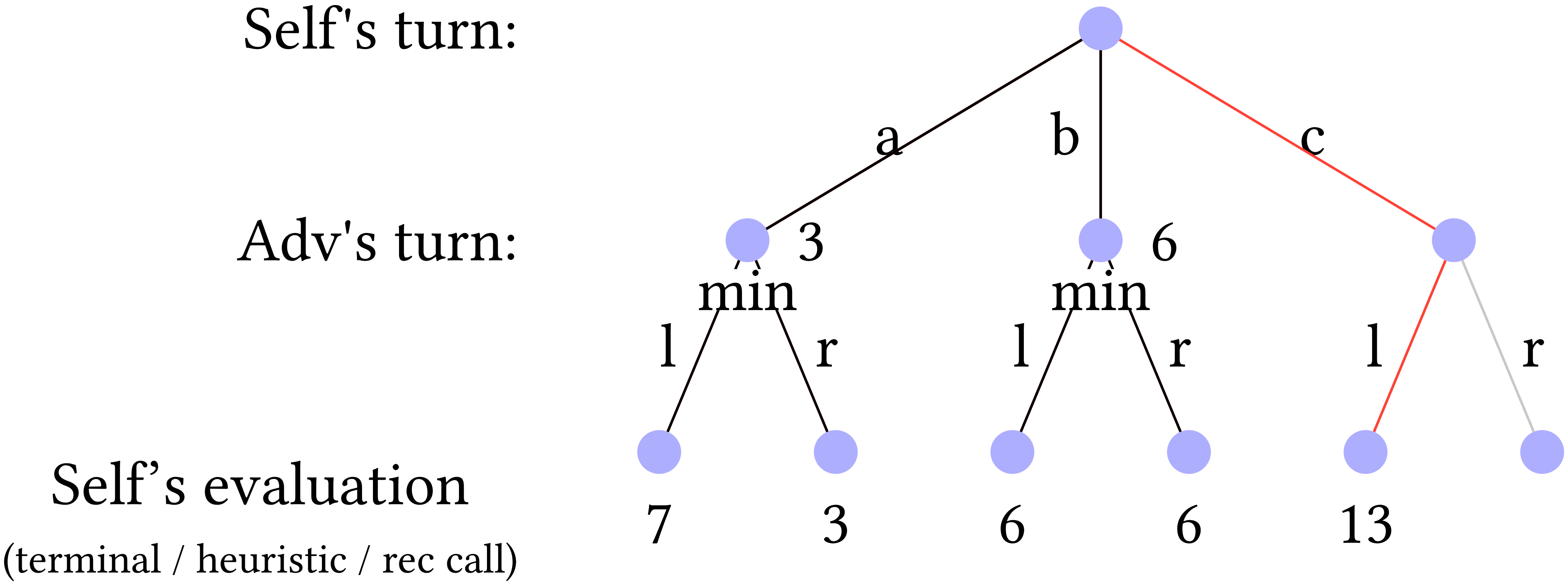
Adv's turn:

Self's evaluation

(terminal / heuristic / rec call)



Minimax game tree



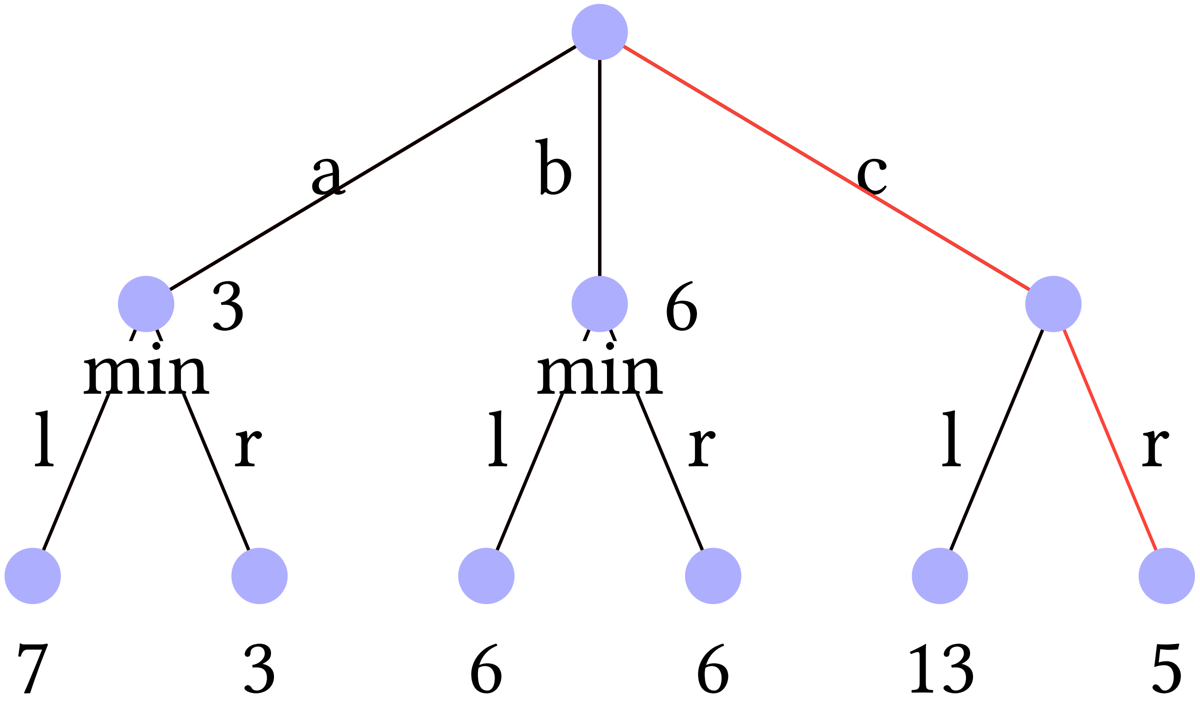
Minimax game tree

Self's turn:

Adv's turn:

Self's evaluation

(terminal / heuristic / rec call)



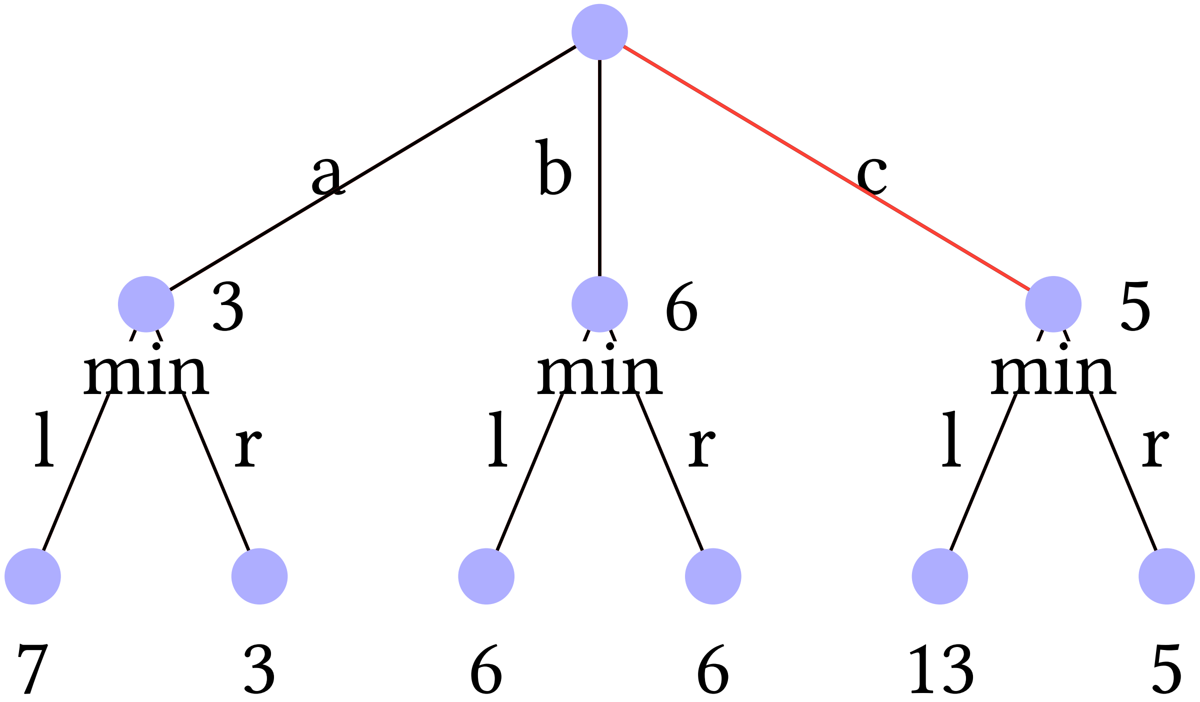
MINIMAX game tree

Self's turn:

Adv's turn:

Self’s evaluation

(terminal / heuristic / rec call)



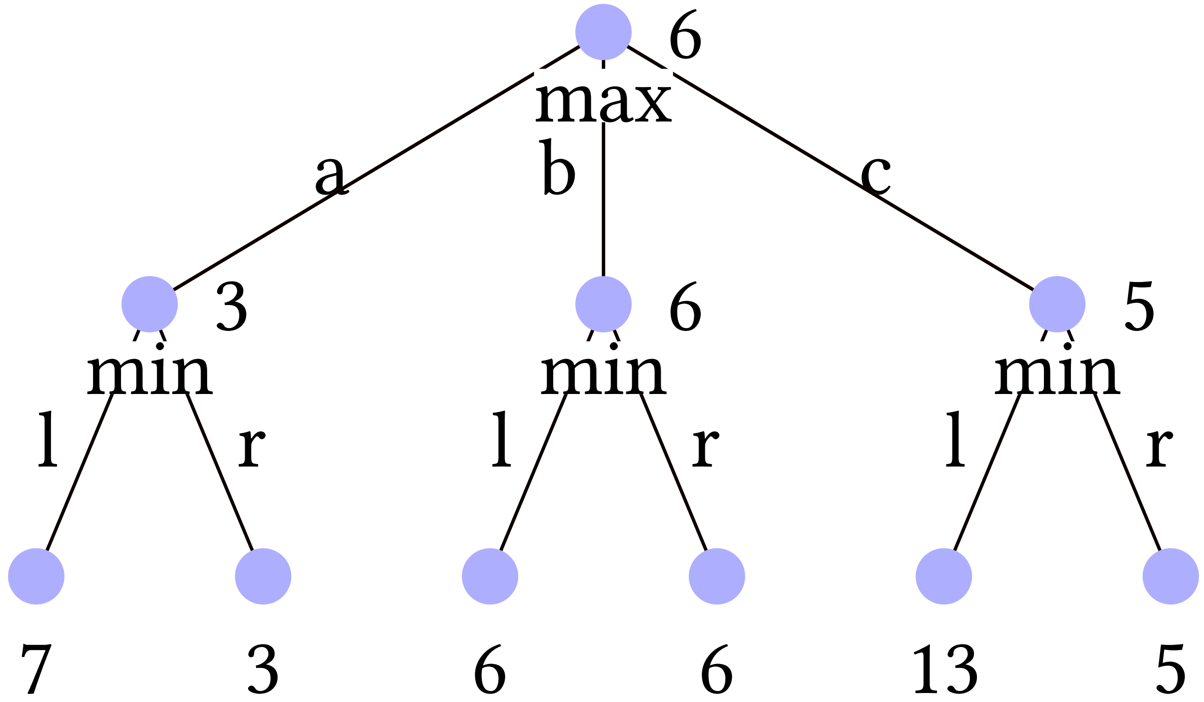
Minimax game tree

Self's turn:

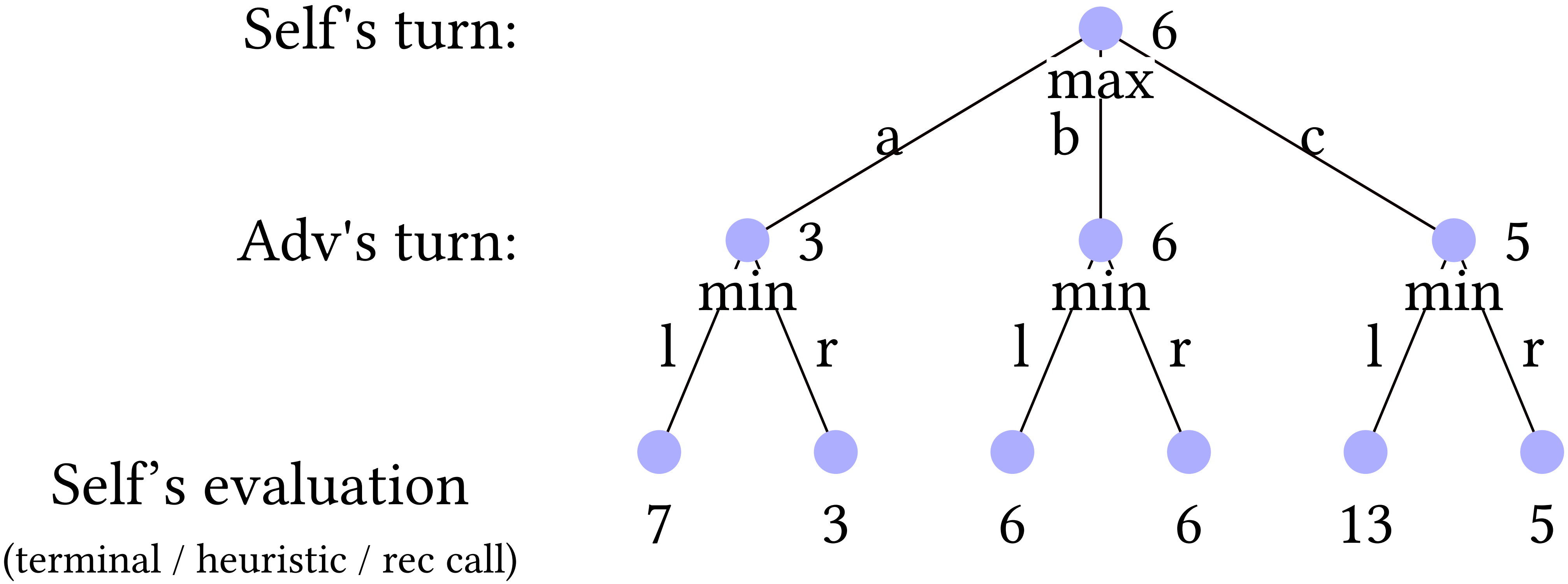
Adv's turn:

Self's evaluation

(terminal / heuristic / rec call)



Minimax game tree



Minimax equations

$$\text{Minimax}(s) := \begin{cases} \text{SelfUtility}(s) & \text{if IsTerminal}(s) \\ \max_{a \in \text{ SelfActions}(s)} \text{Minimax}(\text{Result}(s, a)) & \text{if ToMove}(s) = \text{Self} \\ \min_{a \in \text{ AdvActions}(s)} \text{Minimax}(\text{Result}(s, a)) & \text{if ToMove}(s) = \text{Adv} \end{cases}$$

Minimax properties

Complexity:

- Time: $O\left(b^d\right)$
- Spatial: $O(d)$

where d is the maximum depth and b is the branching factor.

Minimax Optimality

The Minimax algorithm is optimal if both player act rationnaly

♡

Generalization beyond zero-sum games

Each agent has a utility function:

- agent A: $U_{A(s)}$
- agent B: $U_{B(s)}$
- …

Each state s has a utility vector $[U_{A(s)}, U_{B(s)}]$

Generalization beyond zero-sum games

Each agent has a utility function:

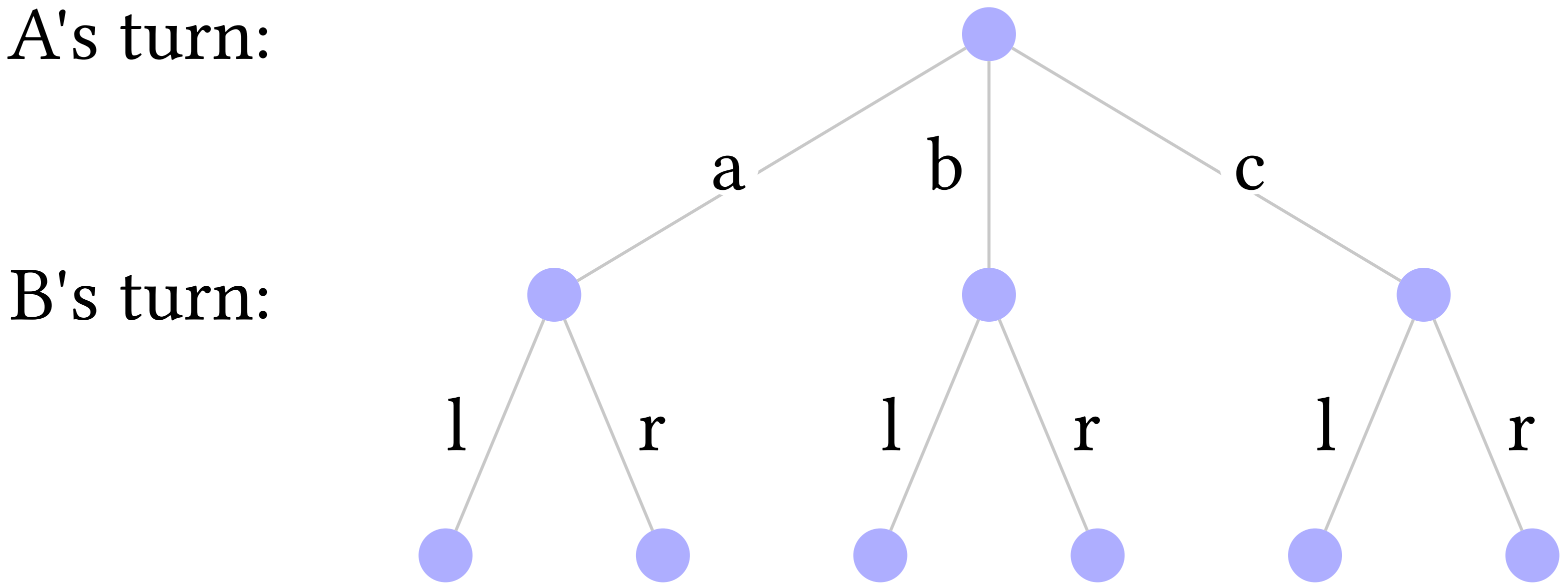
- agent A: $U_{A(s)}$
- agent B: $U_{B(s)}$
- …

Each state s has a utility vector $[U_{A(s)}, U_{B(s)}]$

Optimal game-tree search:

- each agent maximizes its own component of the utility vector
- strict generalization of Minimax

Optimal game-tree search



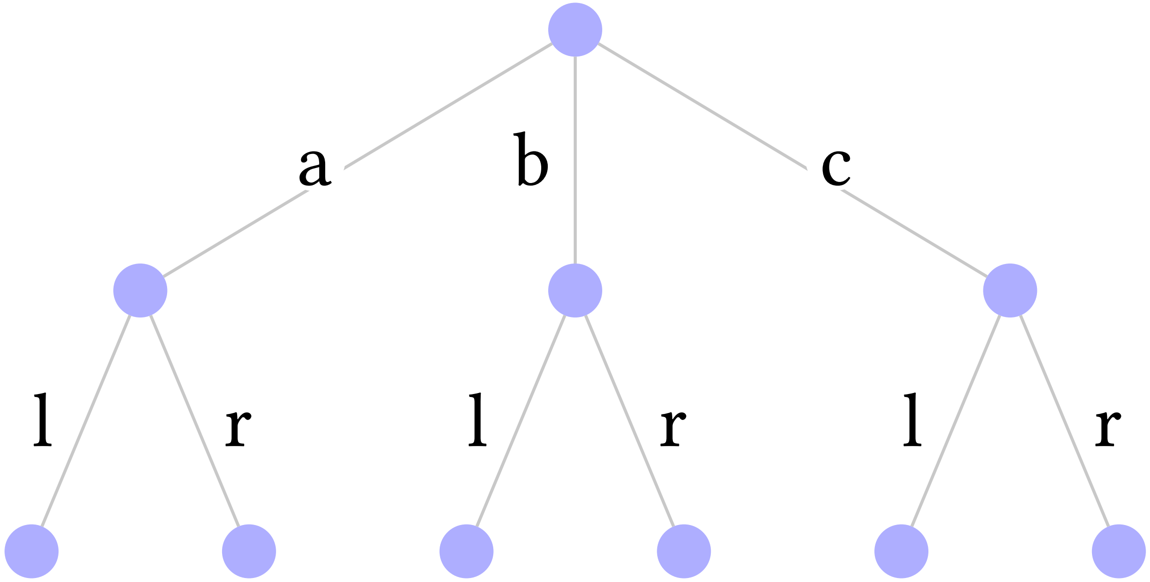
Optimal game-tree search

A's turn:

B's turn:

Evaluation $[U_A, U_B]$

(terminal / heuristic / rec call)



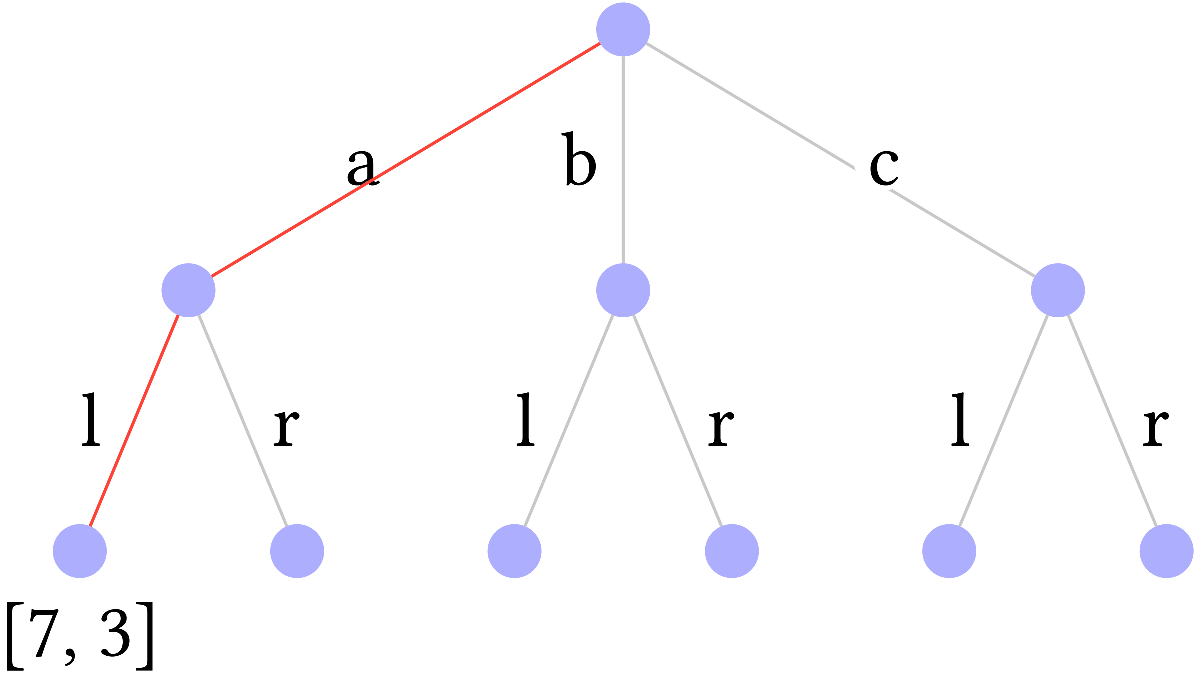
Optimal game-tree search

A's turn:

B's turn:

Evaluation $[U_A, U_B]$

(terminal / heuristic / rec call)



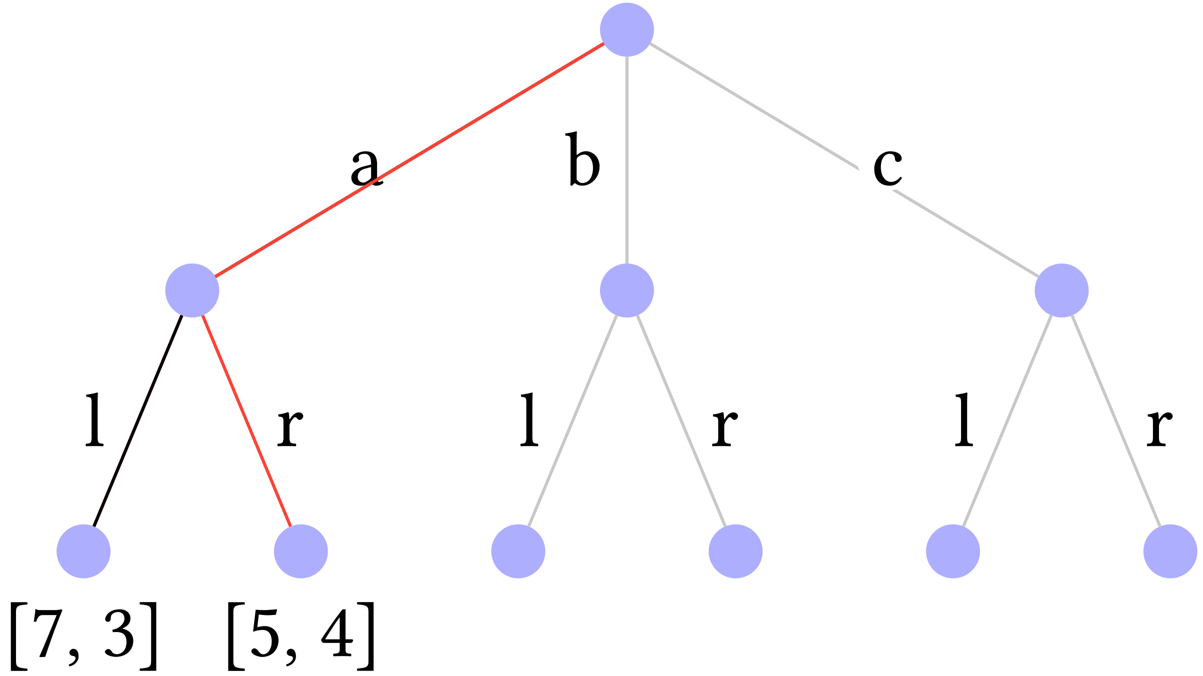
Optimal game-tree search

A's turn:

B's turn:

Evaluation $[U_A, U_B]$

(terminal / heuristic / rec call)



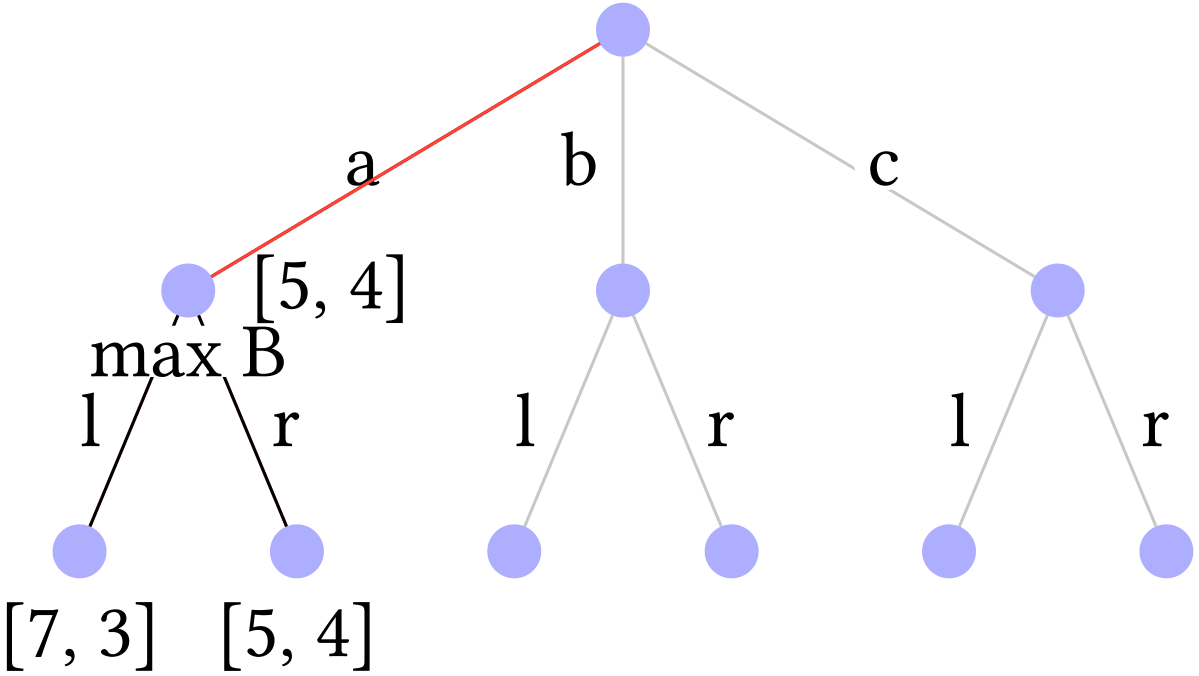
Optimal game-tree search

A's turn:

B's turn:

Evaluation $[U_A, U_B]$

(terminal / heuristic / rec call)



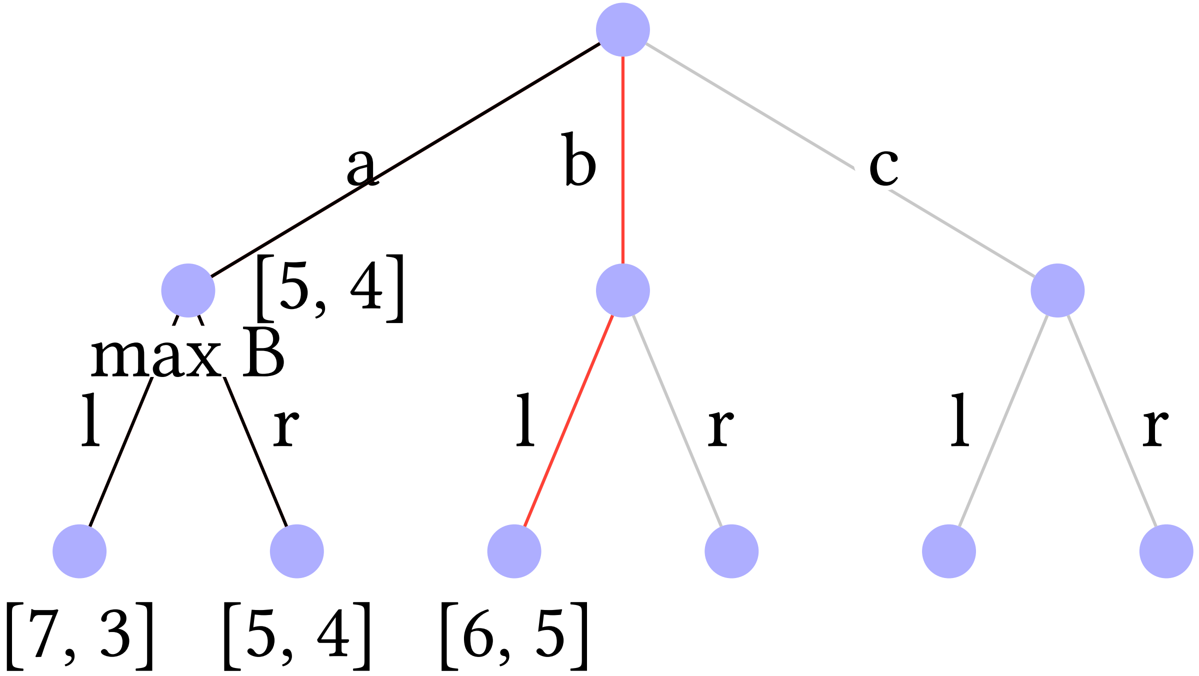
Optimal game-tree search

A's turn:

B's turn:

Evaluation $[U_A, U_B]$

(terminal / heuristic / rec call)



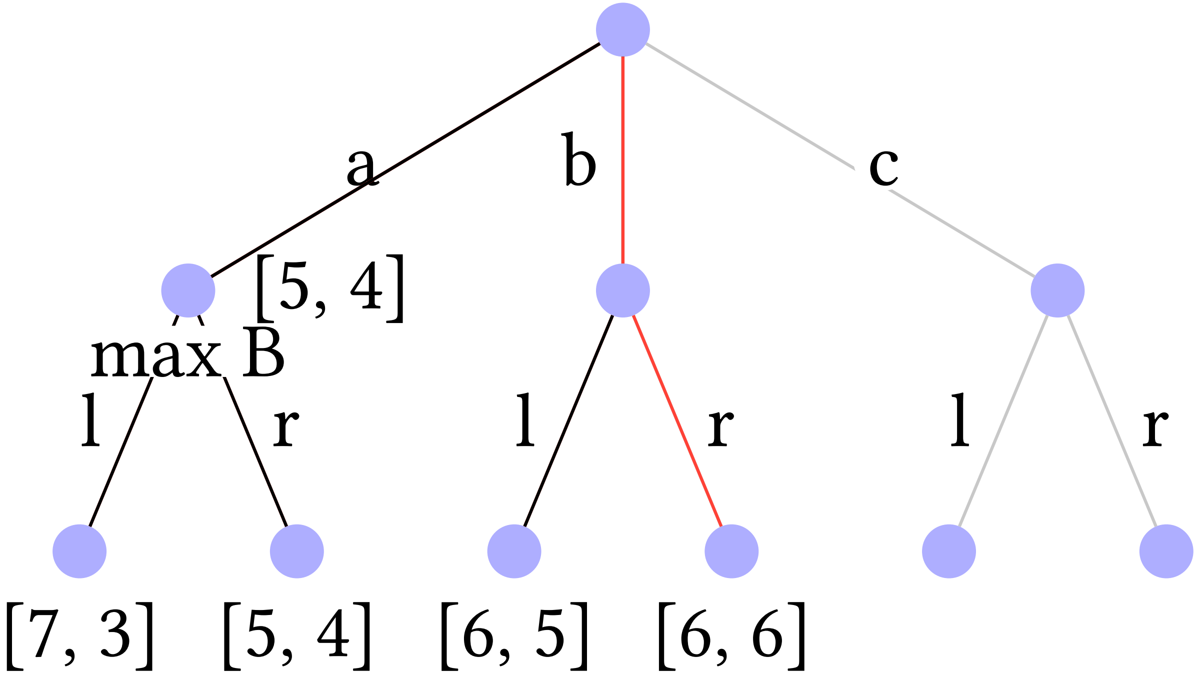
Optimal game-tree search

A's turn:

B's turn:

Evaluation $[U_A, U_B]$

(terminal / heuristic / rec call)



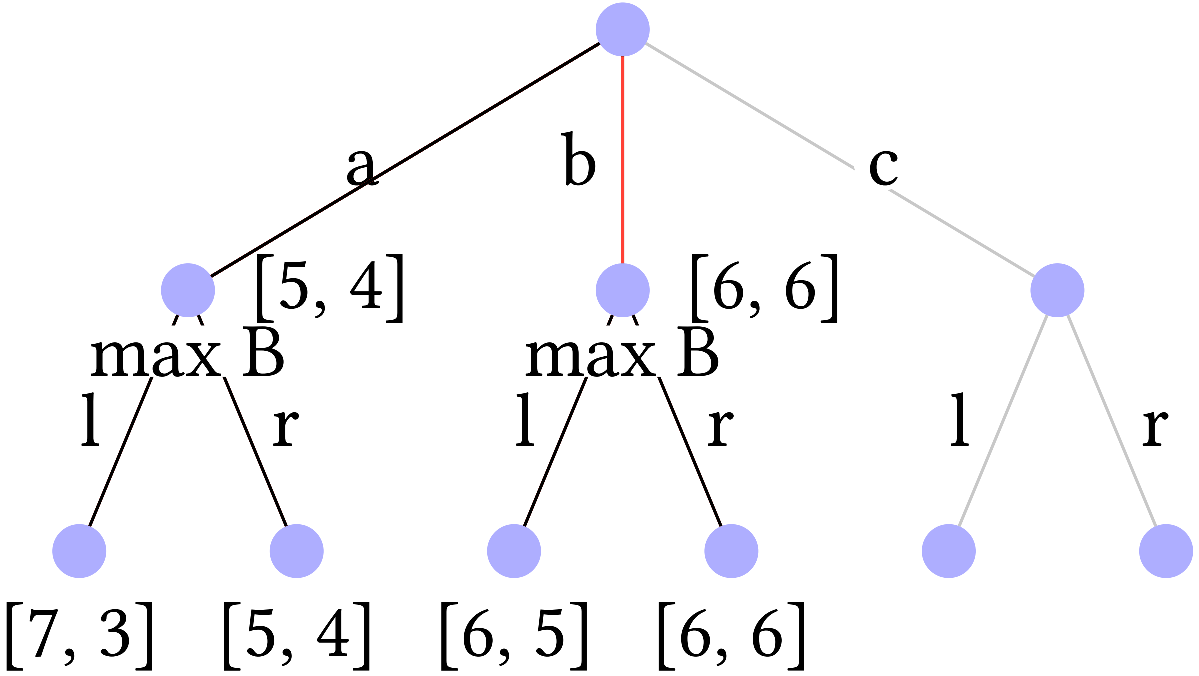
Optimal game-tree search

A's turn:

B's turn:

Evaluation $[U_A, U_B]$

(terminal / heuristic / rec call)



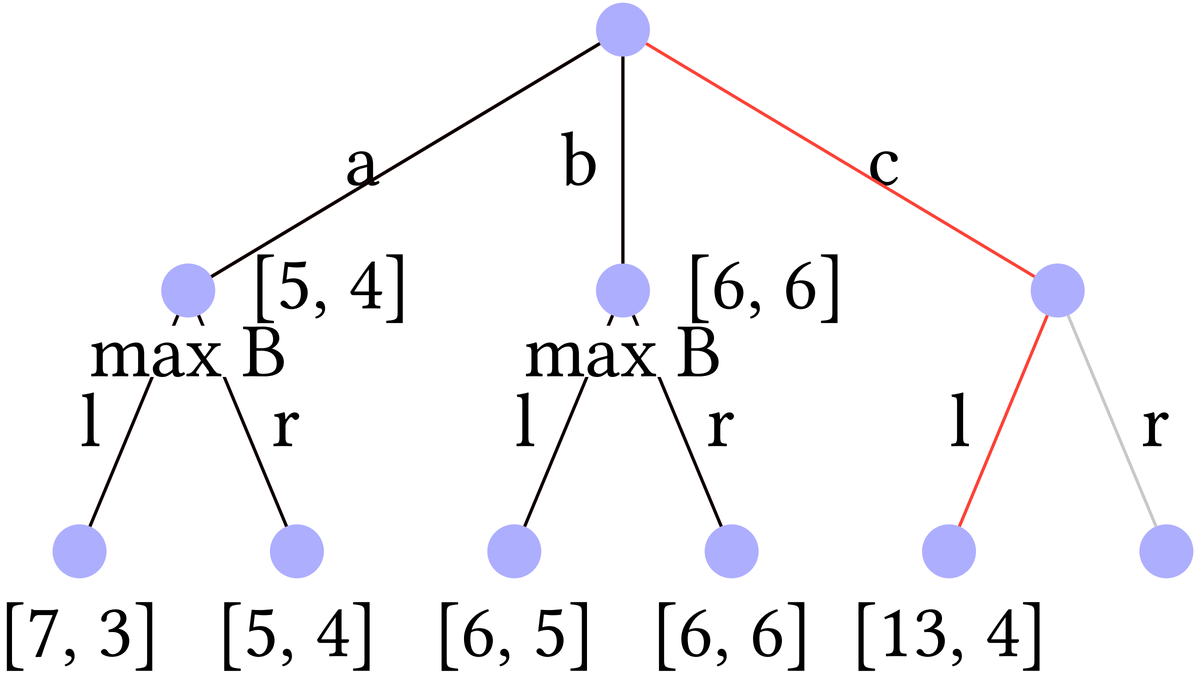
Optimal game-tree search

A's turn:

B's turn:

Evaluation $[U_A, U_B]$

(terminal / heuristic / rec call)



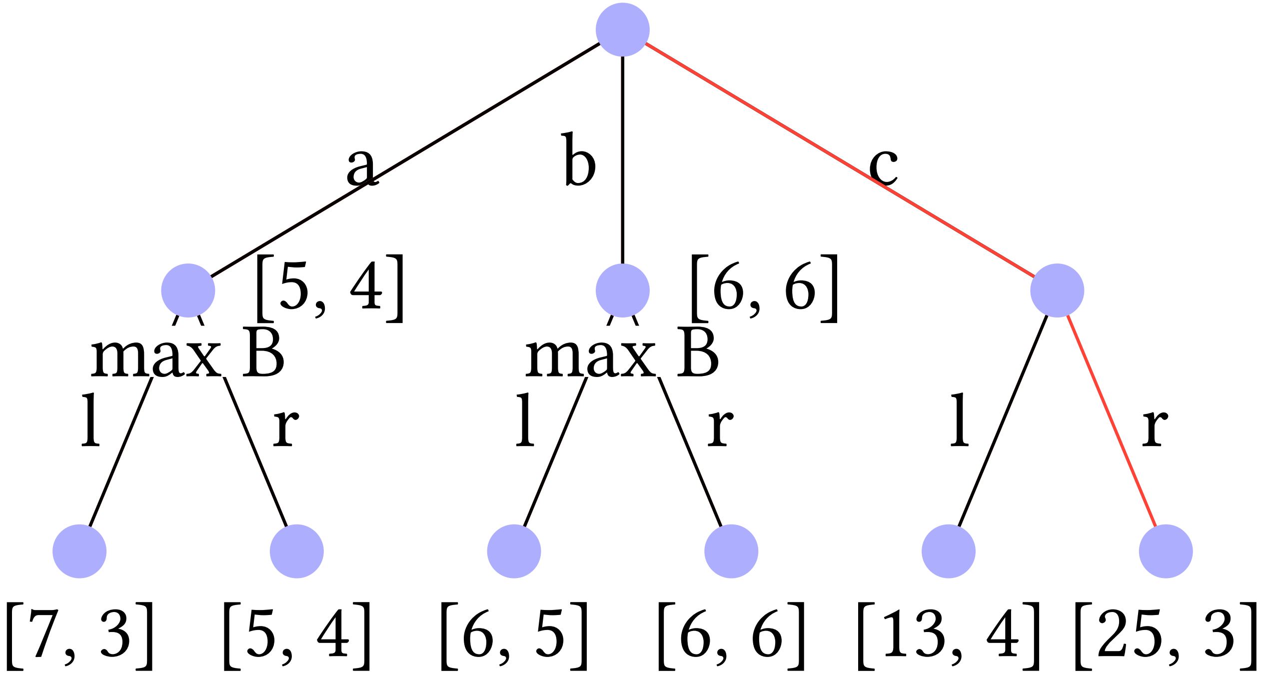
Optimal game-tree search

A's turn:

B's turn:

Evaluation $[U_A, U_B]$

(terminal / heuristic / rec call)



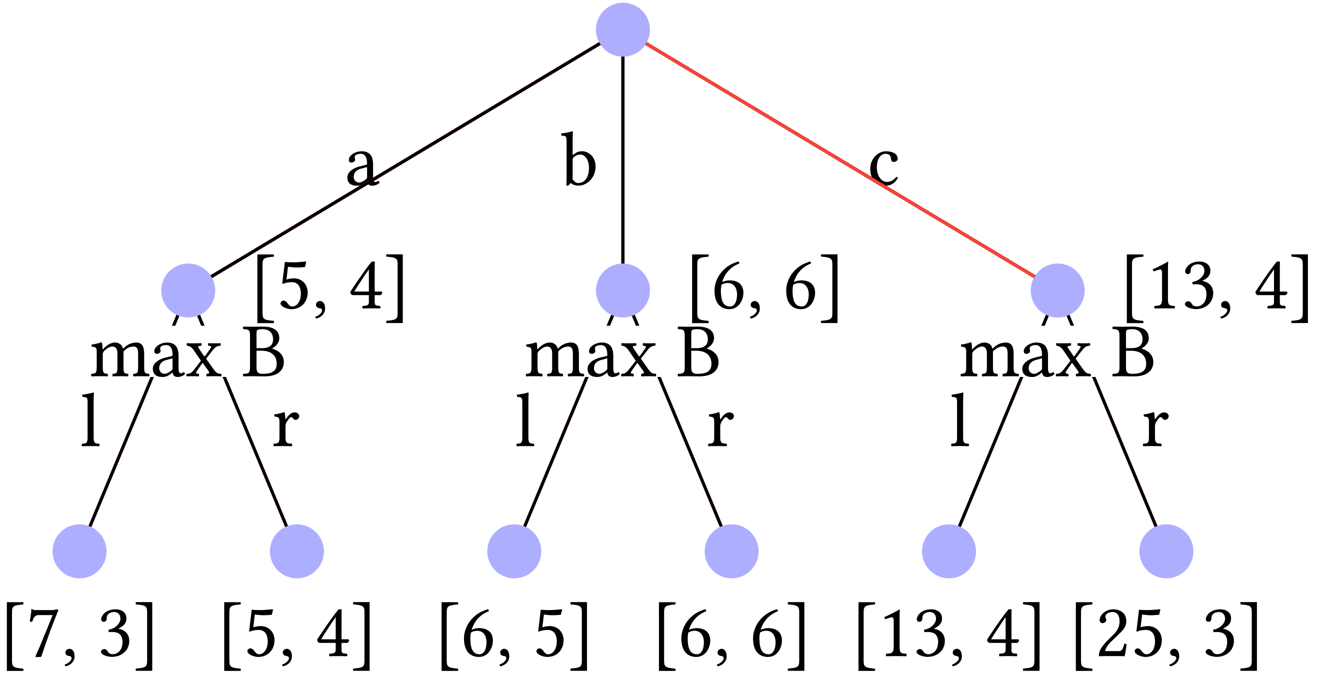
Optimal game-tree search

A's turn:

B's turn:

Evaluation $[U_A, U_B]$

(terminal / heuristic / rec call)



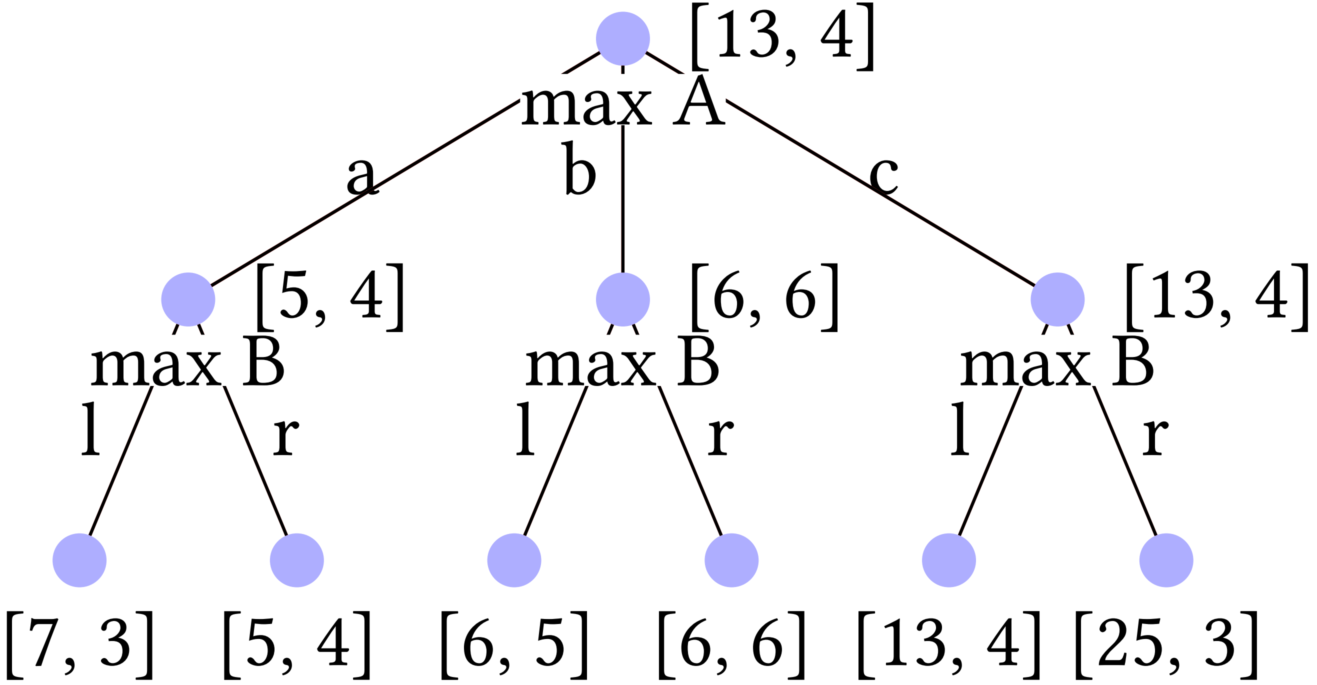
Optimal game-tree search

A's turn:

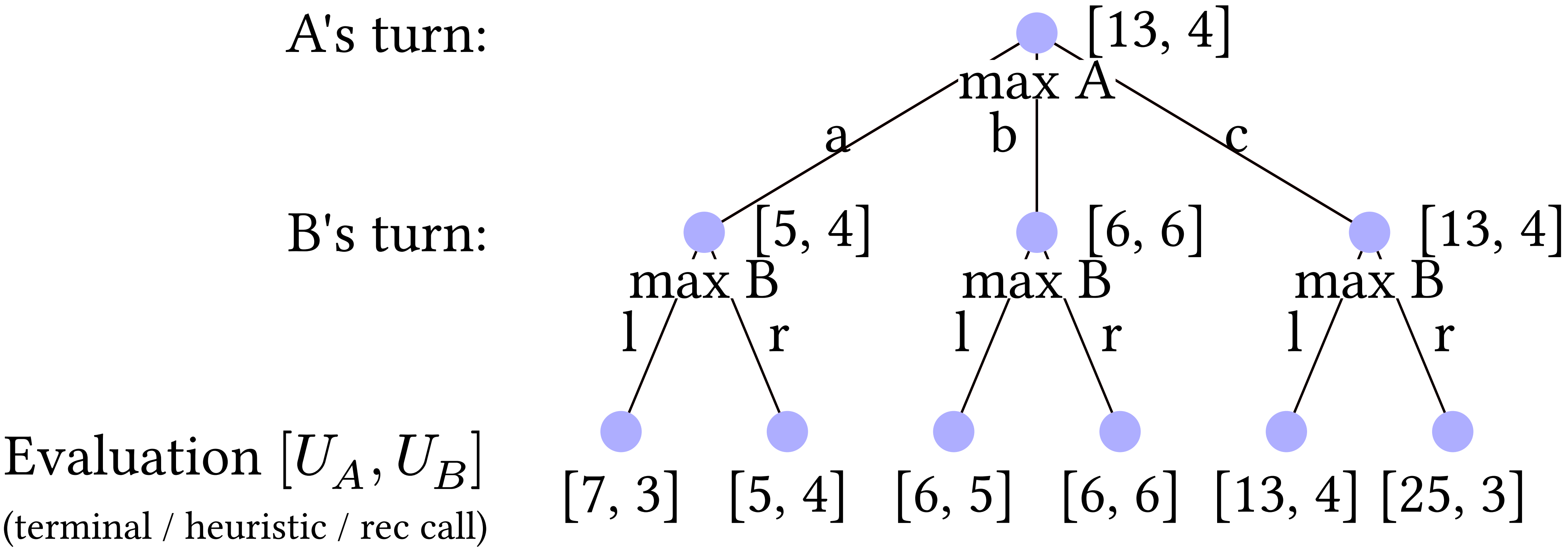
B's turn:

Evaluation $[U_A, U_B]$

(terminal / heuristic / rec call)



Optimal game-tree search

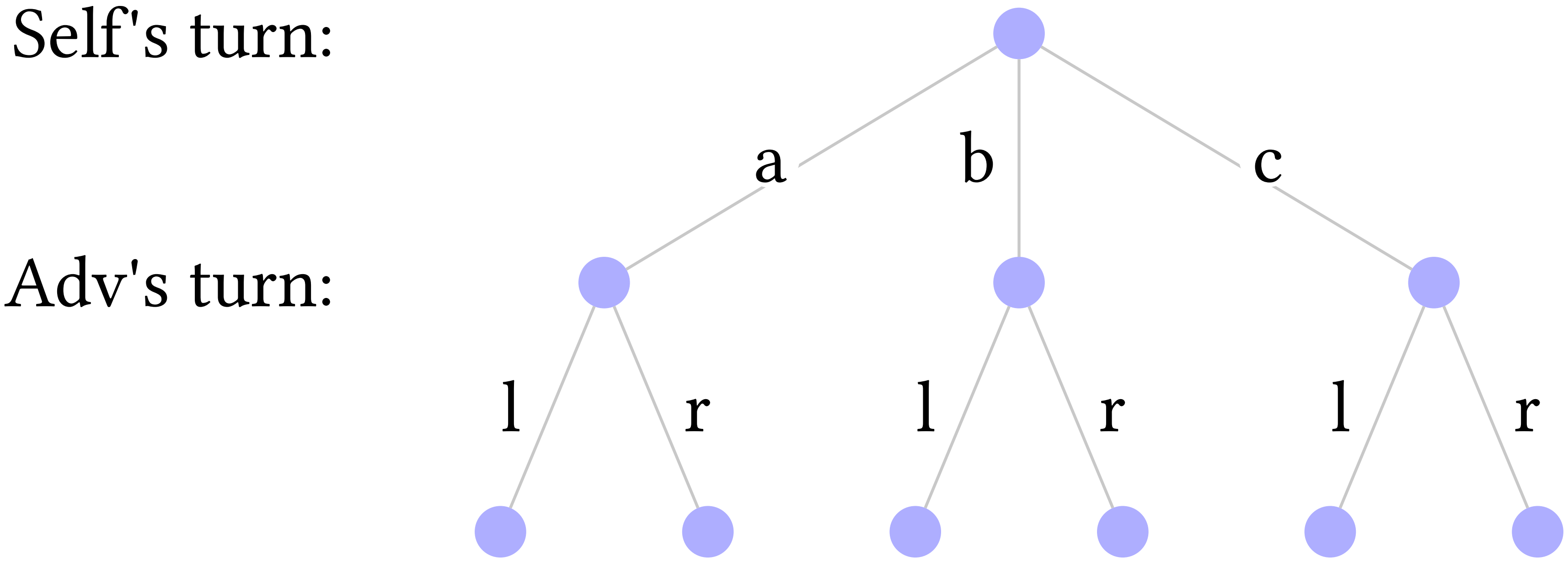


MINIMAX's exponential complexity

Problem: complexity is exponential is the tree depth

Could we avoid useless computations?

Alpha-Beta game tree



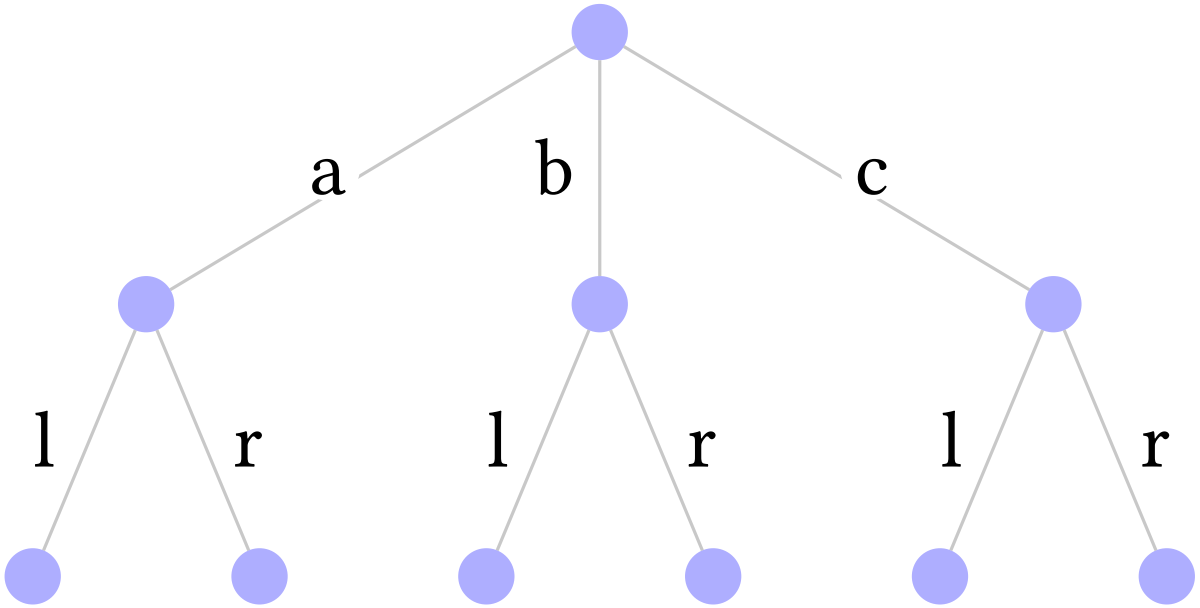
Alpha-Beta game tree

Self's turn:

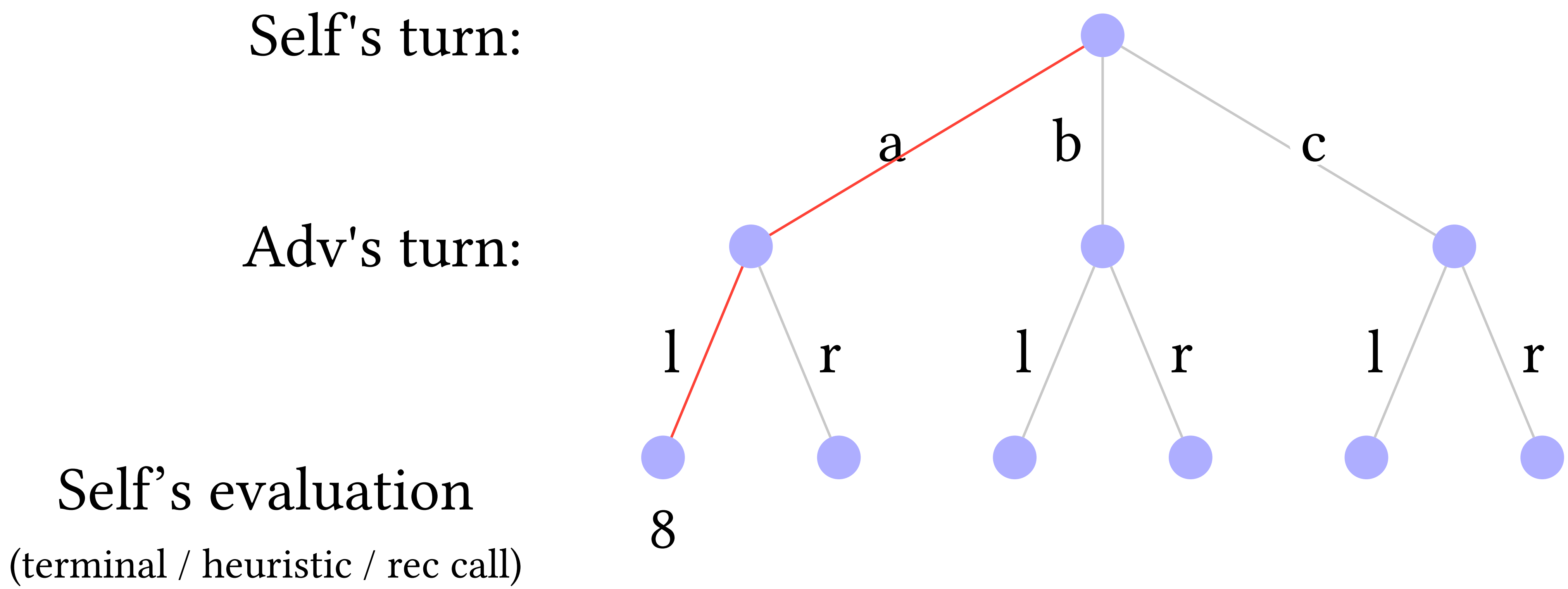
Adv's turn:

Self's evaluation

(terminal / heuristic / rec call)



Alpha-Beta game tree



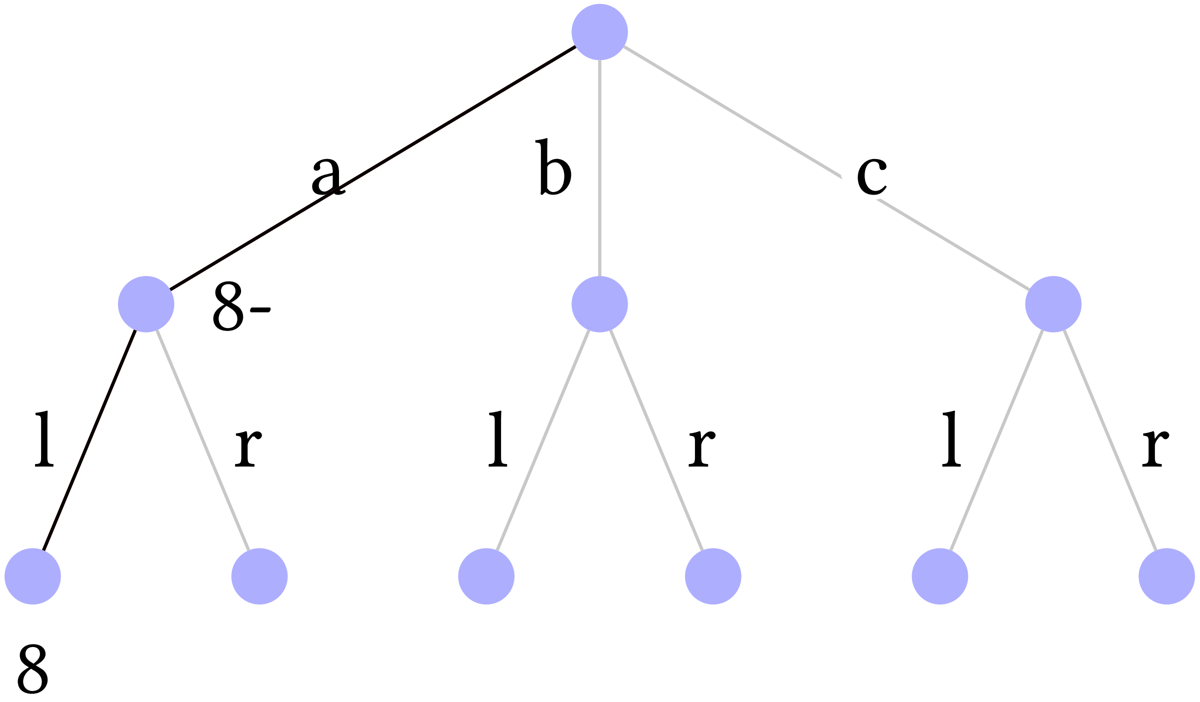
Alpha-Beta game tree

Self's turn:

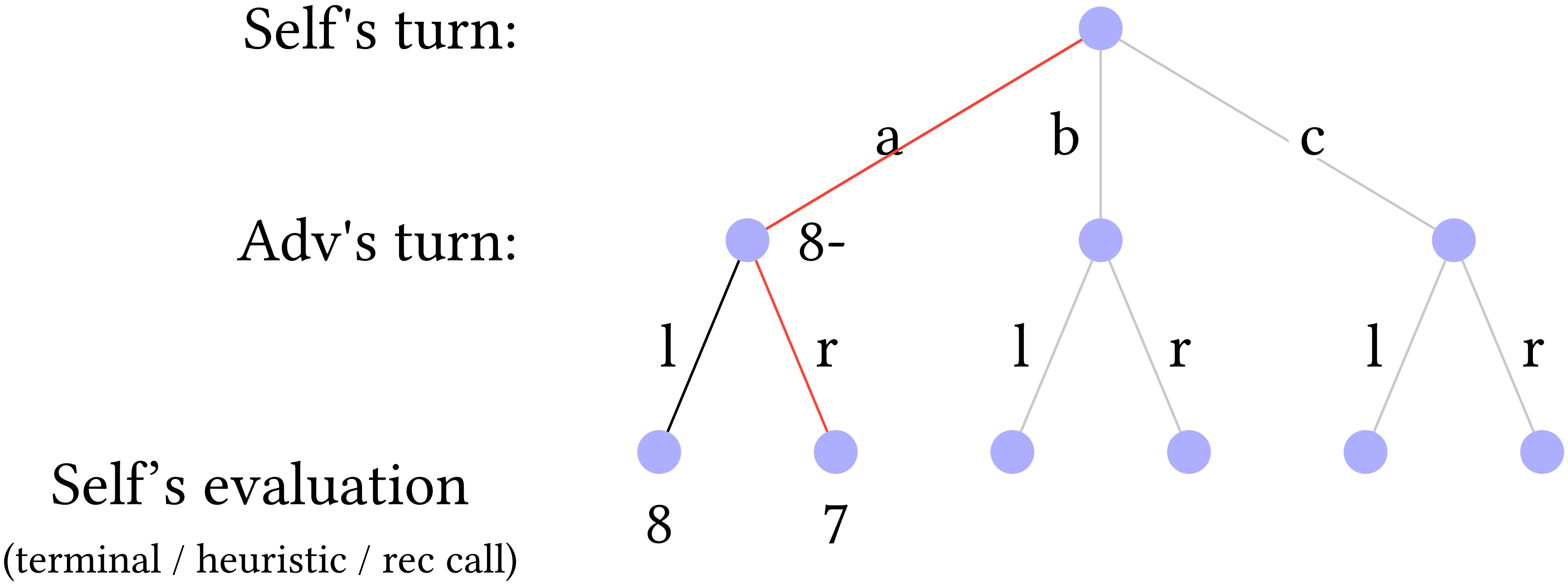
Adv's turn:

Self's evaluation

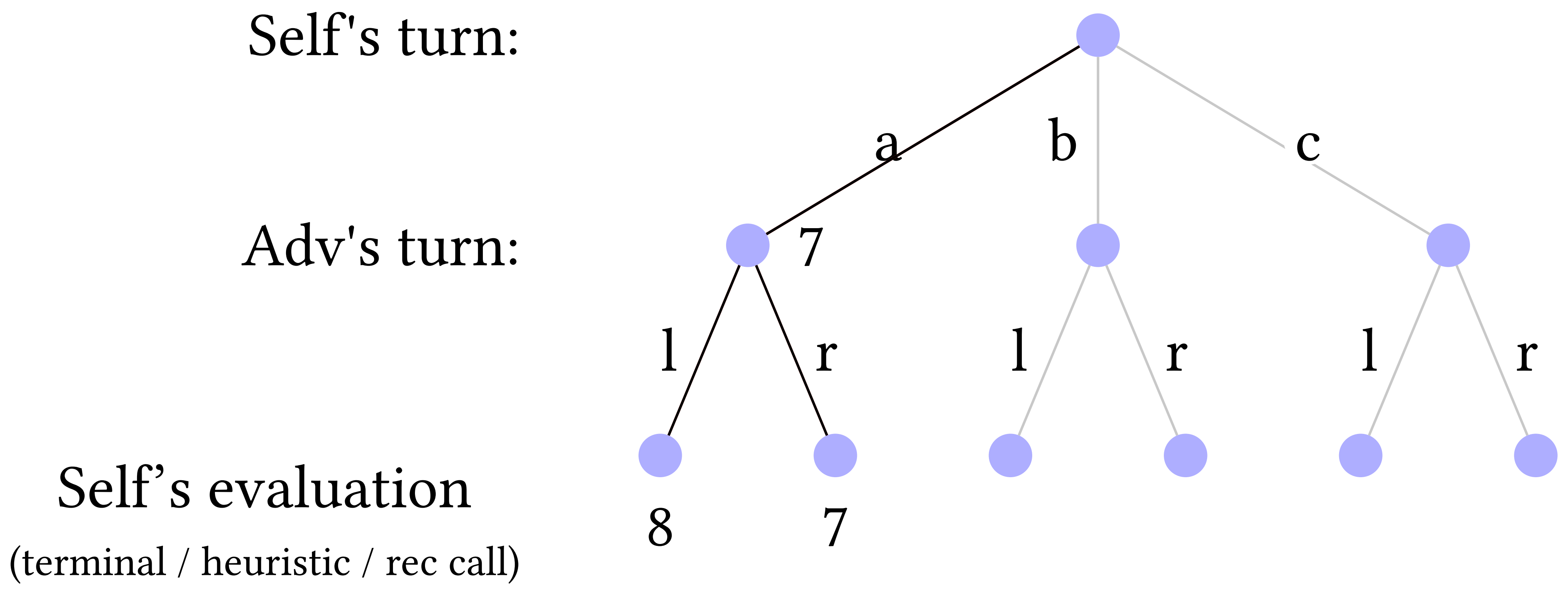
(terminal / heuristic / rec call)



Alpha-Beta game tree



Alpha-Beta game tree



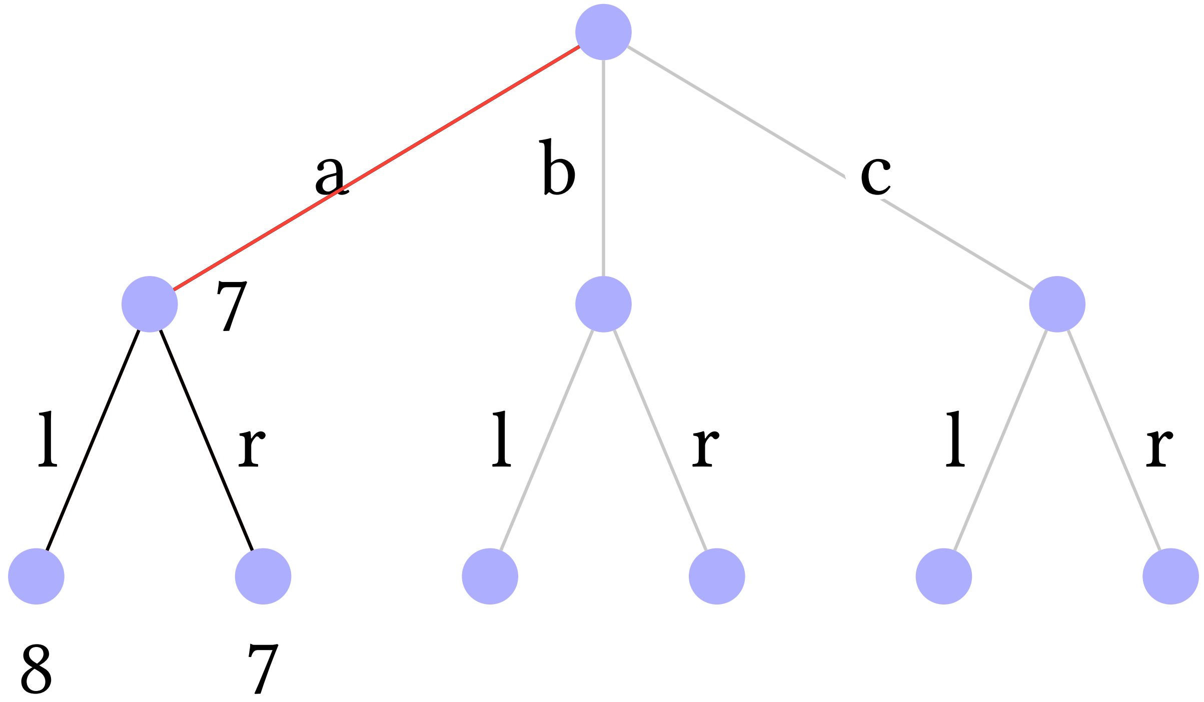
Alpha-Beta game tree

Self's turn:

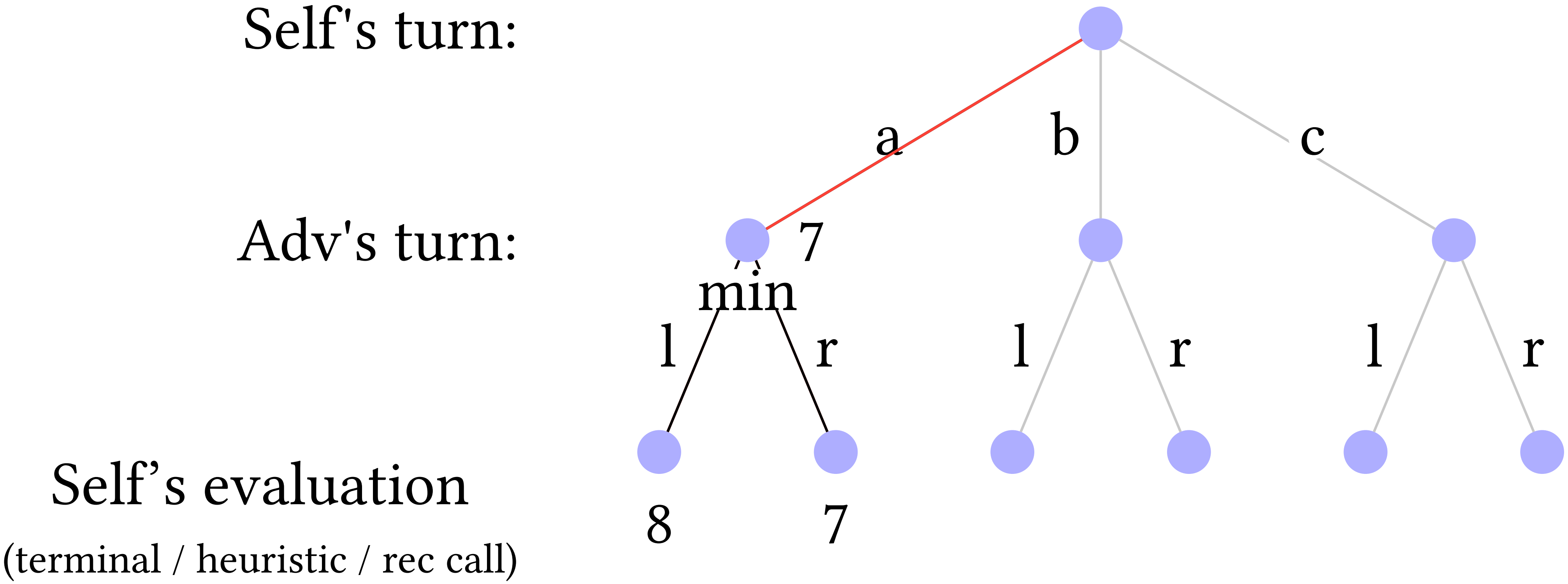
Adv's turn:

Self's evaluation

(terminal / heuristic / rec call)



Alpha-Beta game tree



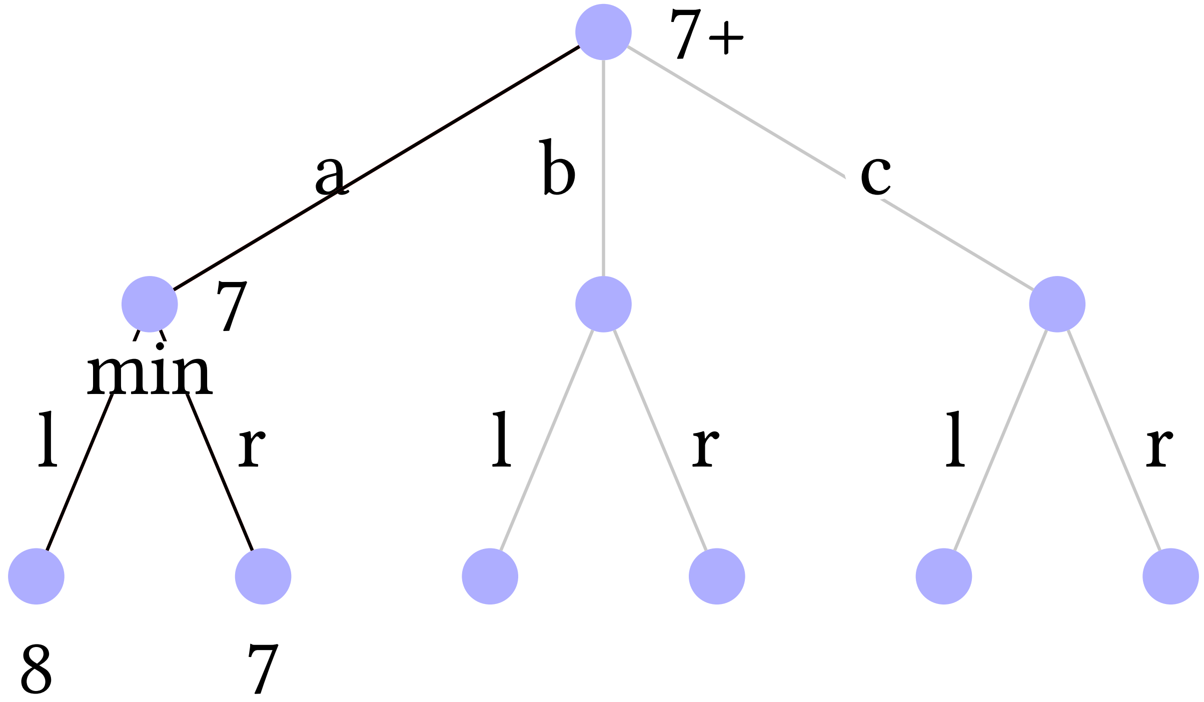
Alpha-Beta game tree

Self's turn:

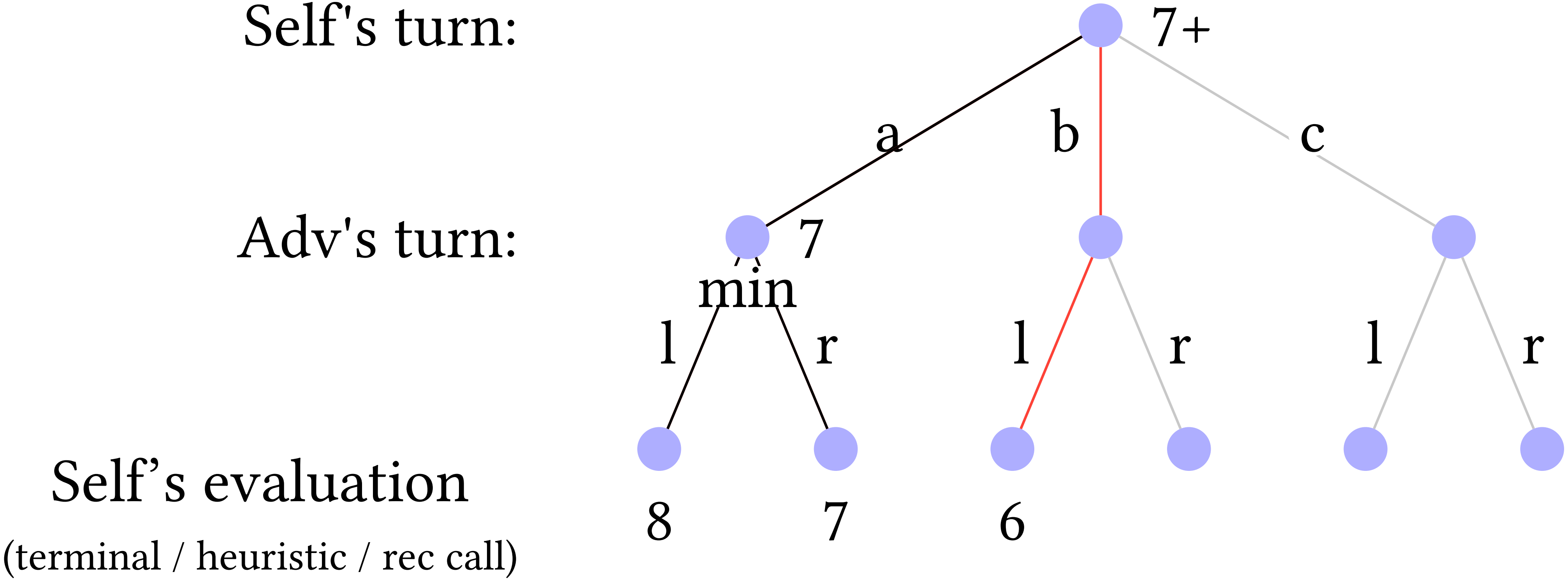
Adv's turn:

Self's evaluation

(terminal / heuristic / rec call)



Alpha-Beta game tree



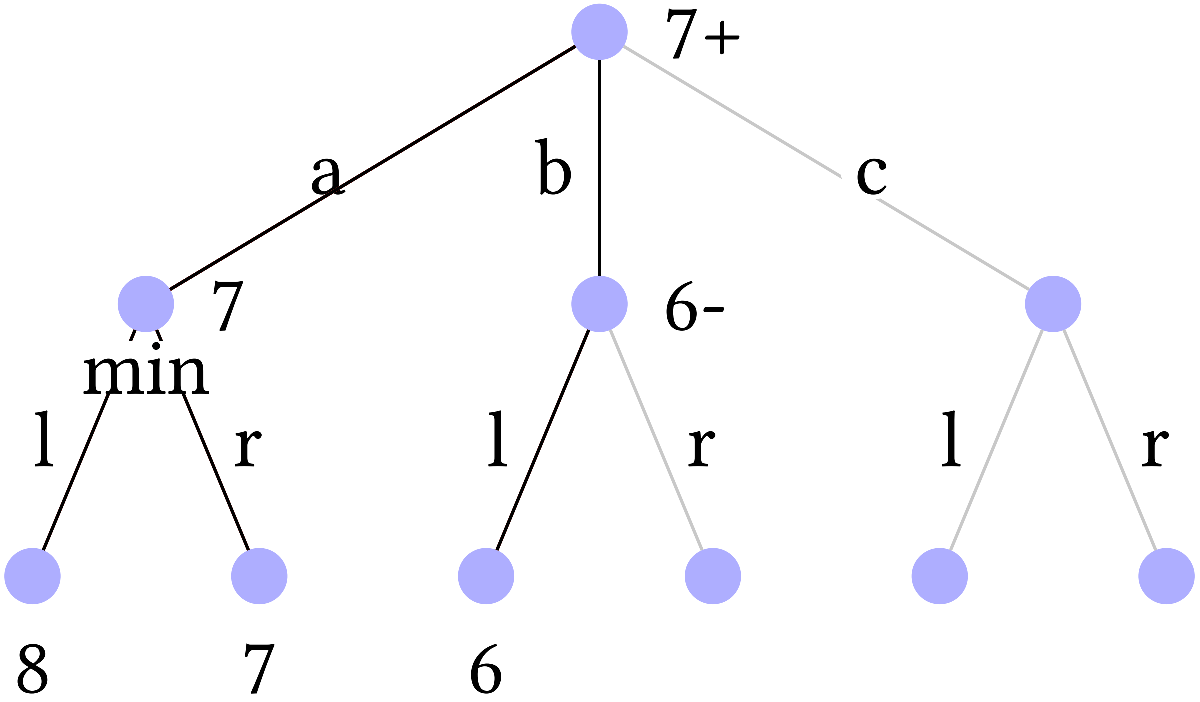
Alpha-Beta game tree

Self's turn:

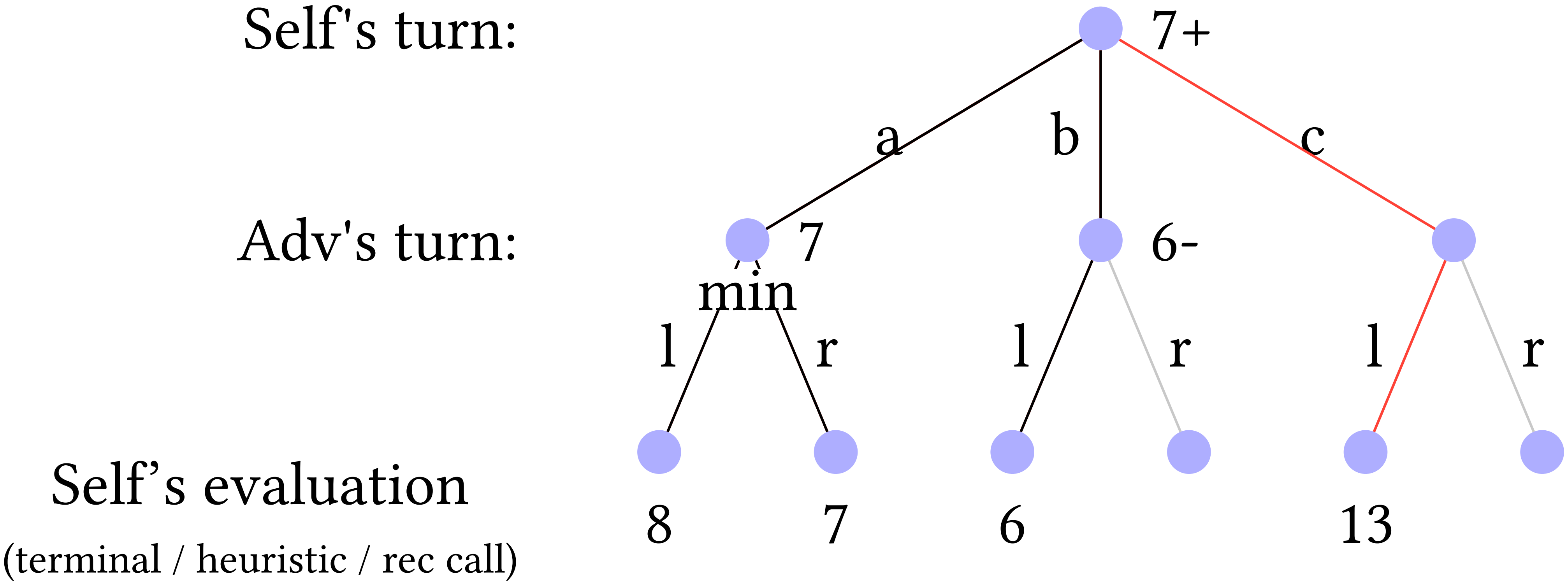
Adv's turn:

Self’s evaluation

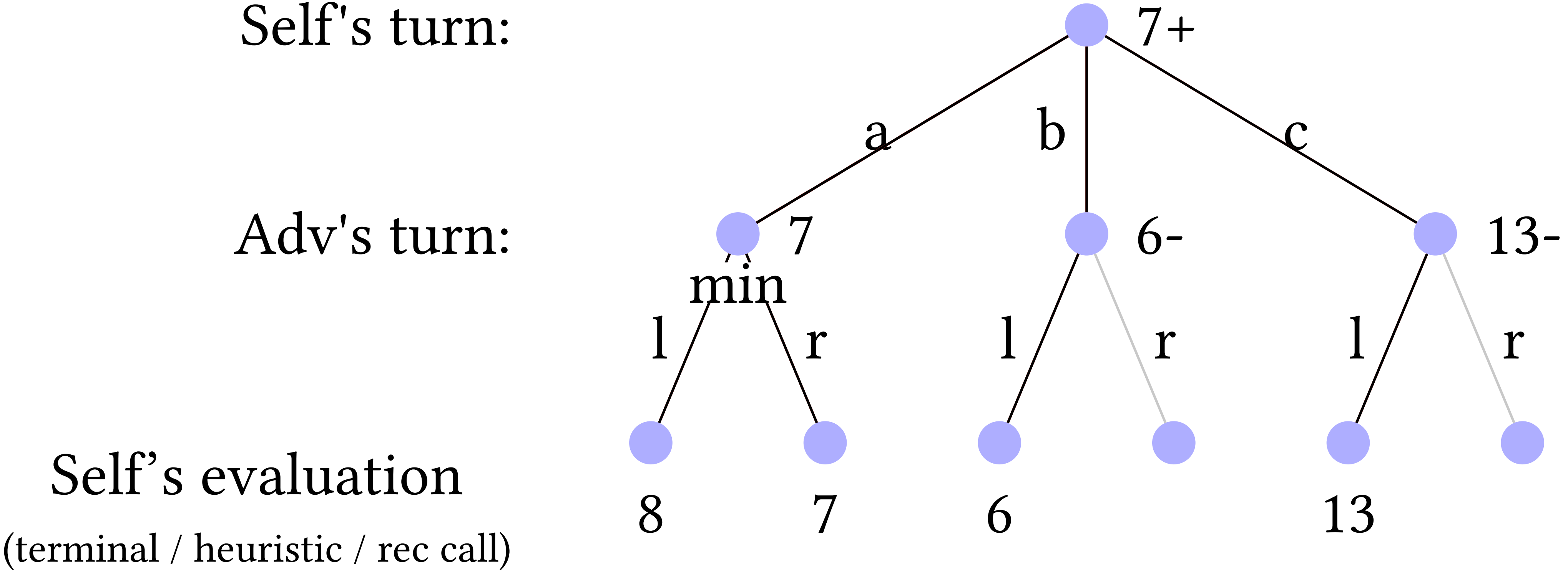
(terminal / heuristic / rec call)



Alpha-Beta game tree



Alpha-Beta game tree



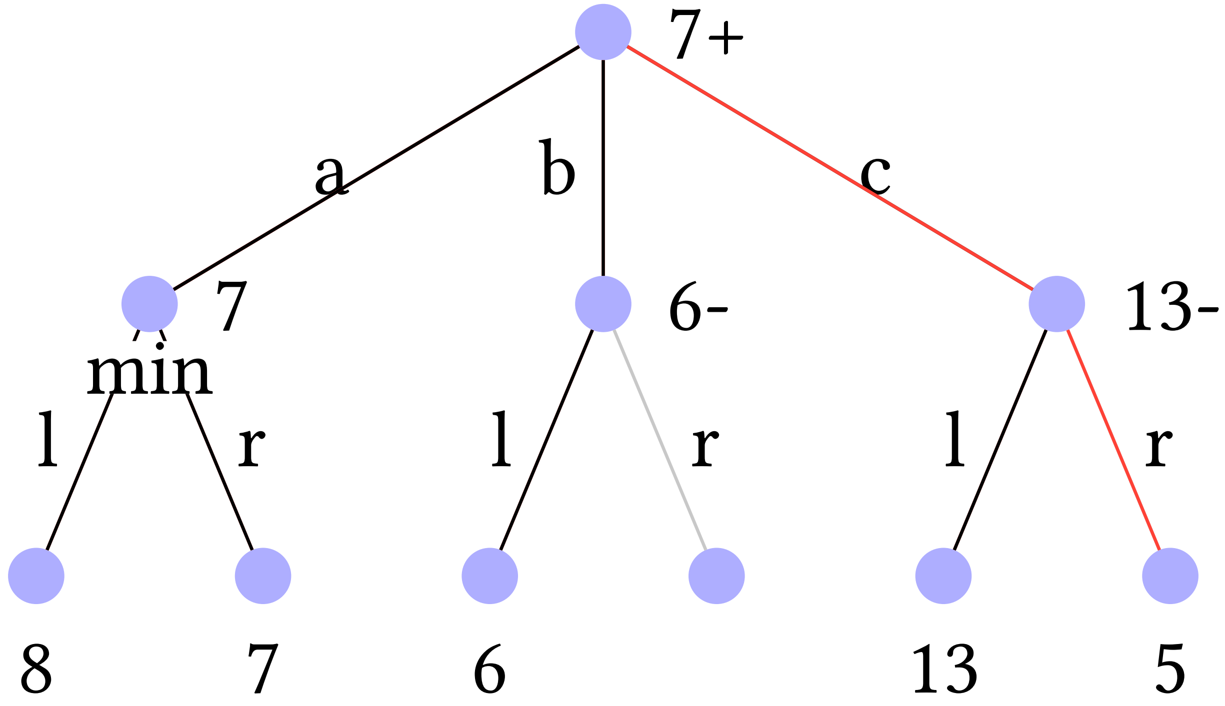
Alpha-Beta game tree

Self's turn:

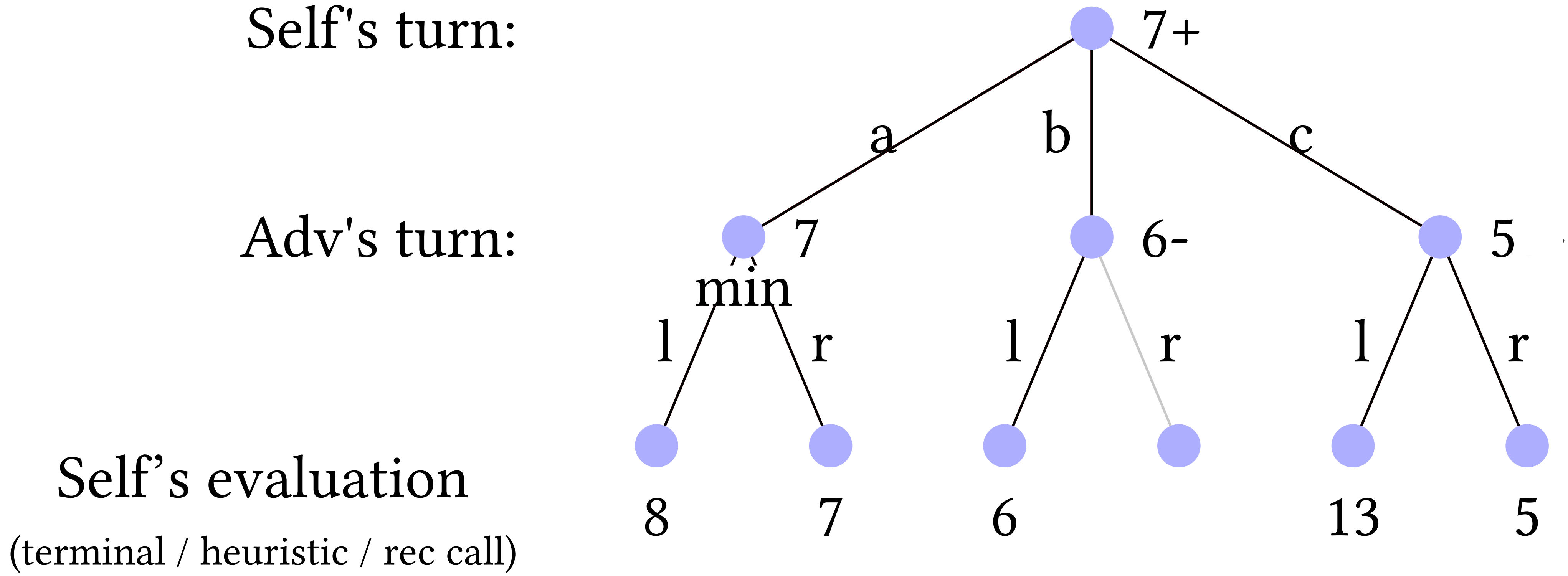
Adv's turn:

Self's evaluation

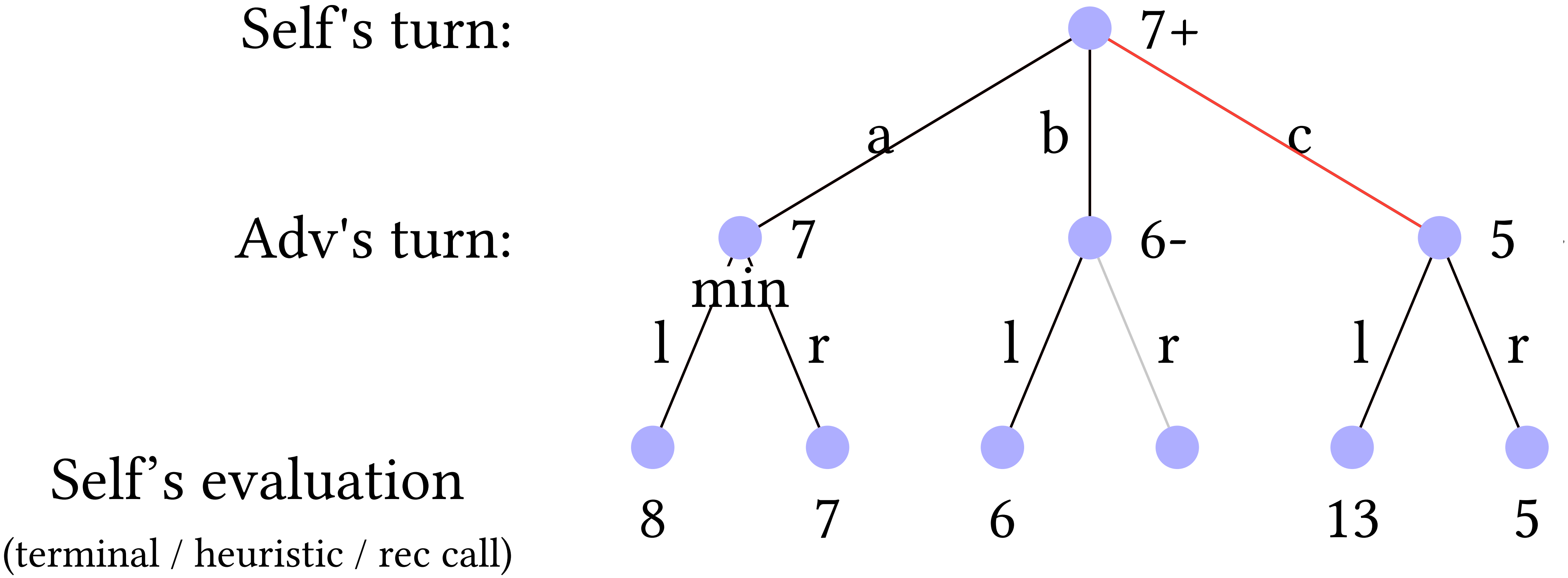
(terminal / heuristic / rec call)



Alpha-Beta game tree



Alpha-Beta game tree



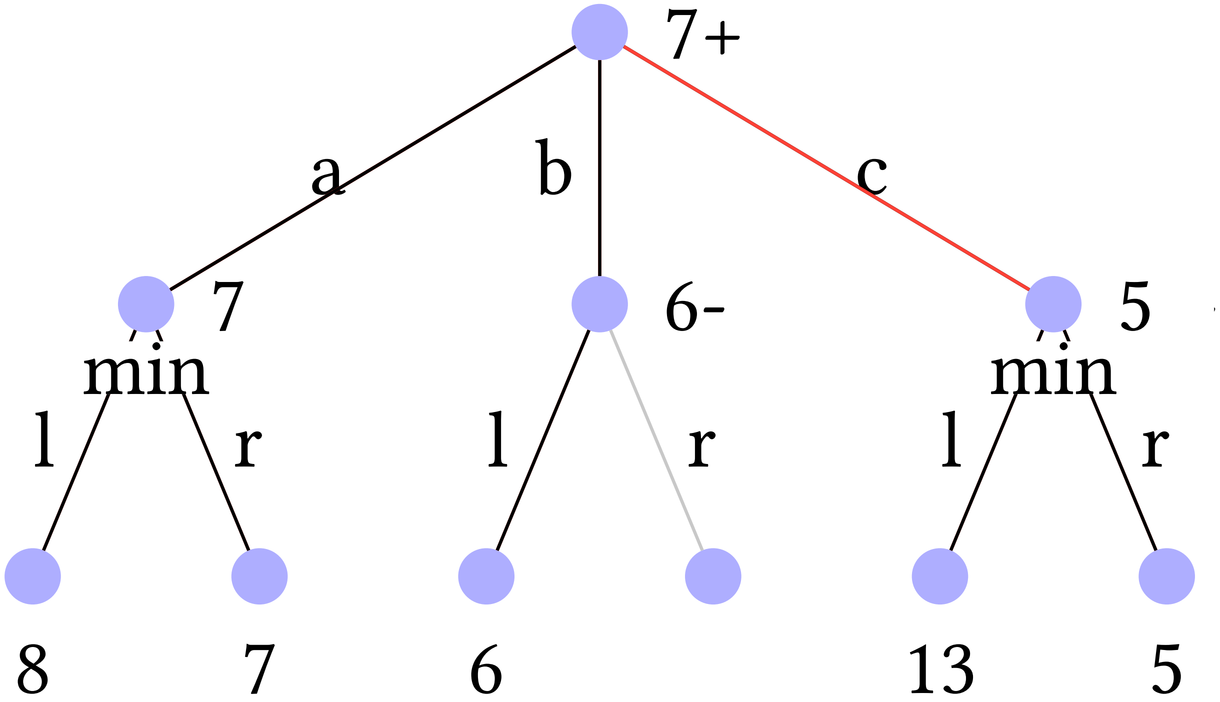
Alpha-Beta game tree

Self's turn:

Adv's turn:

Self’s evaluation

(terminal / heuristic / rec call)



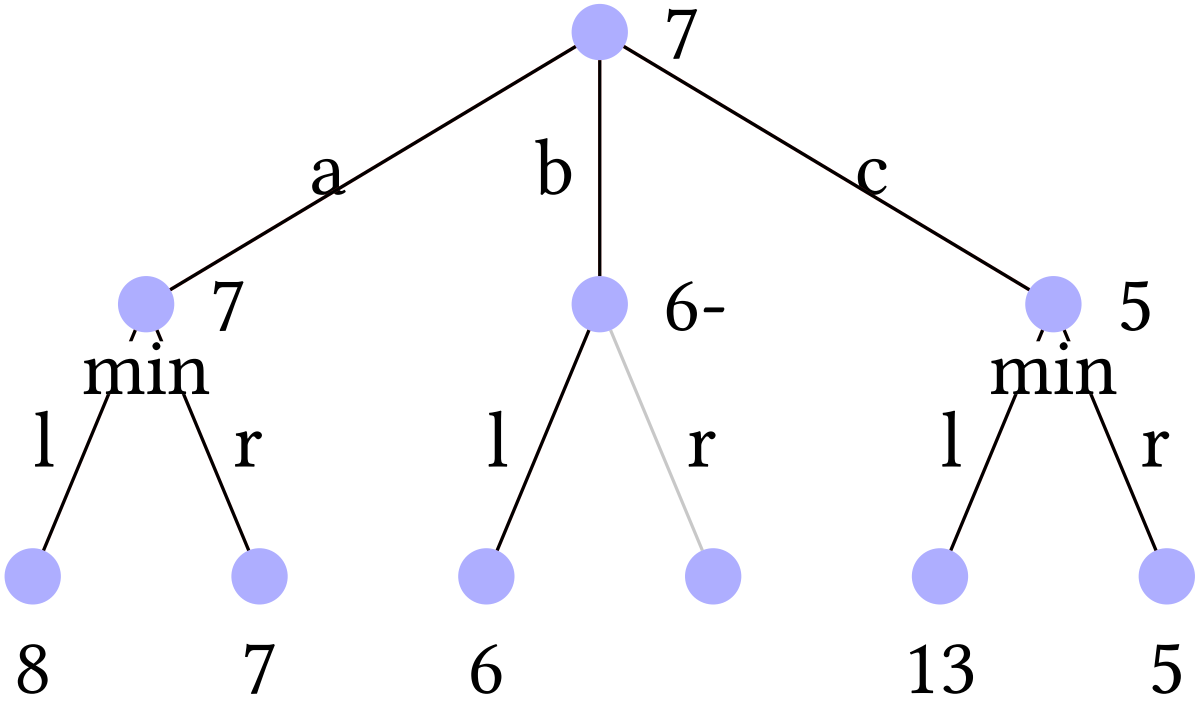
Alpha-Beta game tree

Self's turn:

Adv's turn:

Self's evaluation

(terminal / heuristic / rec call)



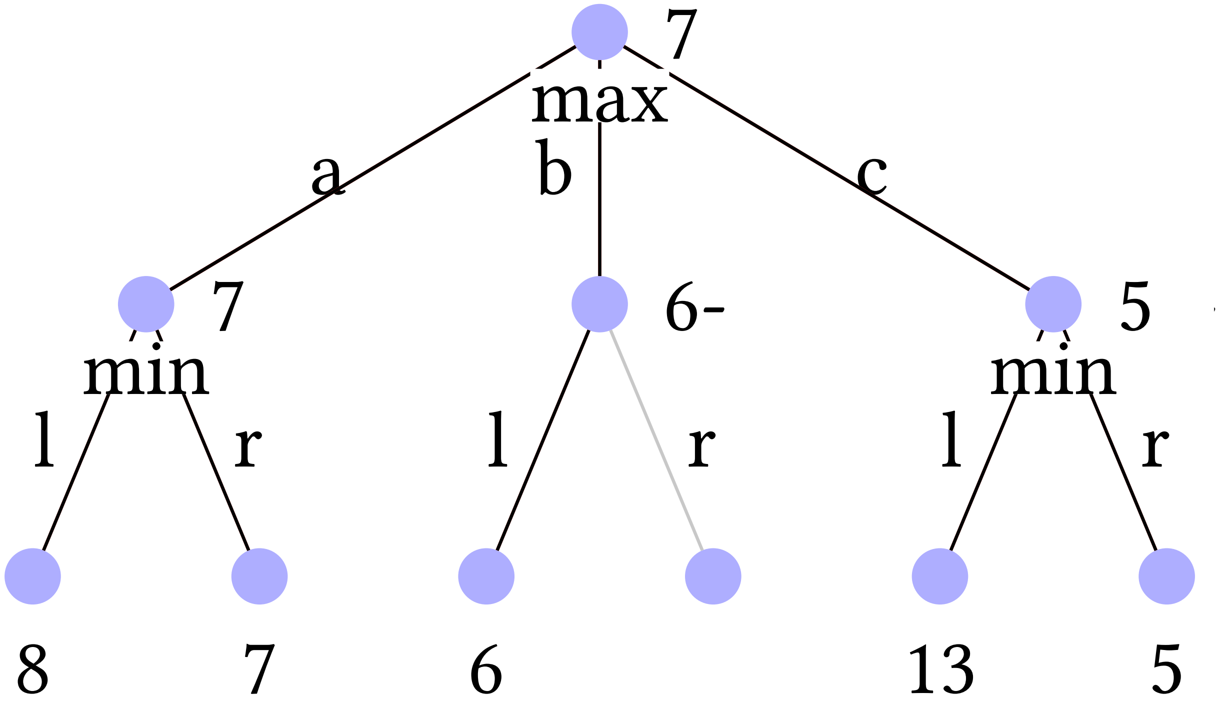
Alpha-Beta game tree

Self's turn:

Adv's turn:

Self's evaluation

(terminal / heuristic / rec call)



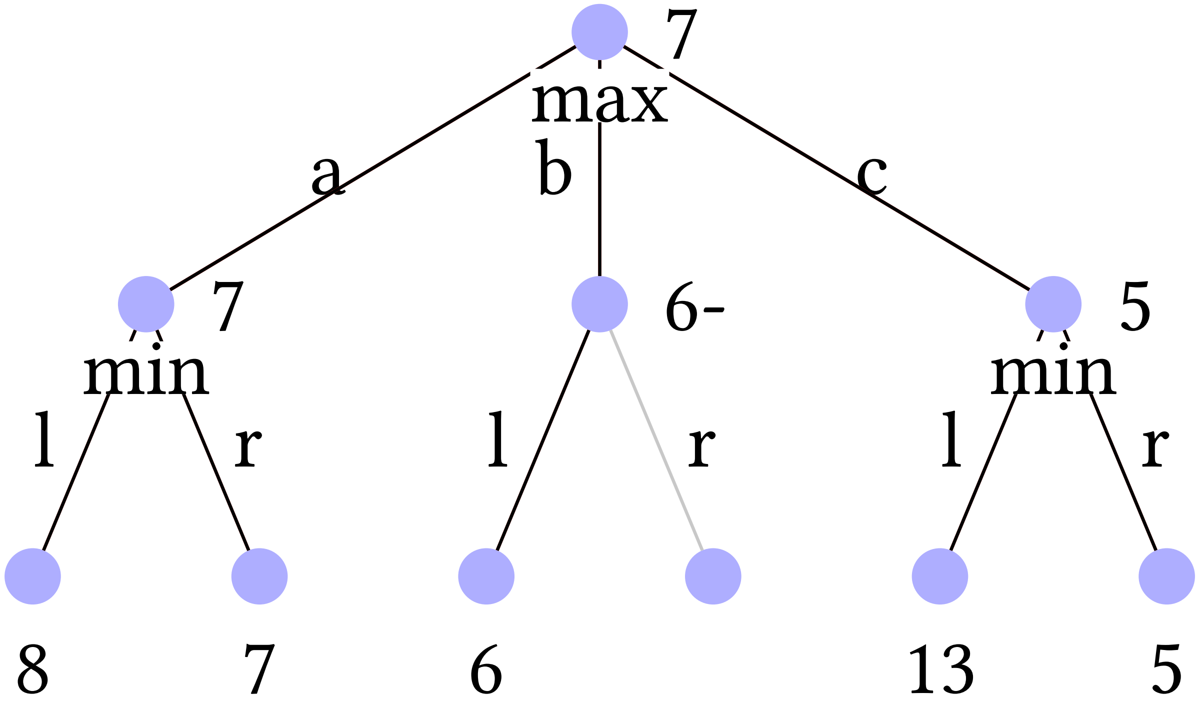
Alpha-Beta game tree

Self's turn:

Adv's turn:

Self's evaluation

(terminal / heuristic / rec call)



Alpha-beta pruning

Alpha-Beta pruning:

- maintains upper/lower bounds on utility
- **ignores branches** that cannot impact final result
  - action r after b cannot help us
- remains **optimal** while evaluating fewer nodes
- is sensitive to order

MINIMAX with cutoff

Even with Alpha-Beta pruning, one cannot expect full exploration of the game tree[1]

Solution:

- define a *cutoff* condition when to stop searching
- use a heuristic evaluation function to estimate utility

[1] For american checkers, done but took 20 years on super computers. (Checkers is solved. J. Schaeffer, 2007, 🔗)

Minimax with cutoff

$$\text{Minimax}(s) := \begin{cases} \text{SelfUtility}(s) & \text{if IsTerminal}(s) \\ \text{Eval}(s) & \text{if IsCutoff}(s) \\ \max_{a \in \text{ SelfActions}(s)} \text{Minimax}(\text{Result}(s, a)) & \text{if ToMove}(s) = \text{Self} \\ \min_{a \in \text{ AdvActions}(s)} \text{Minimax}(\text{Result}(s, a)) & \text{if ToMove}(s) = \text{Adv} \end{cases}$$