Combining Deep Reinforcement Learning and Search for Imperfect-Information Games

Noam Brown*, Anton Bakhtin*, Adam Lerer, Qucheng Gong

Facebook AI Research

***Equal Contribution**

FACEBOOK AI

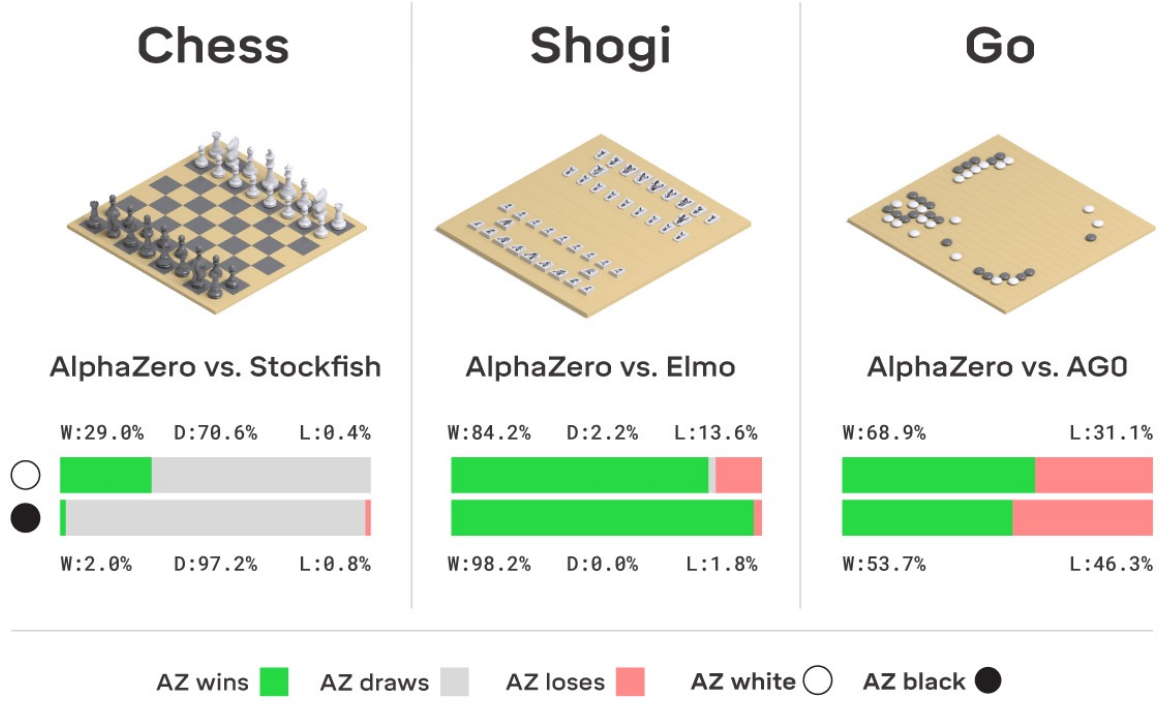
AlphaGo

- Milestone AI achievement
- Algorithm was specific to Go:
  - Used human data
  - Used expert features

AlphaZero

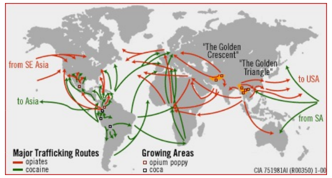
- A single algorithm that plays Chess, Go, and Shogi

- Very general technique:
  - No human data
  - No expert features



- Limited to **perfect-information** games

Imperfect-Information Games



Perfect-Information Games

No-Limit Texas Hold’em Poker

- Long-standing challenge problem in AI and game theory
- 2017: AI surpasses top humans in two-player no-limit hold’em
- 2019: AI surpasses top humans in six-player no-limit hold’em
- Techniques used in poker AIs have been very different from AlphaGo/AlphaZero

ReBeL

- Can a single algorithm work for both perfect- and imperfect-information games?

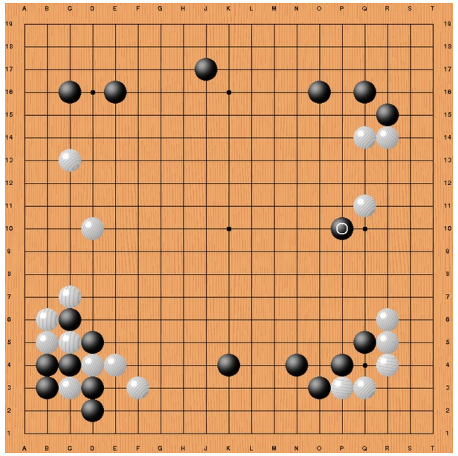
- **ReBeL** (Recursive Belief-based Learning)
  - Provably converges to Nash in two-player zero-sum games
  - Superhuman in two-player no-limit hold’em poker
  - Uses far less domain knowledge than prior poker bots
  - In perfect-info games, ReBeL reduces to an algorithm similar to AlphaZero

A Simplified Overview of AlphaZero

What is a "state" in a game?

- A **state** must be a **sufficient statistic**
  - Must contain all relevant info needed to compute the optimal next move

- Board configuration alone might not be enough (e.g., ko rule in Go)
  - AlphaZero uses last 8 board configurations



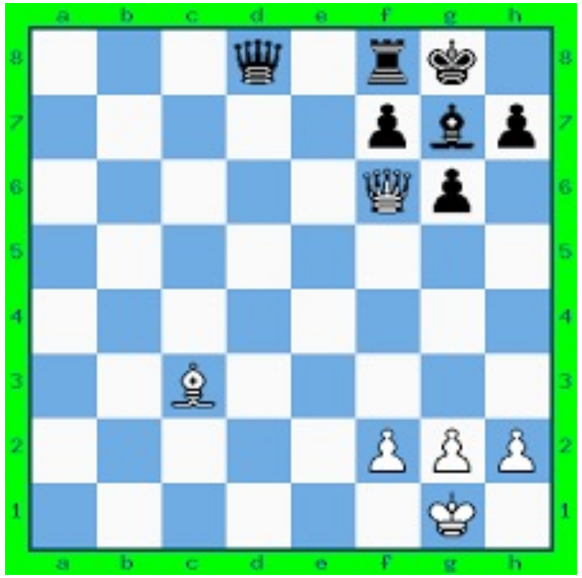
- **Worst case:** "state" in a perfect-info game is the **entire sequence of actions**

Search in Perfect-Information Games

- In perfect-information games, the **value of a state** is the **unique** value resulting from both players playing optimally from that point forward

- A **value network** takes a state as input and outputs an estimate of the state value

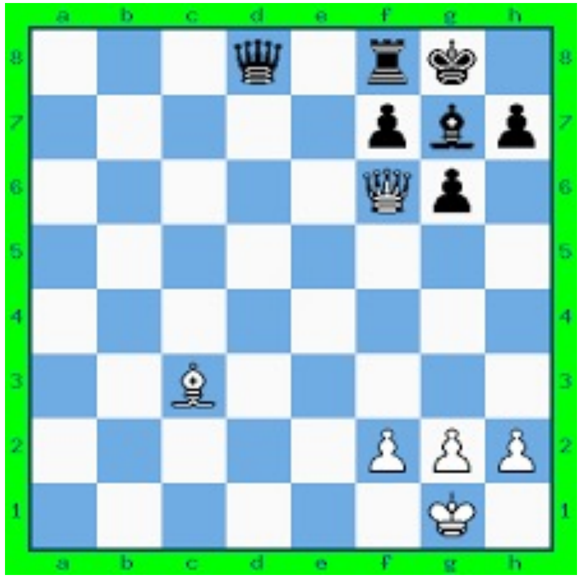
$f_{white}($



$) = 1$

Search in Perfect-Information Games

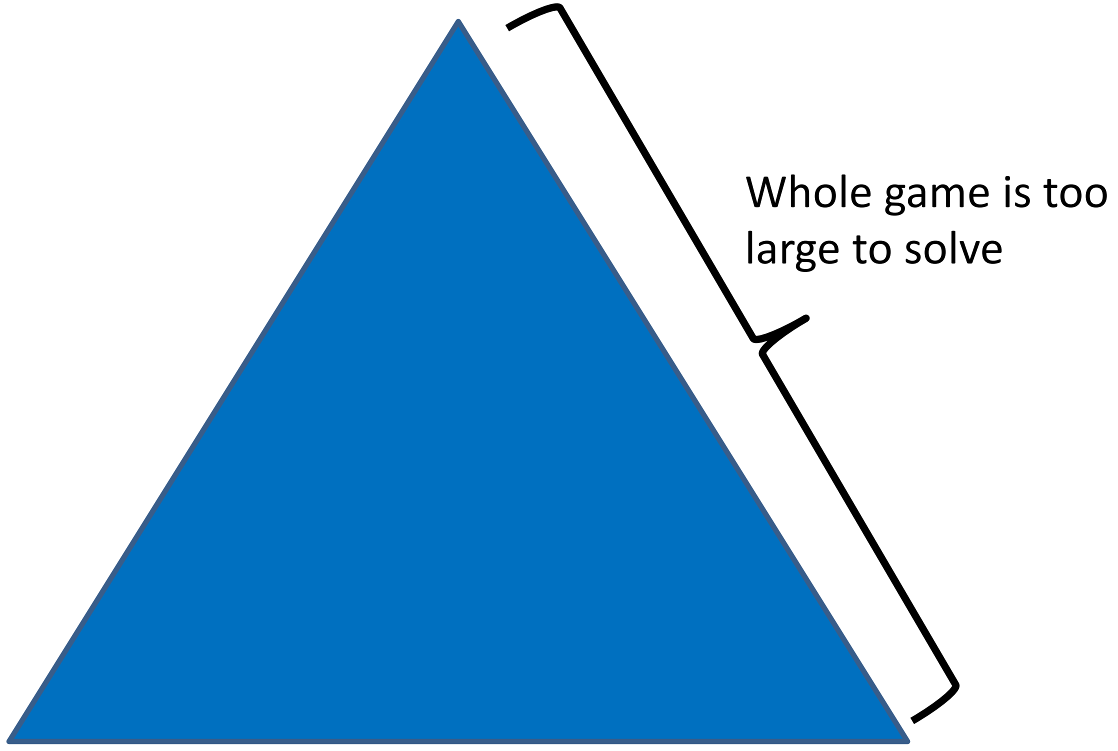
- Where does the value network come from?
  - It can be a handcrafted heuristic function [Deep Blue]
  - It can be learned by training on expert human games [AlphaGo]
  - It can be learned through self play [AlphaZero]

$f_{white}($  $) = 1$

Search in Perfect-Information Games

- In principle, backward induction alone can solve Chess

- But this would be far too expensive in practice

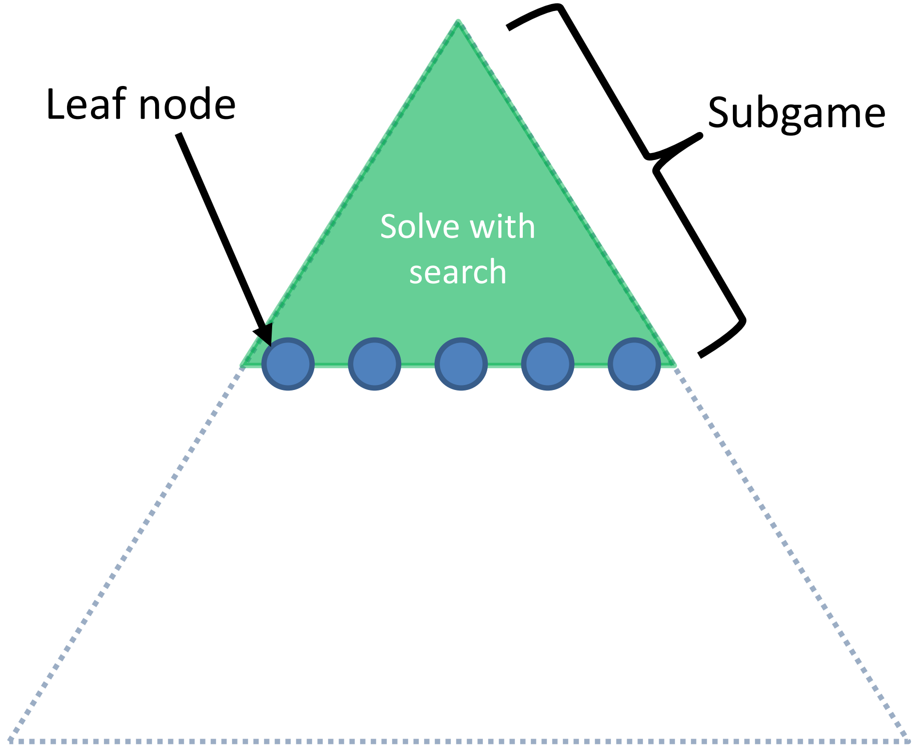


Search in Perfect-Information Games

- Instead, chess AI's do **search:**
  1. Look ~5 moves ahead
  2. Estimate those state values using the value network
  3. Do backward induction using those state values (ignore the game below those states)

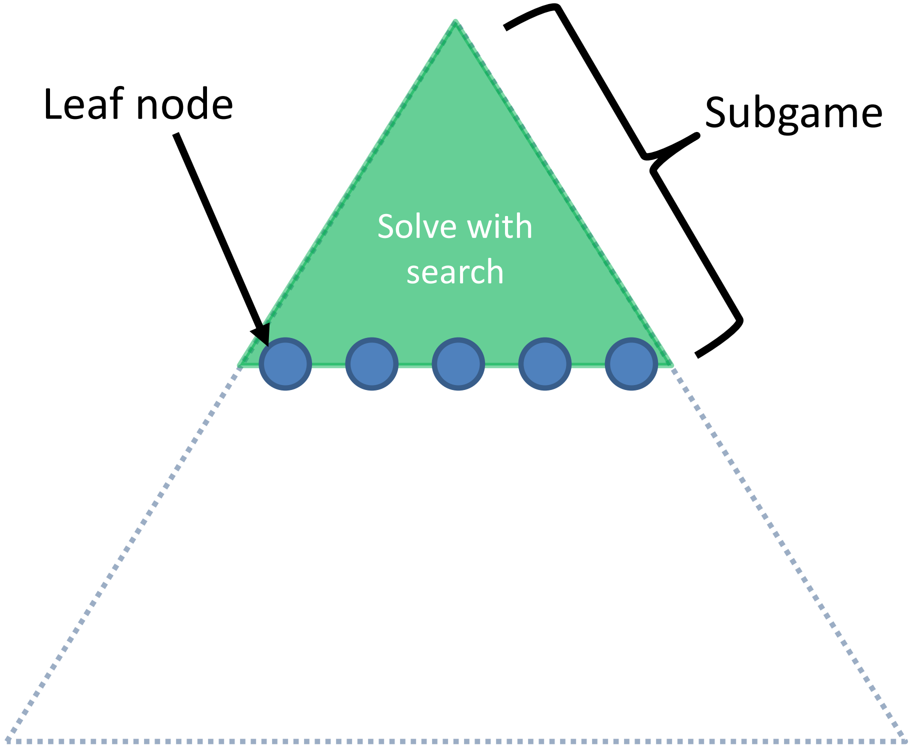
- In other words, solve a **subgame**

- If the value network is perfect, this computes the optimal action



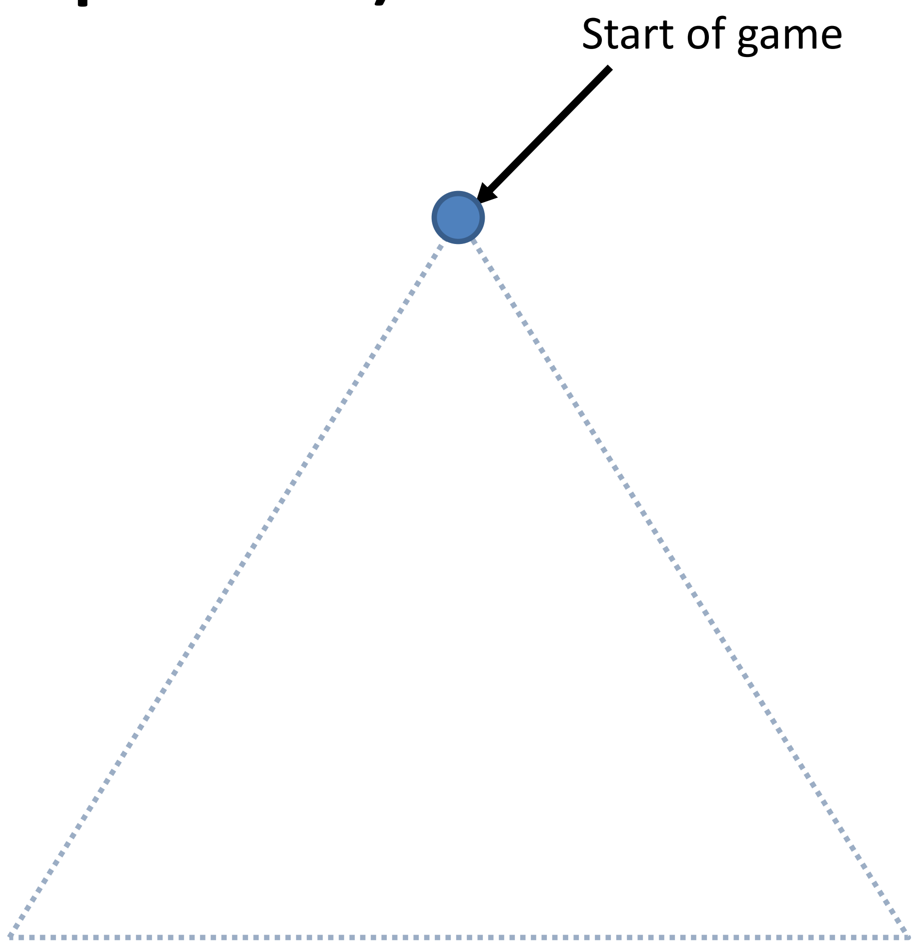
Search in Perfect-Information Games

- In AlphaZero, the subgame grows in size as it is solved
- In principle, ReBeL can do the same
- For simplicity, we assume subgames are **fixed** in size
  - Imagine subgames as containing every state reachable within 5 actions



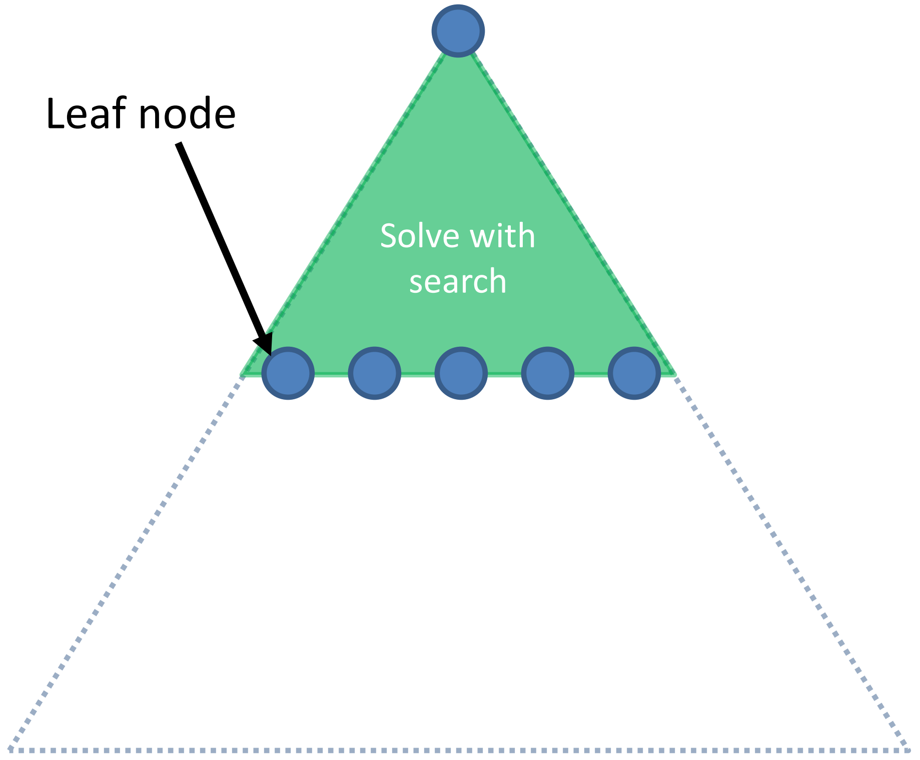
AlphaZero (Simplified)

- Whenever an agent acts, generate a subgame and solve it
  - Set leaf node values based on value net



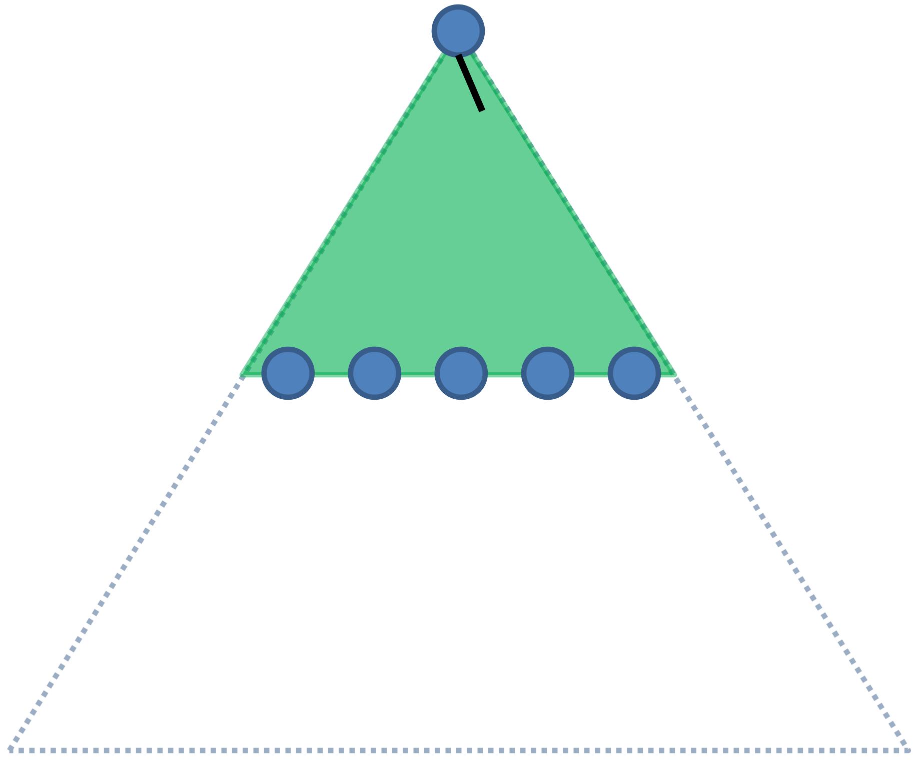
AlphaZero (Simplified)

- Whenever an agent acts, generate a subgame and solve it
  - Set leaf node values based on value net



AlphaZero (Simplified)

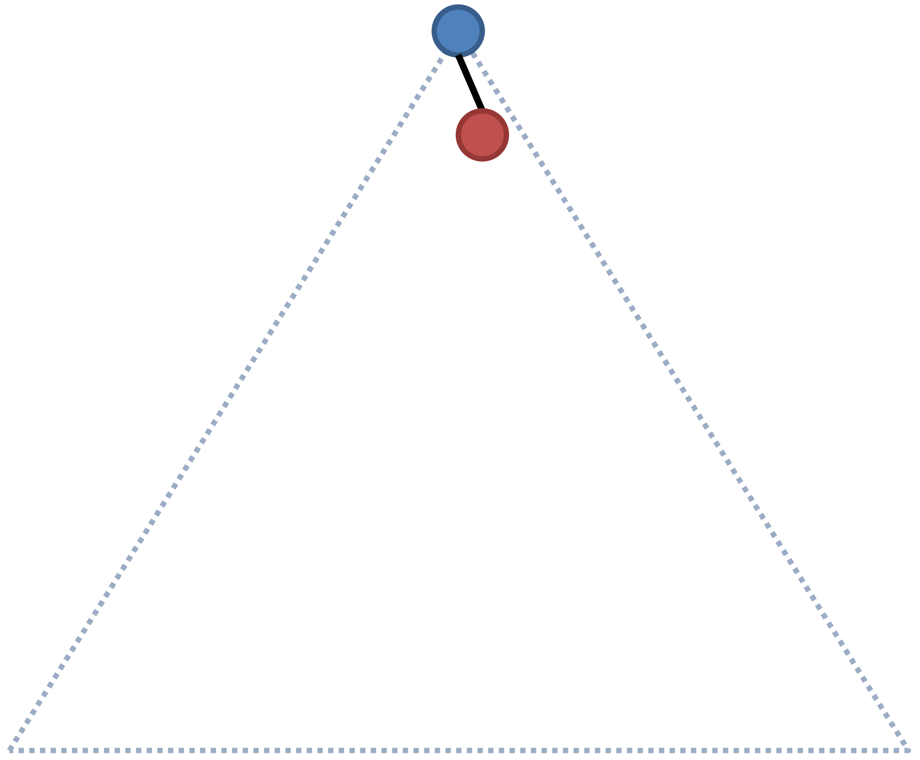
- Whenever an agent acts, generate a subgame and solve it
  - Set leaf node values based on value net
- Choose next action based on solution to subgame



AlphaZero (Simplified)

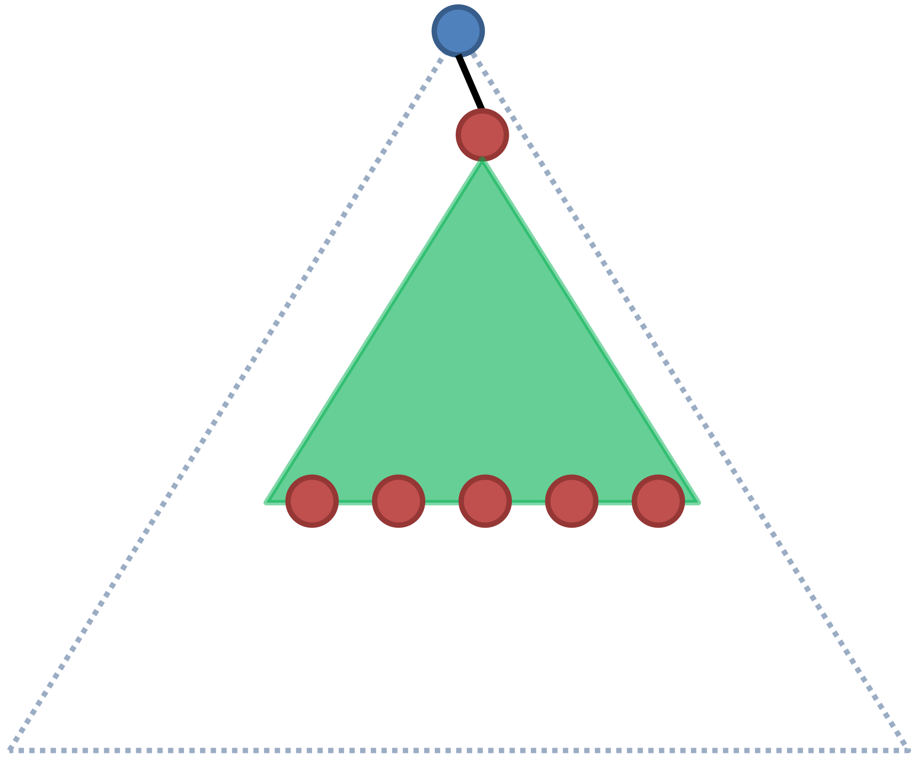
- Whenever an agent acts, generate a subgame and solve it
  - Set leaf node values based on value net

- Choose next action based on solution to subgame



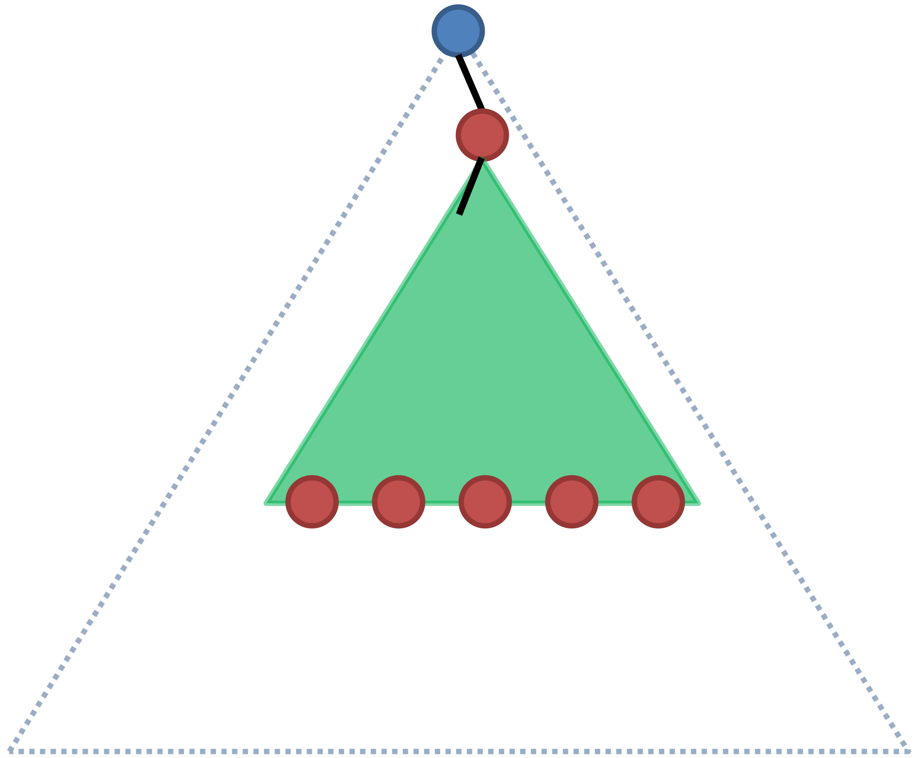
AlphaZero (Simplified)

- Whenever an agent acts, generate a subgame and solve it
  - Set leaf node values based on value net
- Choose next action based on solution to subgame
- Repeat until end of game



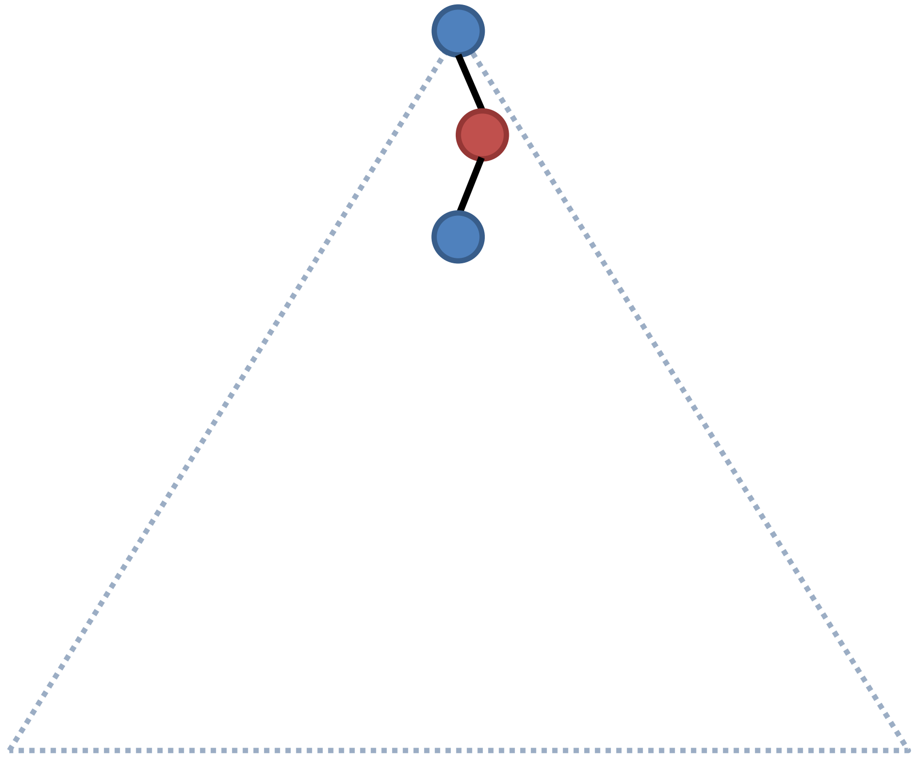
AlphaZero (Simplified)

- Whenever an agent acts, generate a subgame and solve it
  - Set leaf node values based on value net
- Choose next action based on solution to subgame
- Repeat until end of game



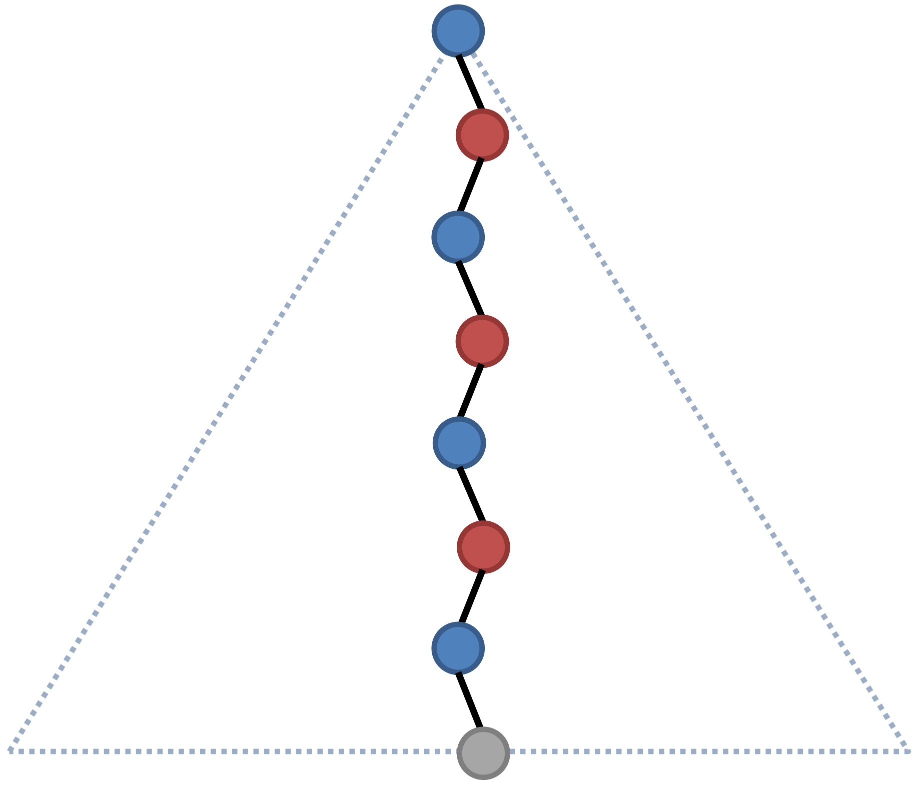
AlphaZero (Simplified)

- Whenever an agent acts, generate a subgame and solve it
  - Set leaf node values based on value net
- Choose next action based on solution to subgame
- Repeat until end of game



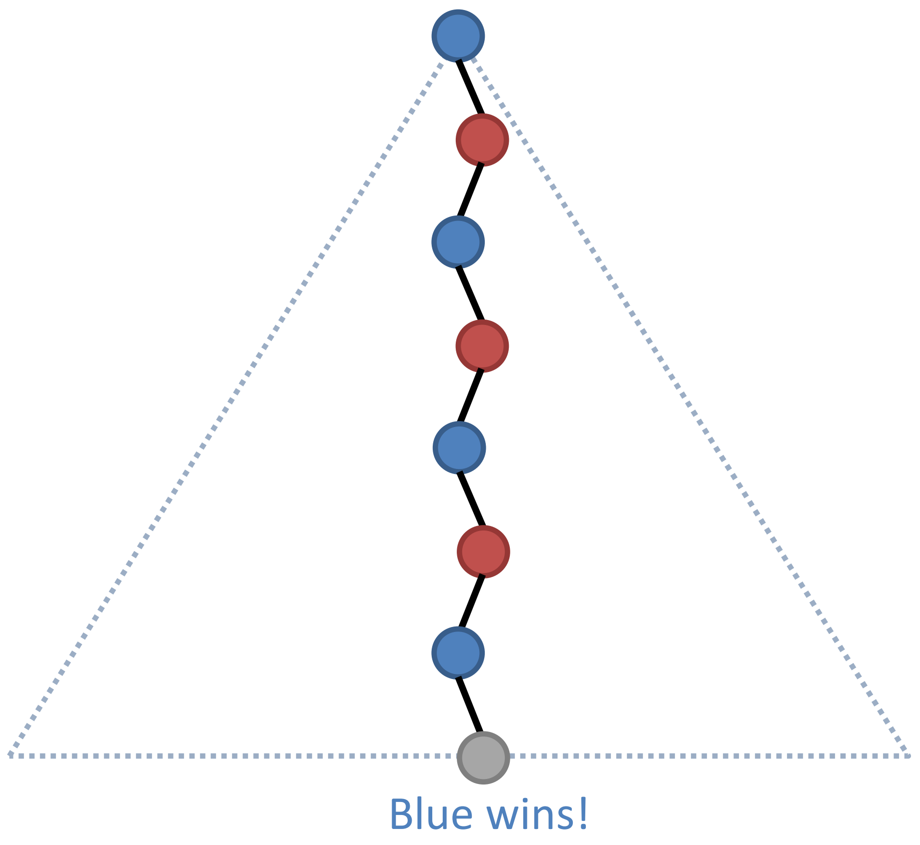
AlphaZero (Simplified)

- Whenever an agent acts, generate a subgame and solve it
  - Set leaf node values based on value net
- Choose next action based on solution to subgame
- Repeat until end of game



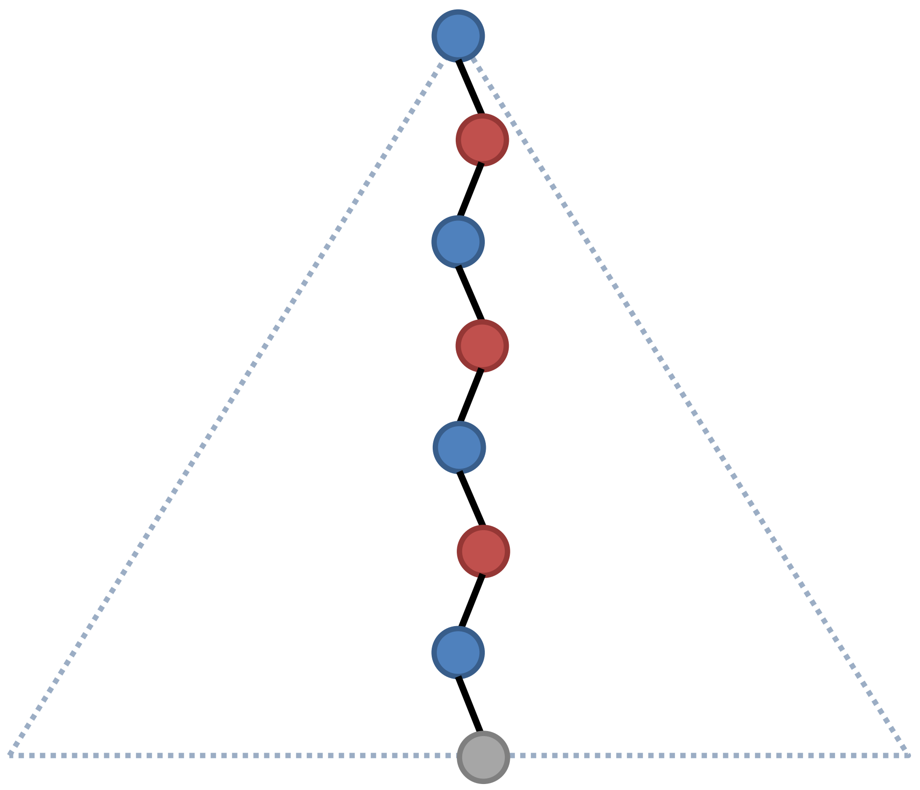
AlphaZero (Simplified)

- Whenever an agent acts, generate a subgame and solve it
  - Set leaf node values based on value net
- Choose next action based on solution to subgame
- Repeat until end of game
- Final value is used as a training example for all encountered states



AlphaZero (Simplified)

- With some random exploration, AlphaZero will eventually encounter every state and learn every state's true value

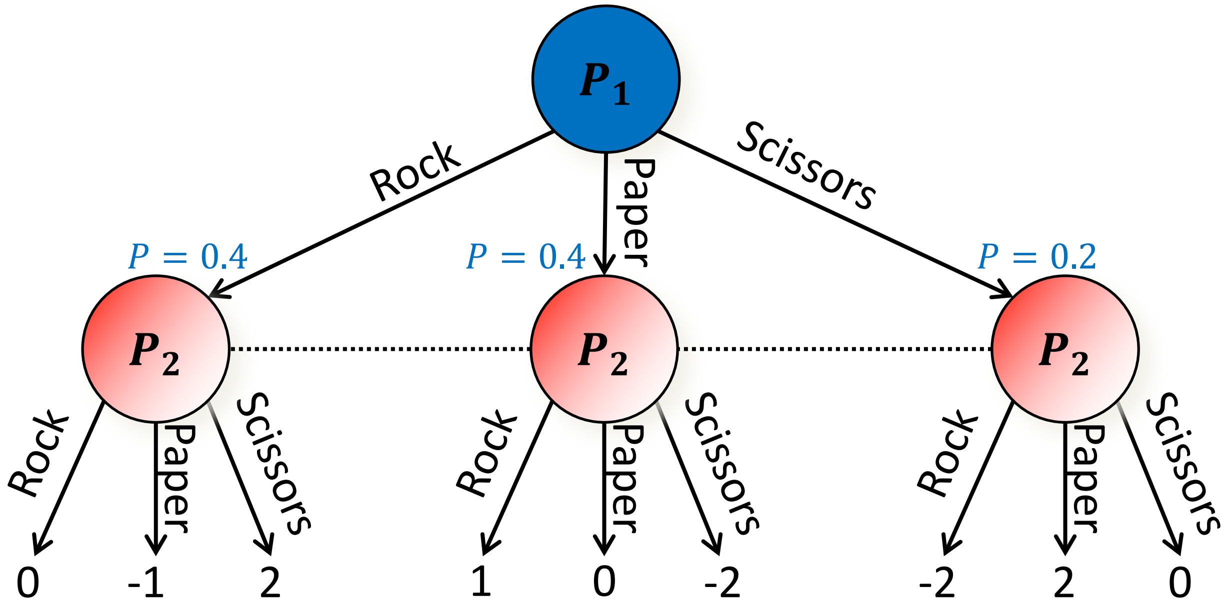


Why doesn't AlphaZero work in imperfect-information games?

Because perfect-info "world states" don't have unique values in imperfect-info games

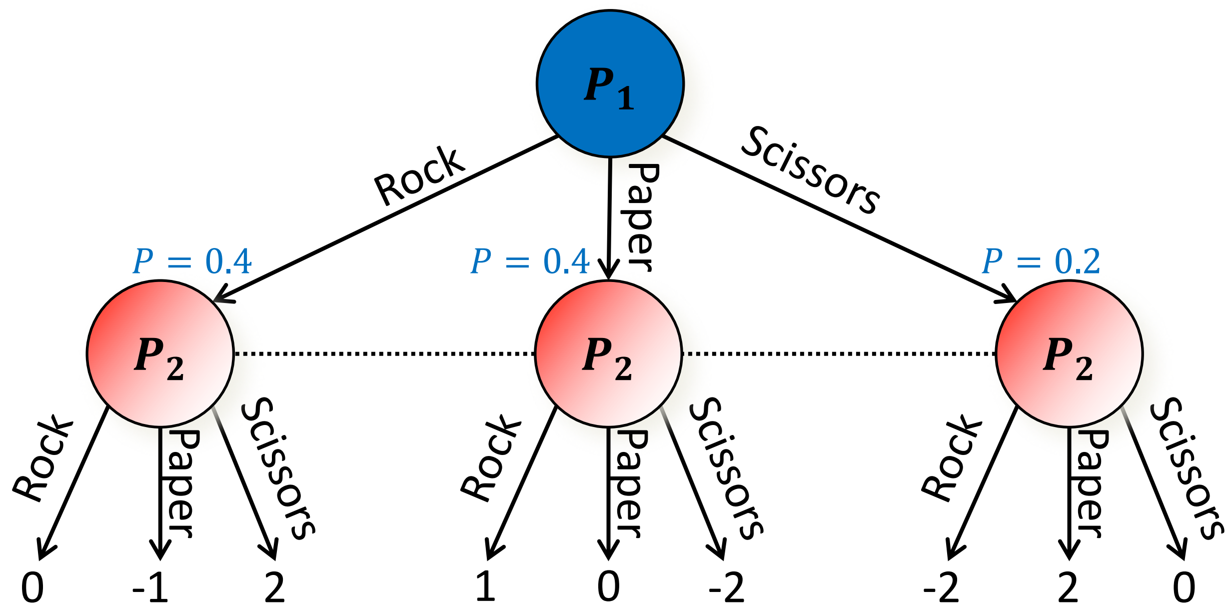
Search in Imperfect-Information Games

Rock-Paper-Scissors+

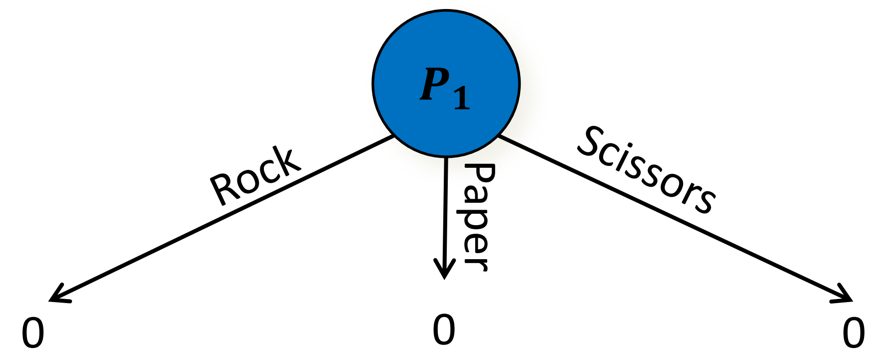


Search in Imperfect-Information Games

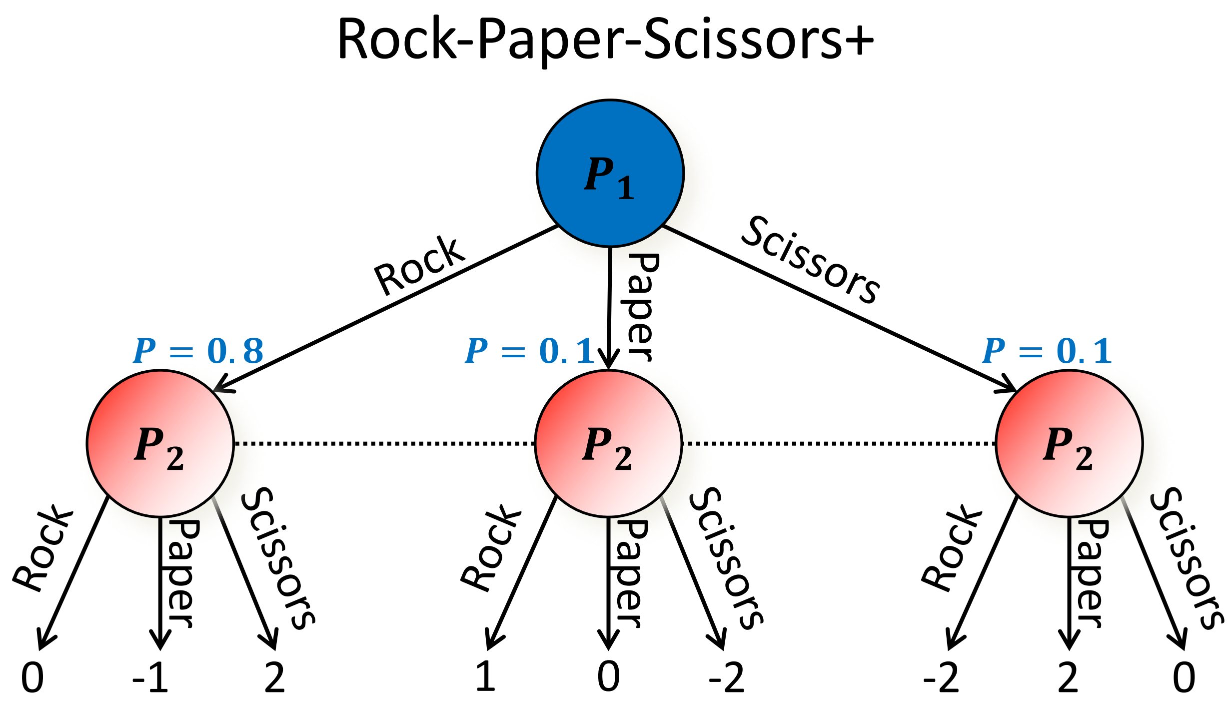
Rock-Paper-Scissors+

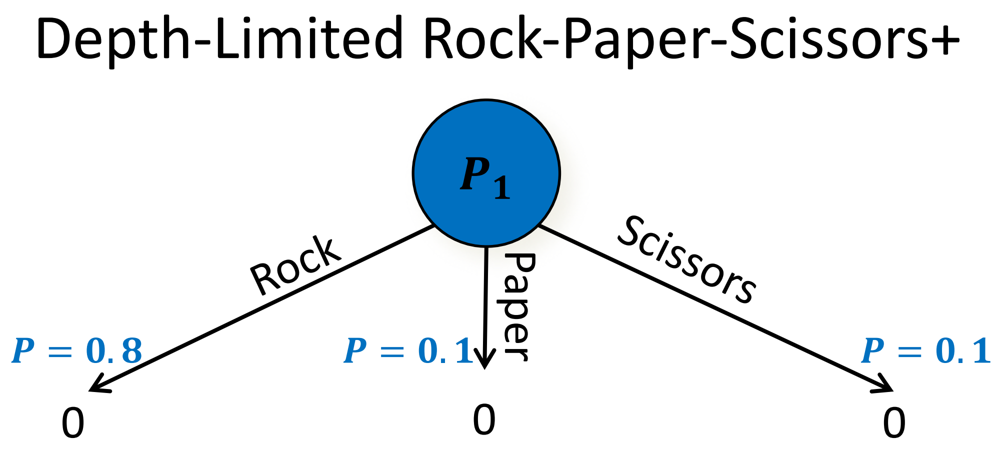


Depth-Limited Rock-Paper-Scissors+



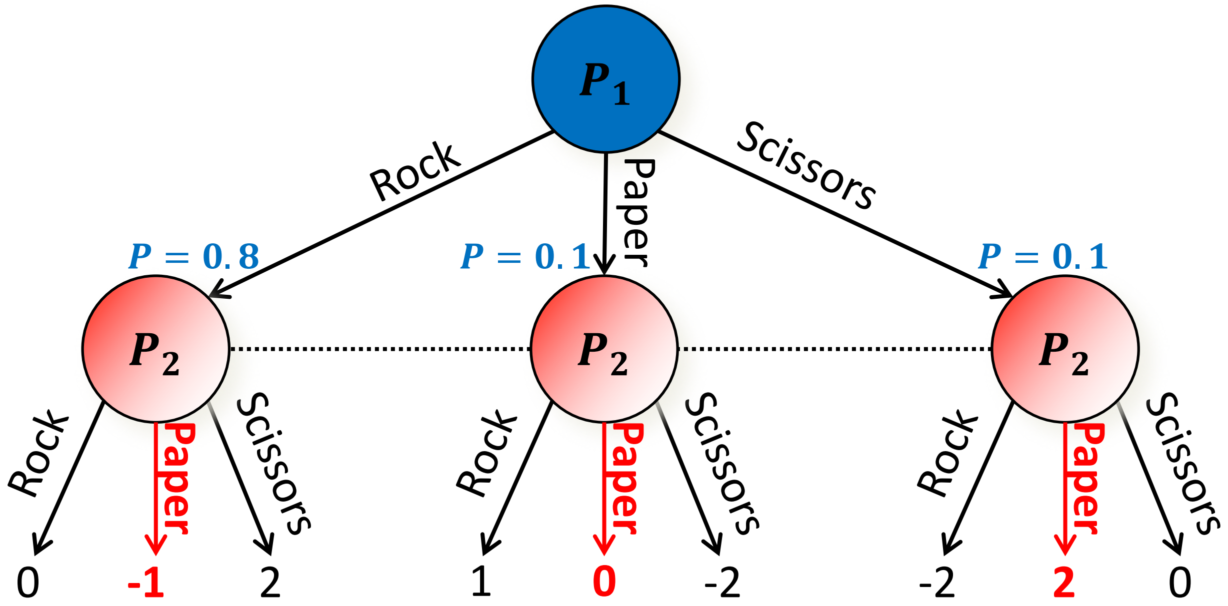
Search in Imperfect-Information Games



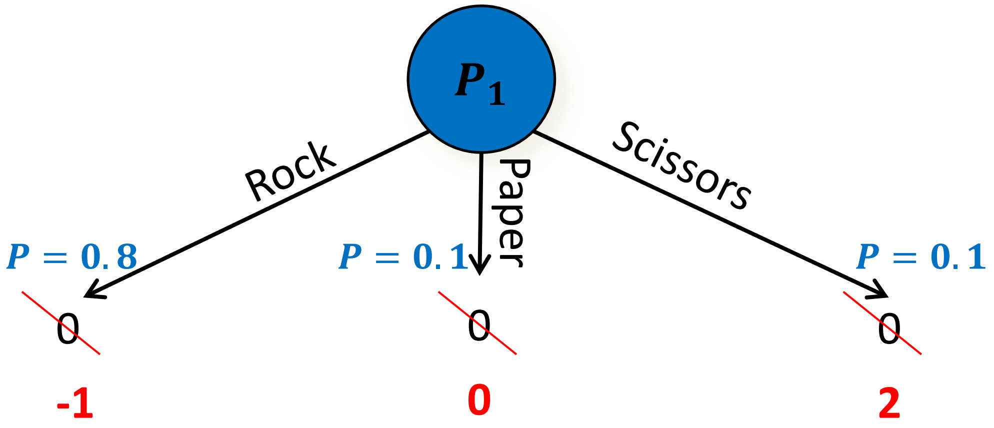


Search in Imperfect-Information Games

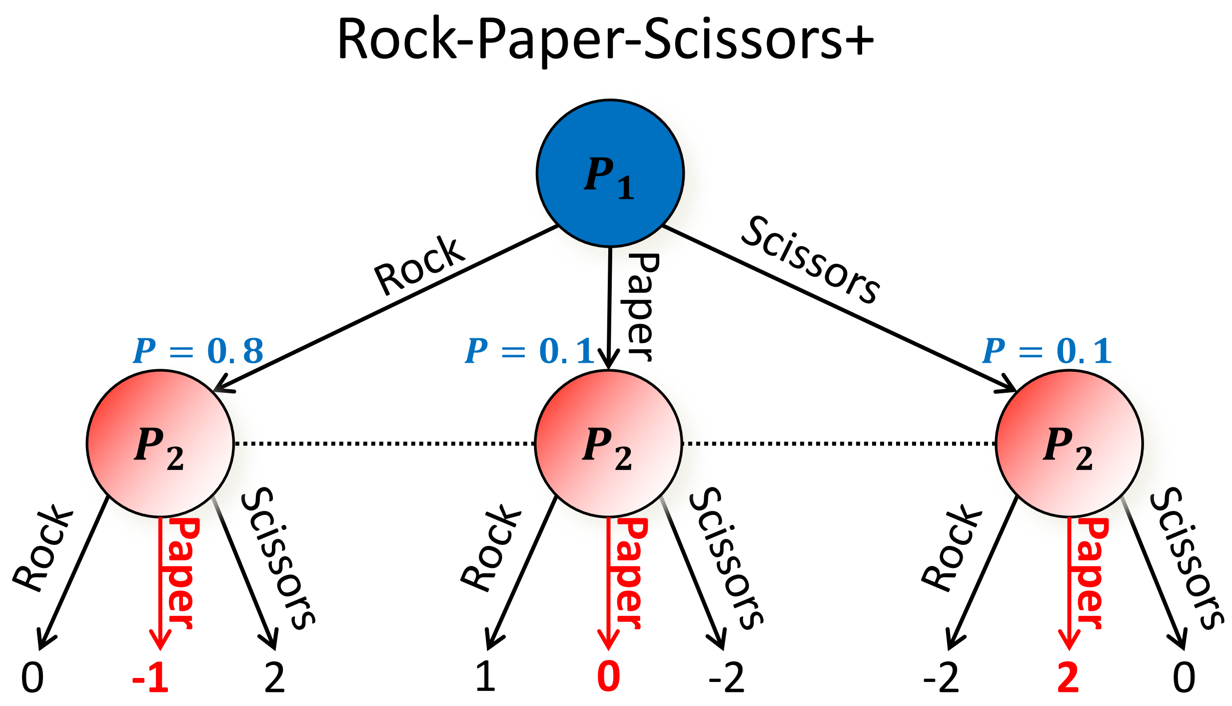
Rock-Paper-Scissors+

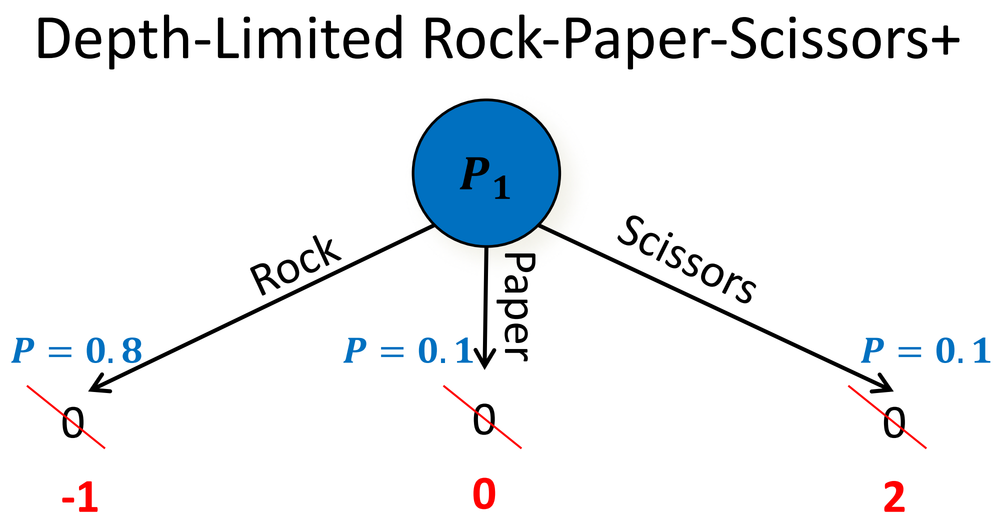


Depth-Limited Rock-Paper-Scissors+



Search in Imperfect-Information Games

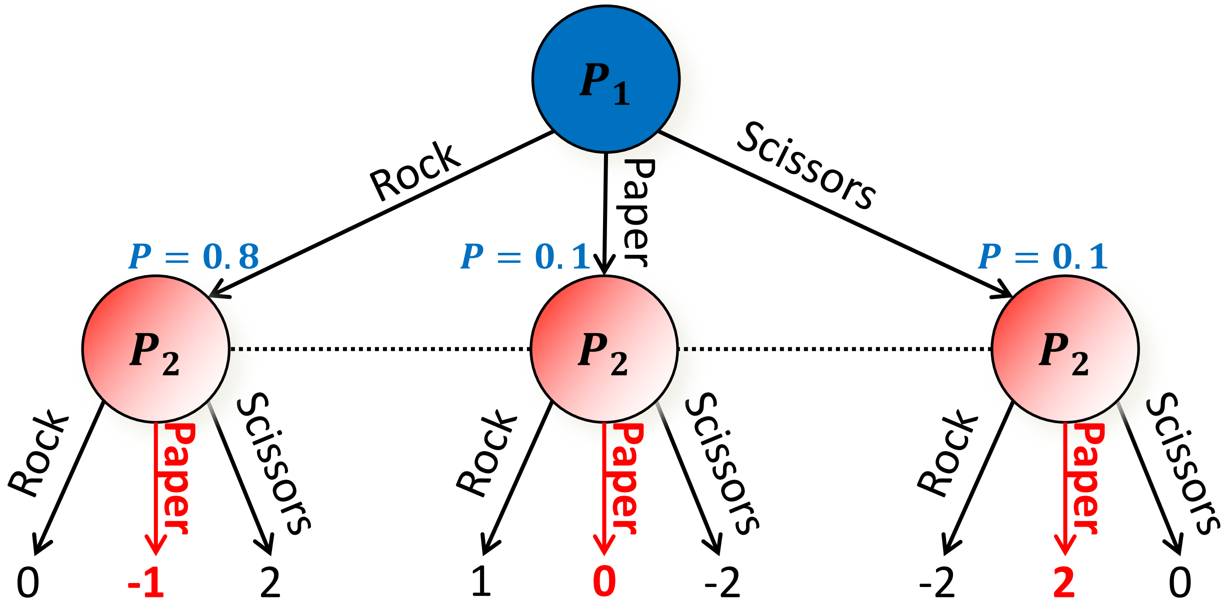




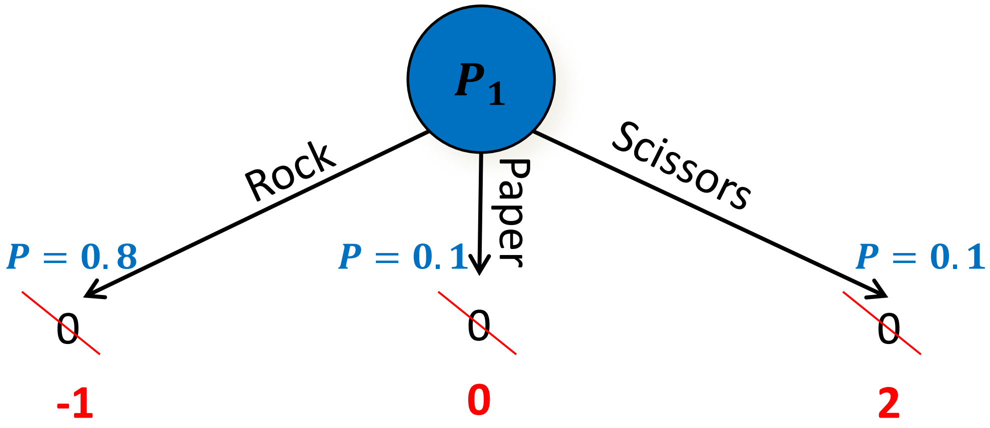
Critical assumption: Our entire policy is **common** knowledge, but the outcomes of random processes are **not** common knowledge

Search in Imperfect-Information Games

Rock-Paper-Scissors+

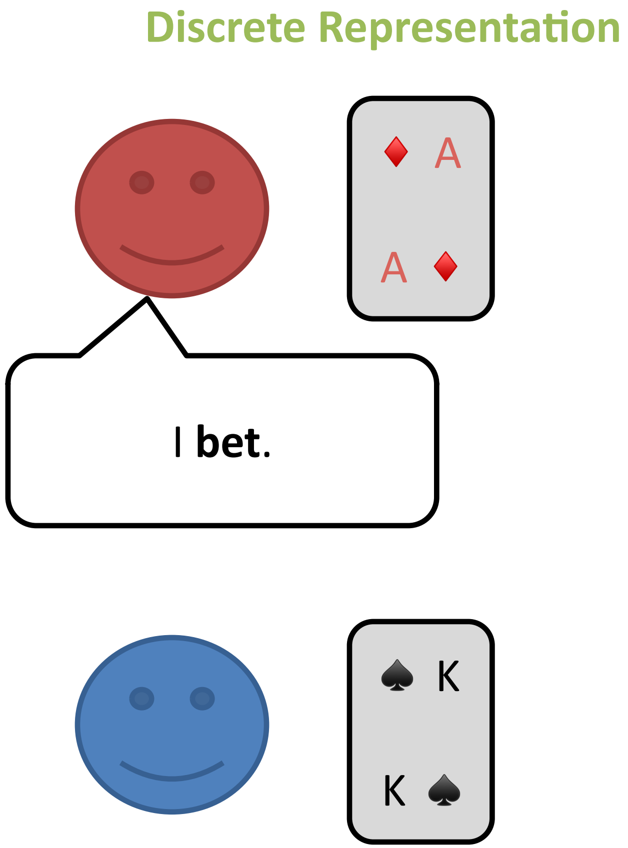


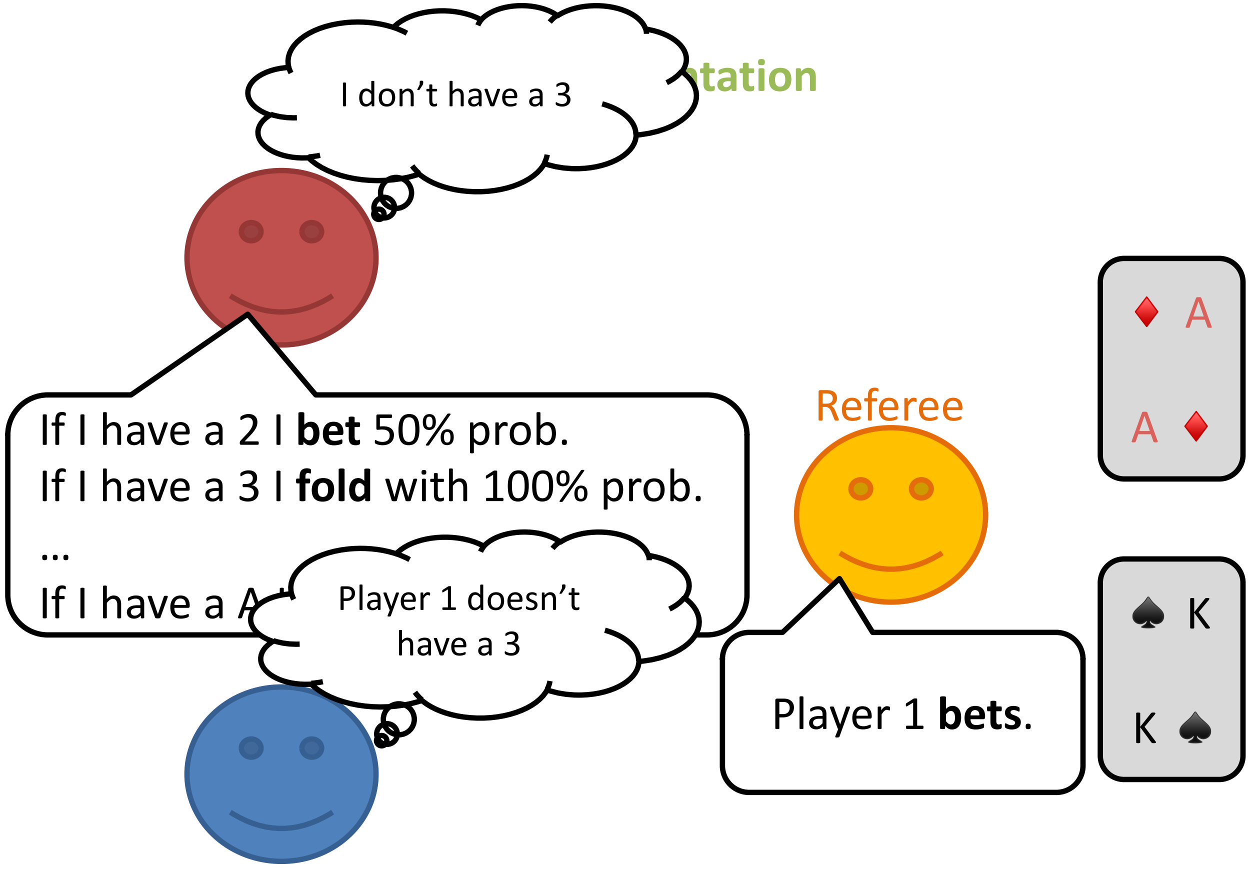
Depth-Limited Rock-Paper-Scissors+



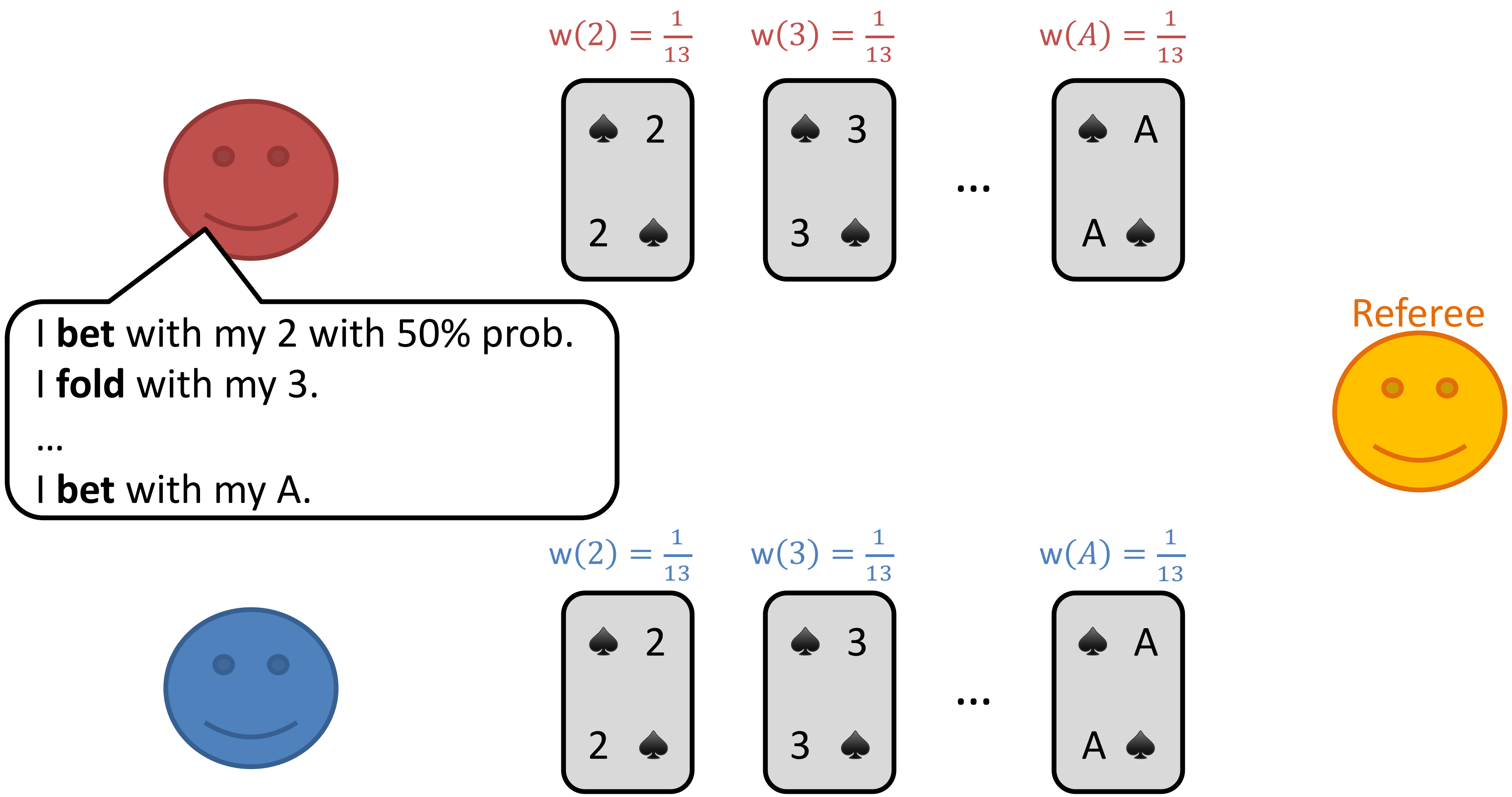
- One solution: define an imperfect-information game “state” as a **probability distribution over infosets** [Nayyar et al. IEEE-13]
  - $v(Rock)$ is not well-defined
  - $v([0.8\ Rock, 0.1\ Paper, 0.1\ Scissors]) = -0.6$
  - In more complex games, need to include probability distribution for **both** players

Converting imperfect-info games to continuous-state perfect-info games





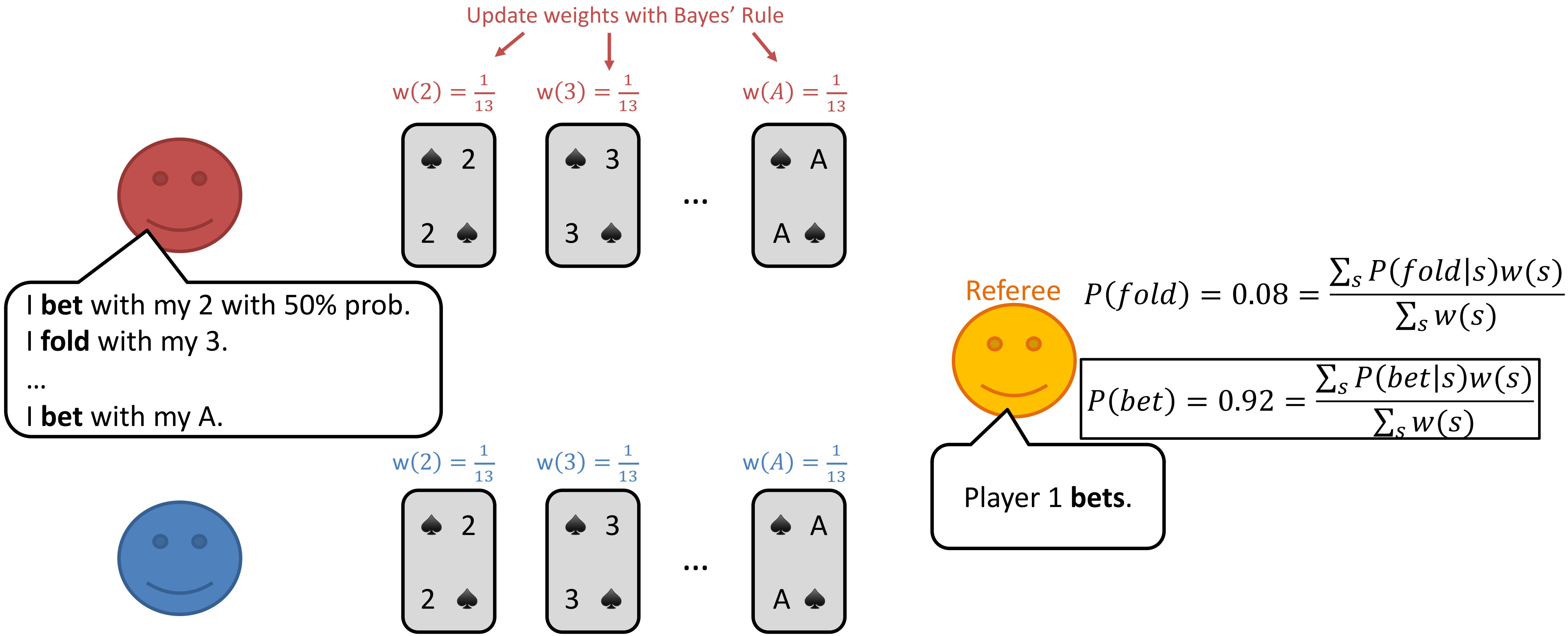
Converting imperfect-info games to continuous-state perfect-info games



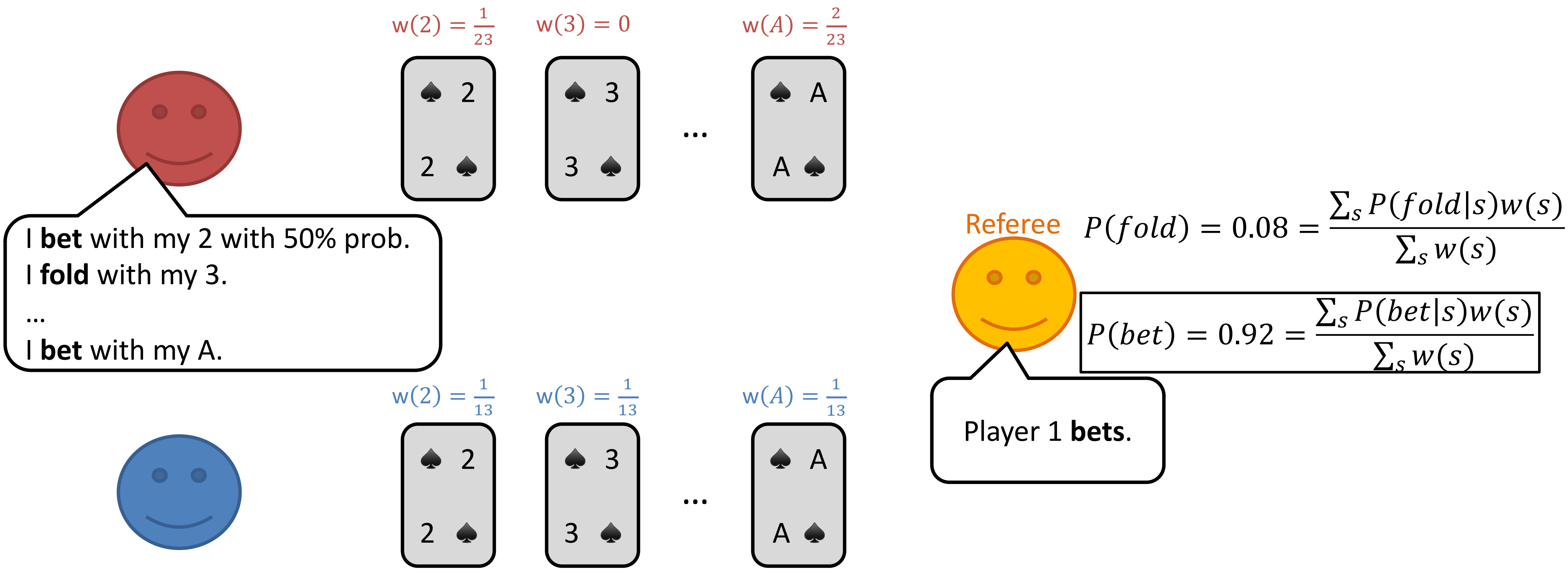
$$P(fold) = 0.08 = \frac{\sum_s P(fold|s)w(s)}{\sum_s w(s)}$$

$$P(bet) = 0.92 = \frac{\sum_s P(bet|s)w(s)}{\sum_s w(s)}$$

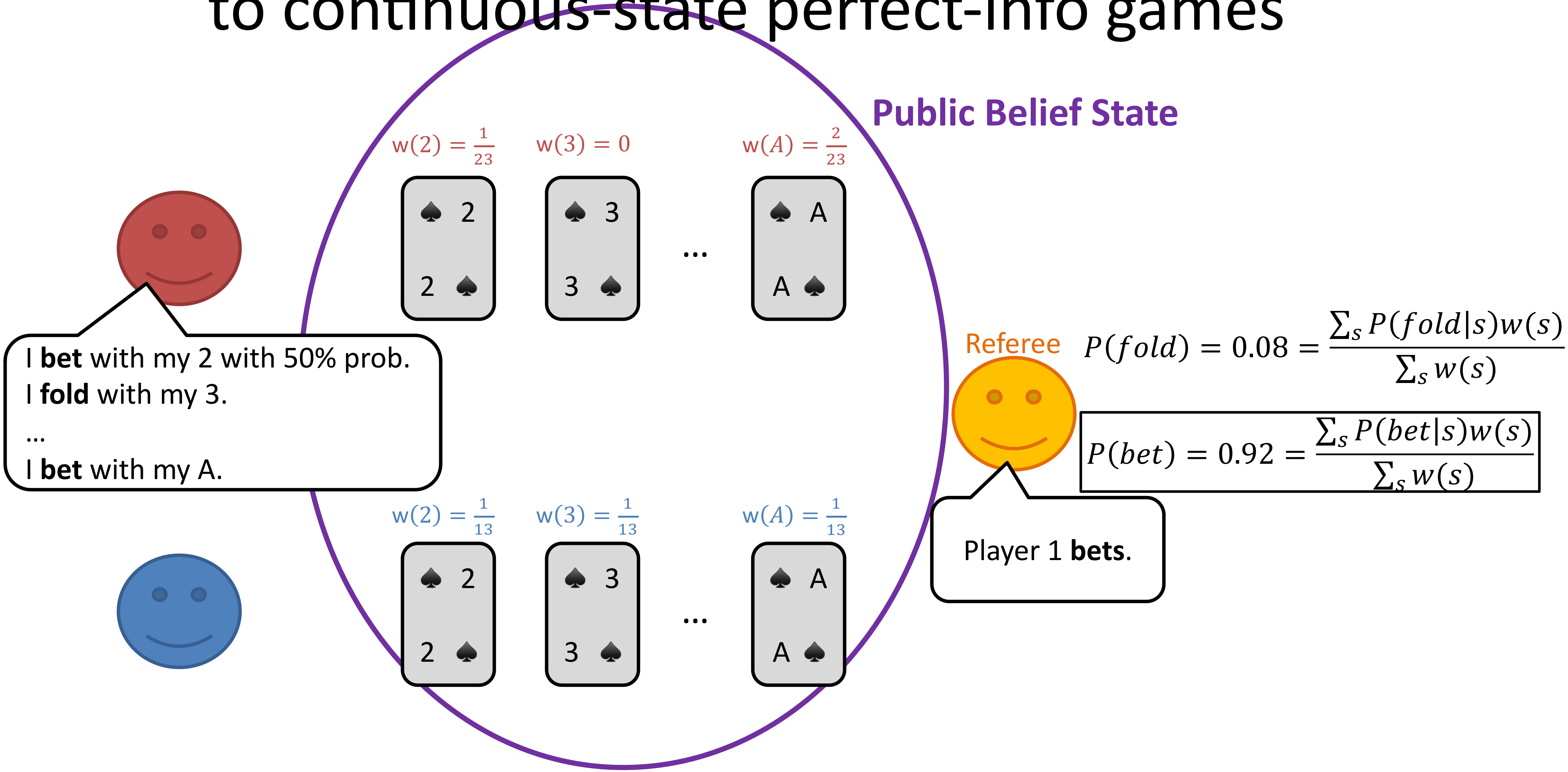
Converting imperfect-info games to continuous-state perfect-info games



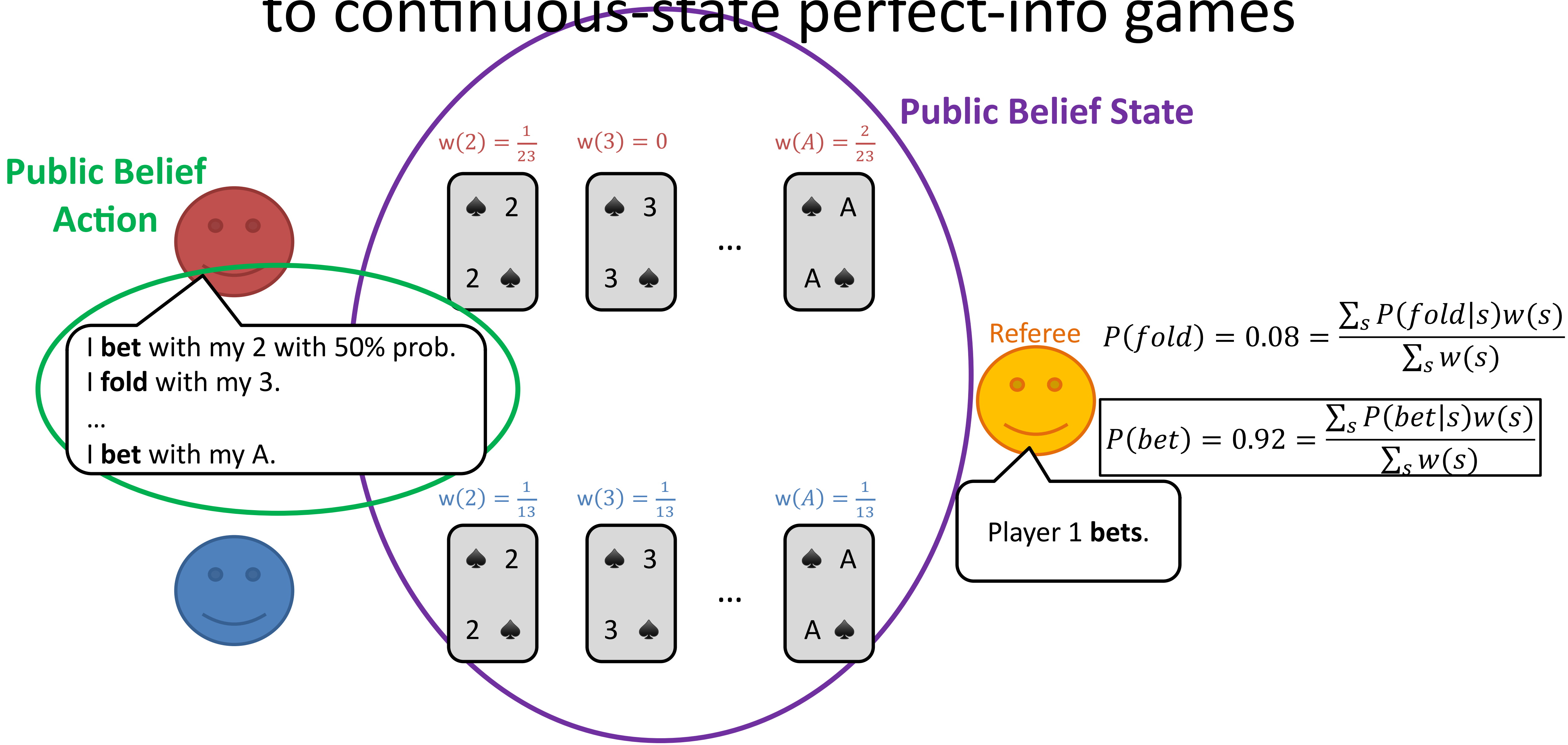
Converting imperfect-info games to continuous-state perfect-info games



Converting imperfect-info games to continuous-state perfect-info games



Converting imperfect-info games to continuous-state perfect-info games



Search in ReBeL

- We've shown all imperfect-information games can be converted into perfect-information games! Can we now run AlphaZero?

- In practice, no.
  - Action space is continuous with potentially *thousands* of dimensions
  - AlphaZero's Monte Carlo tree search would be completely intractable

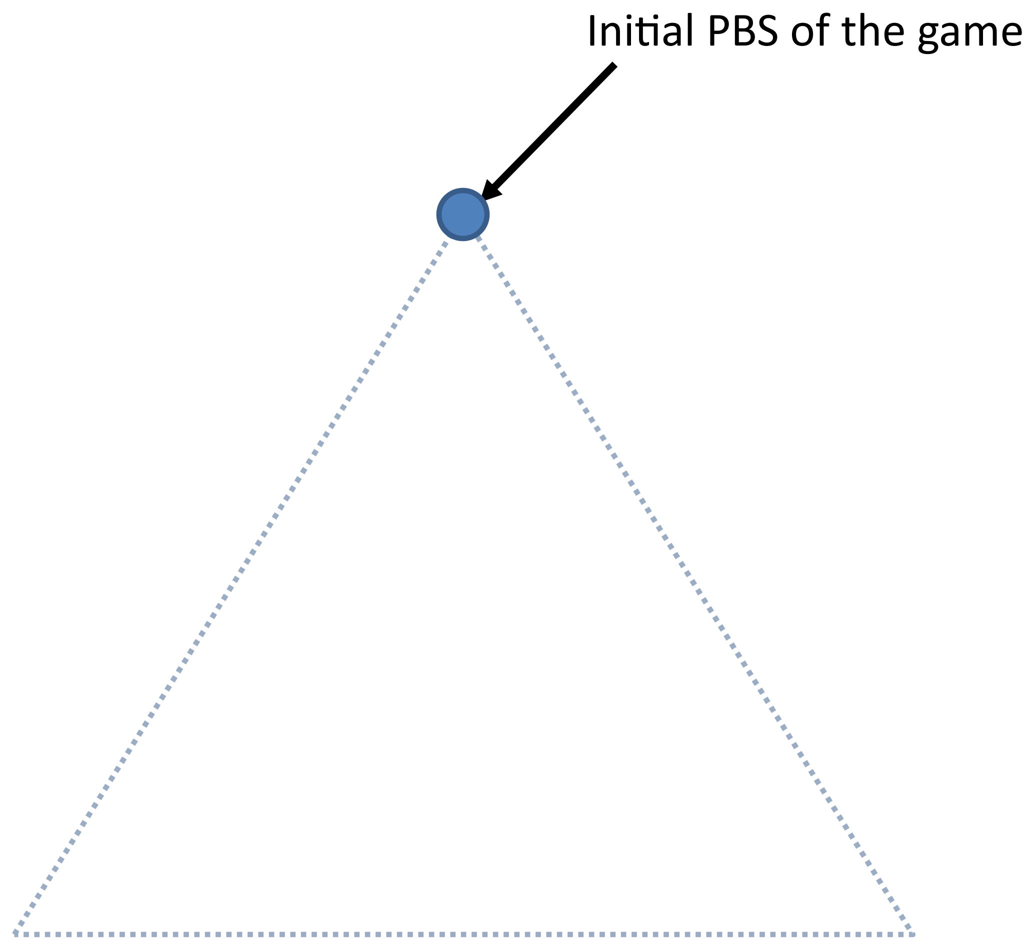
Search in ReBeL

- But! The continuous action space has special structure
  - Basically, it's convex
  - Technically a "bilinear saddle point problem"

- We can efficiently solve the imperfect-information subgames using **CFR**
  - Other equilibrium-finding algorithms also work

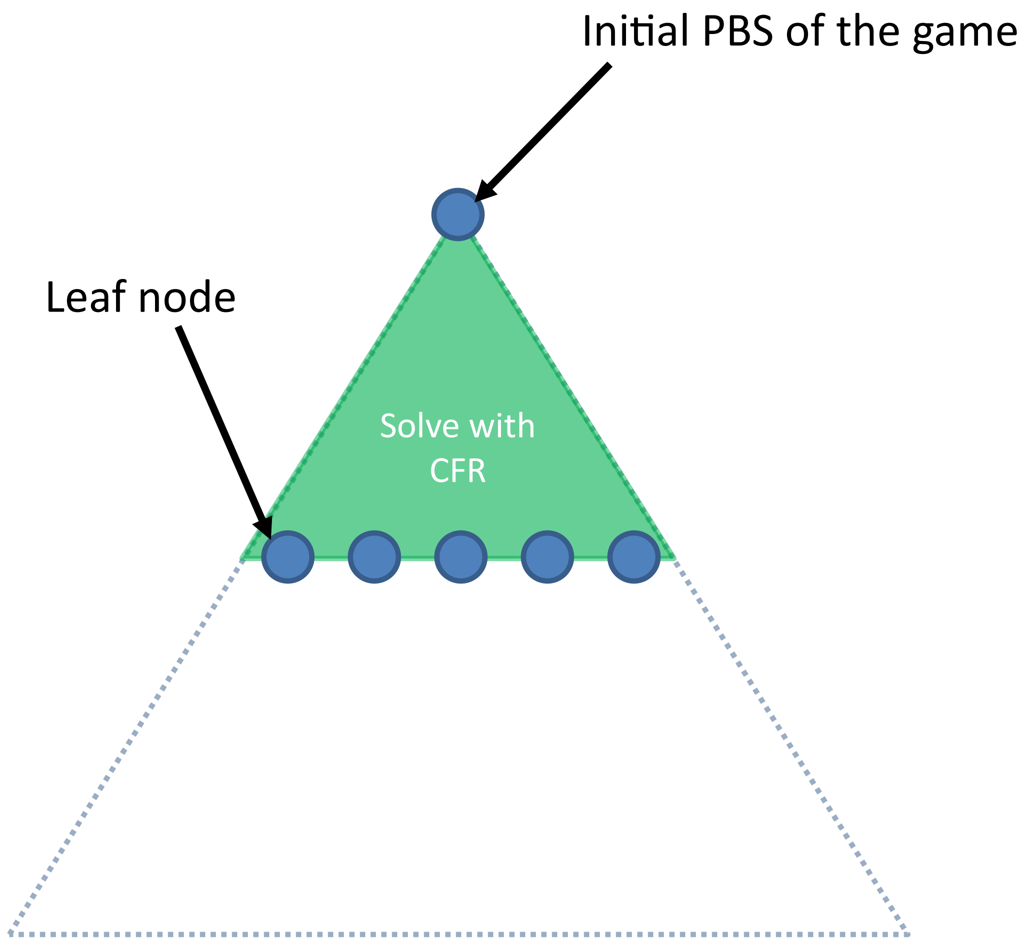
ReBeL

- Whenever an agent acts, generate a subgame and solve it



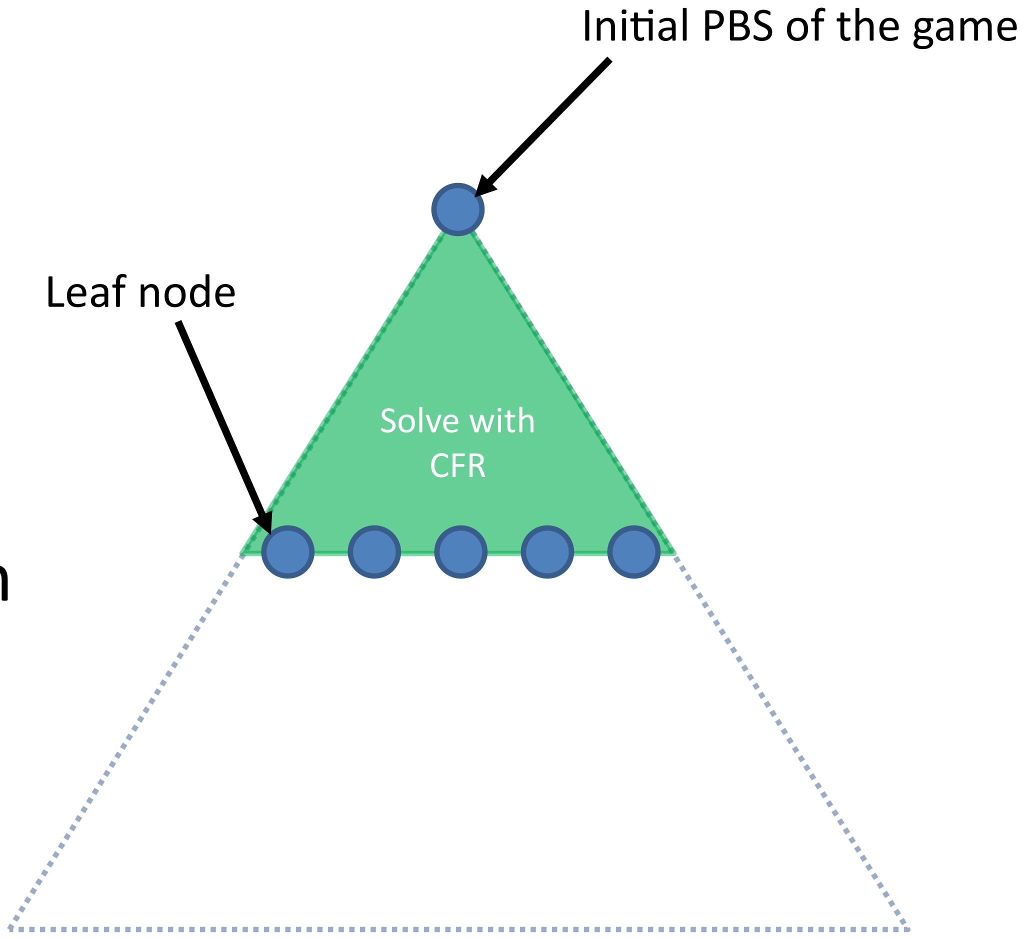
ReBeL

- Whenever an agent acts, generate a subgame and solve it
  - Solve using CFR



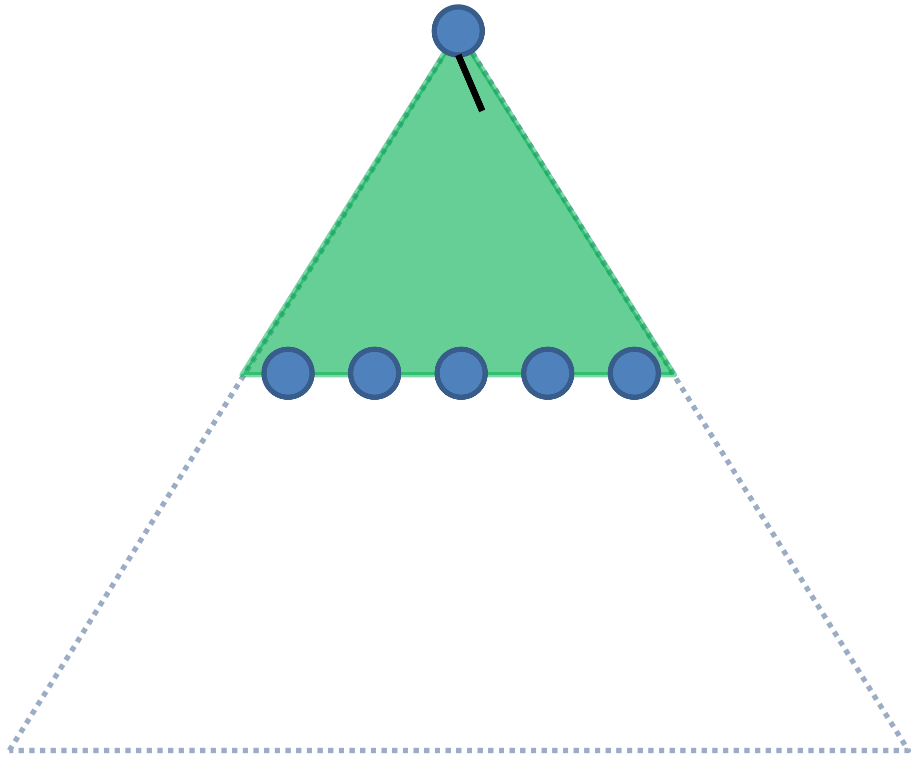
One more modification…

- Whenever an agent acts, generate a subgame and solve it
  - Solve using CFR
- CFR is an **iterative** algorithm, so value net must be accurate on **every** iteration
- To ensure proper exploration, we stop CFR on a **random** iteration



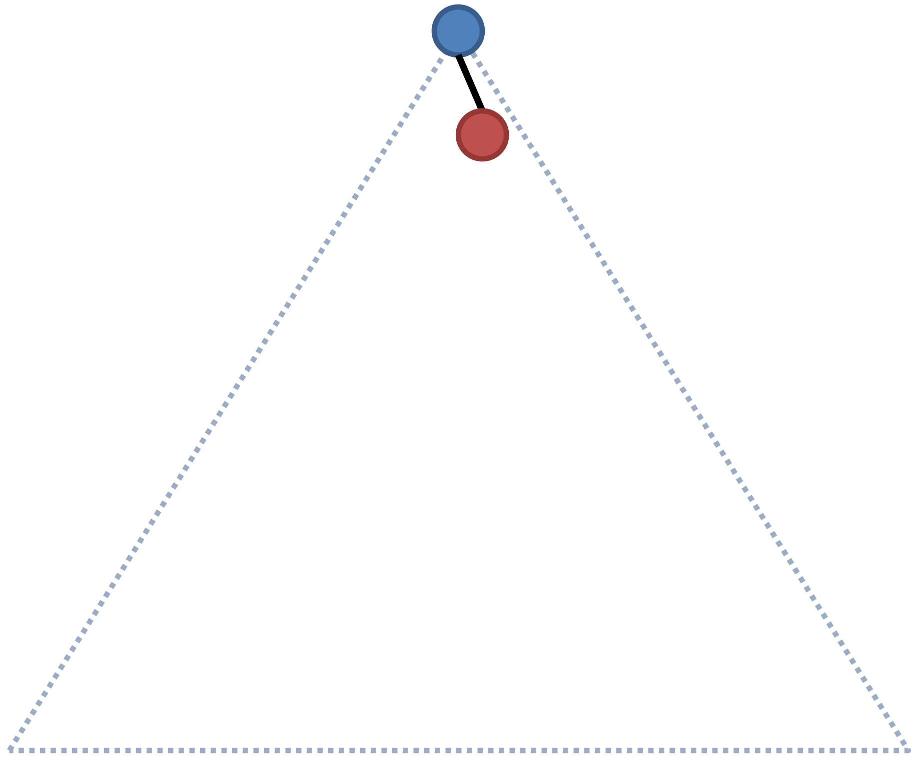
ReBeL

- Whenever an agent acts, generate a subgame and solve it
  - Solve using CFR
  - Stop on a random iteration
  - Take next action



ReBeL

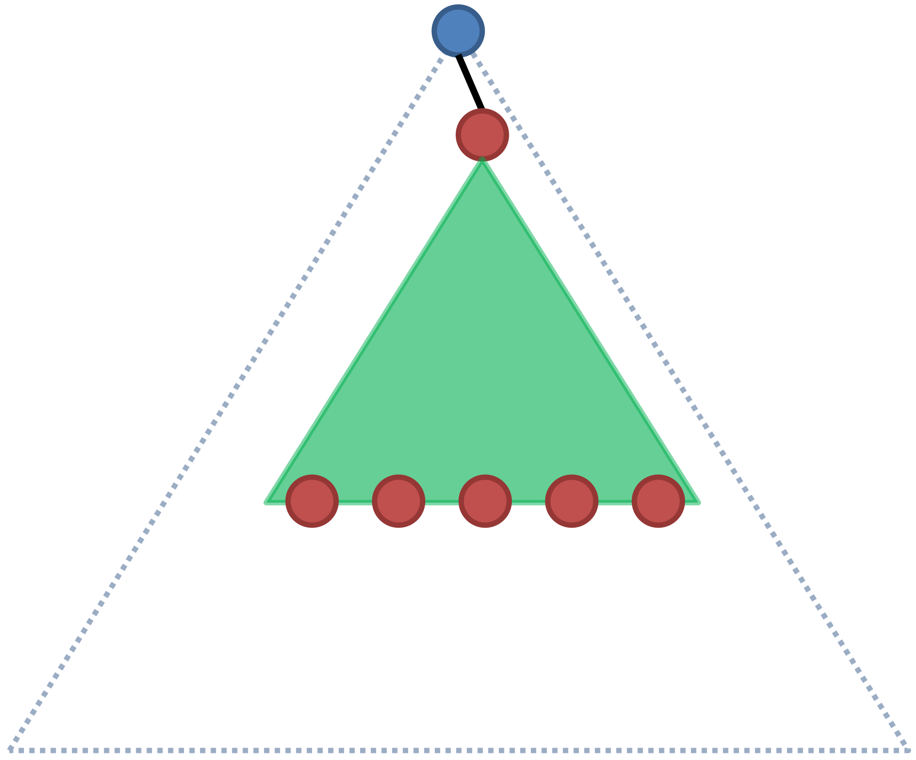
- Whenever an agent acts, generate a subgame and solve it
  - Solve using CFR
  - Stop on a random iteration
  - Take next action



ReBeL

- Whenever an agent acts, generate a subgame and solve it
  - Solve using CFR
  - Stop on a random iteration
  - Take next action

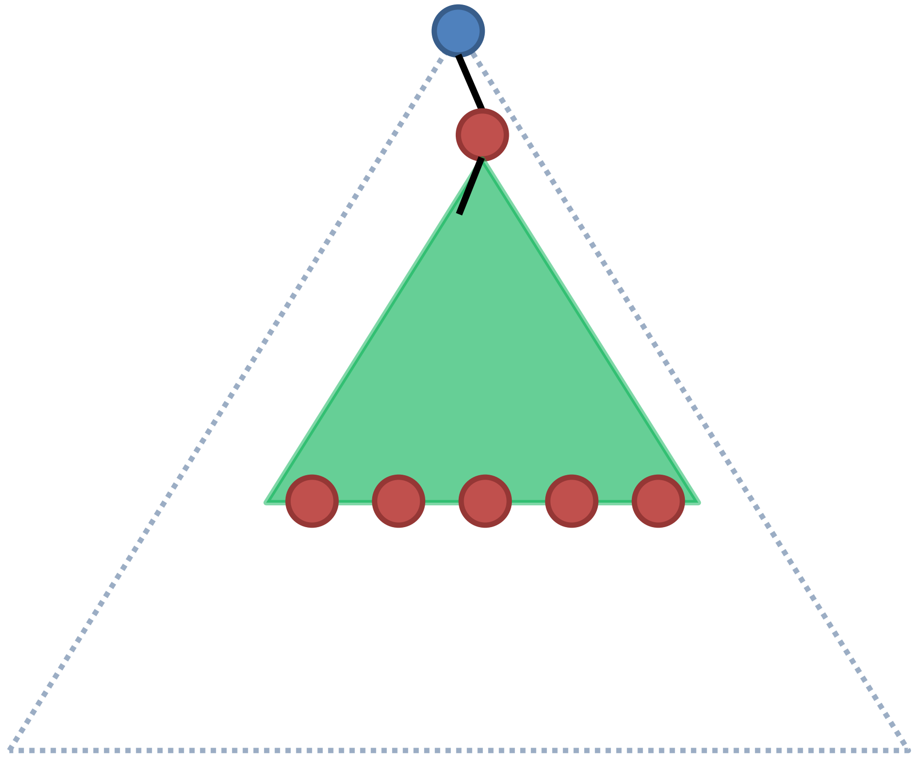
- Repeat until end of game



ReBeL

- Whenever an agent acts, generate a subgame and solve it
  - Solve using CFR
  - Stop on a random iteration
  - Take next action

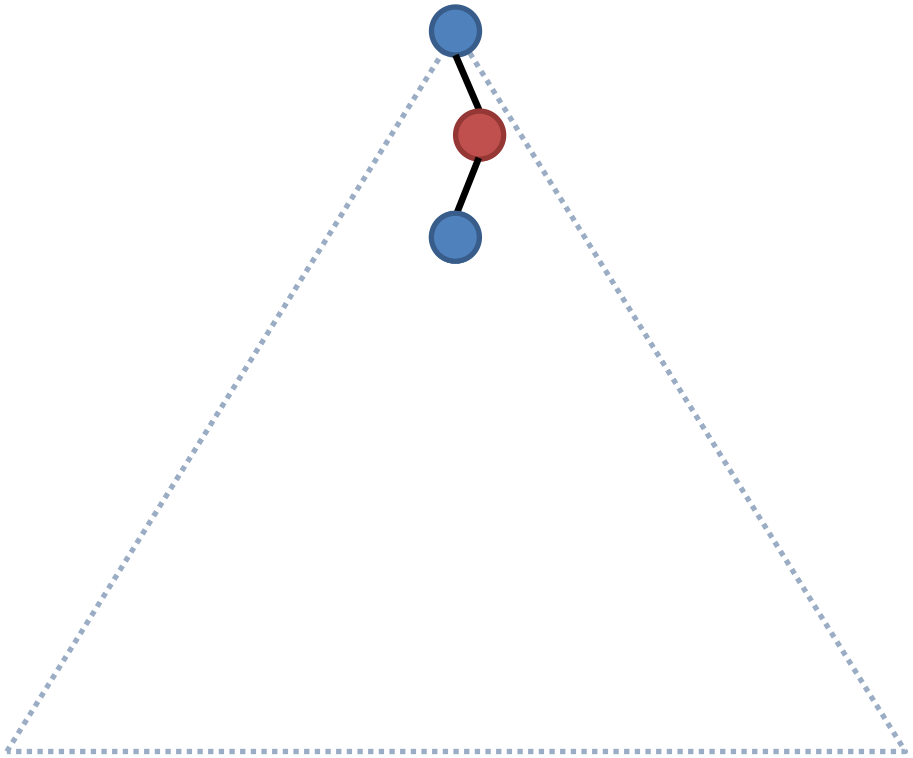
- Repeat until end of game



ReBeL

- Whenever an agent acts, generate a subgame and solve it
  - Solve using CFR
  - Stop on a random iteration
  - Take next action

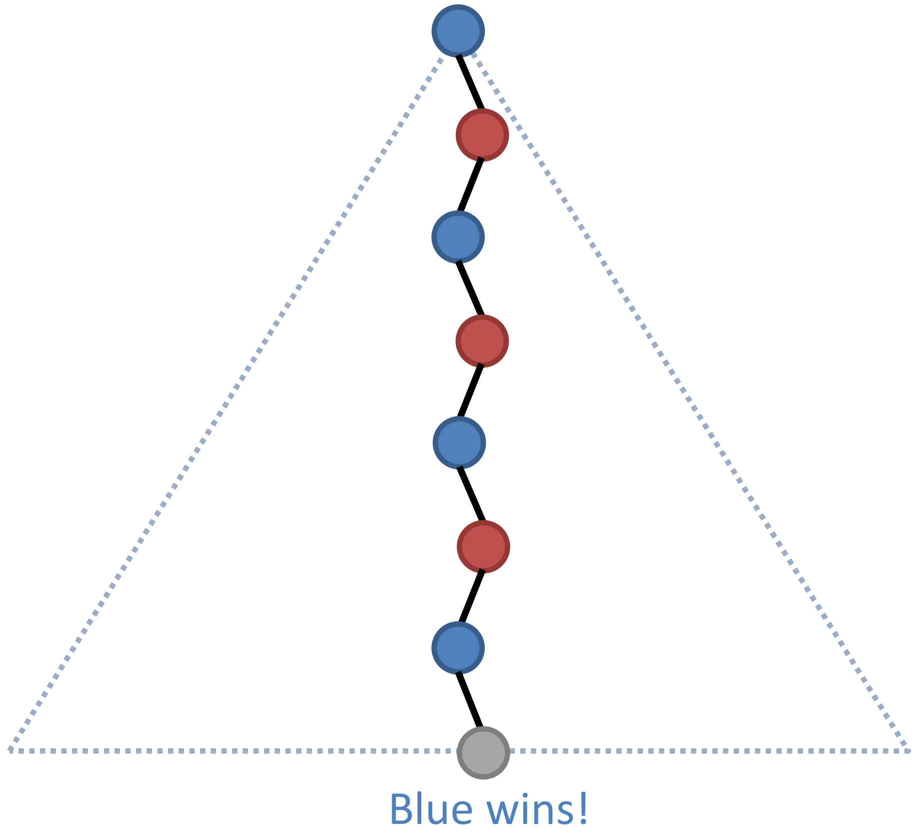
- Repeat until end of game



ReBeL

- Whenever an agent acts, generate a subgame and solve it
  - Solve using CFR
  - Stop on a random iteration
  - Take next action

- Repeat until end of game

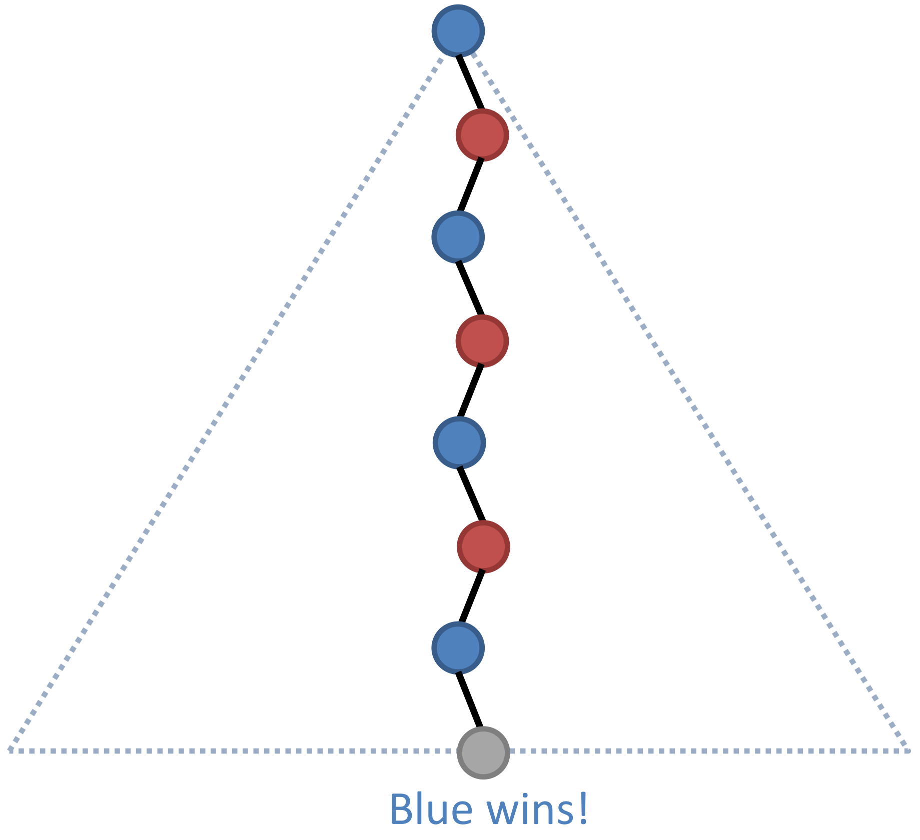


ReBeL

- Whenever an agent acts, generate a subgame and solve it
  - Solve using CFR
  - Stop on a random iteration
  - Take next action

- Repeat until end of game

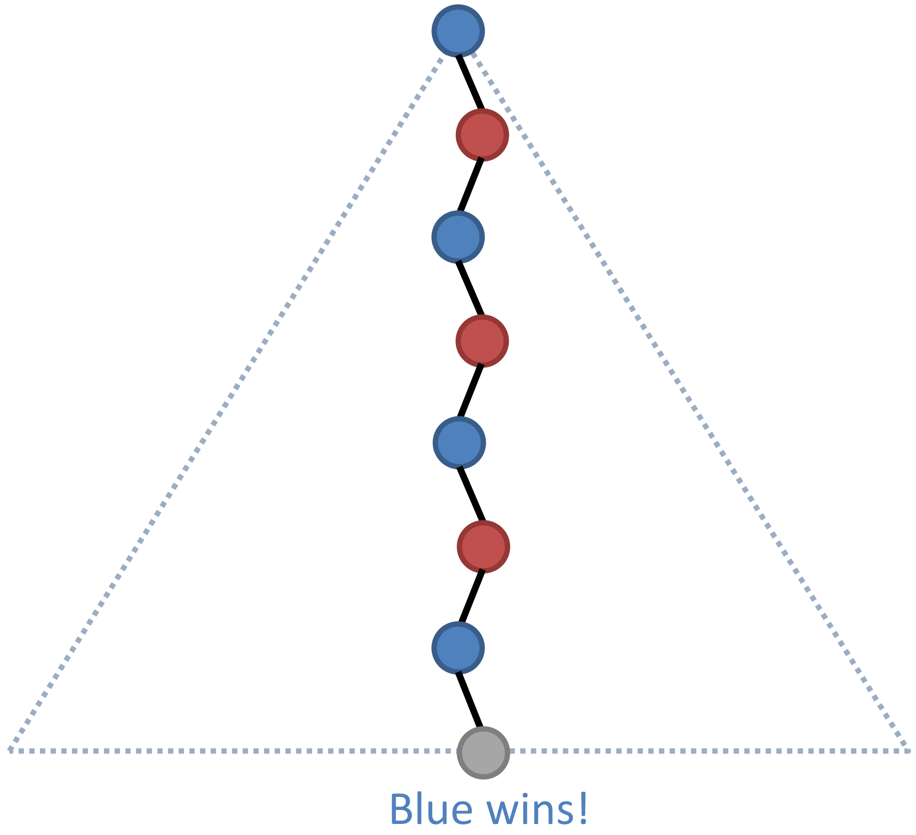
- Final value is used as a training example for all encountered PBSs



ReBeL

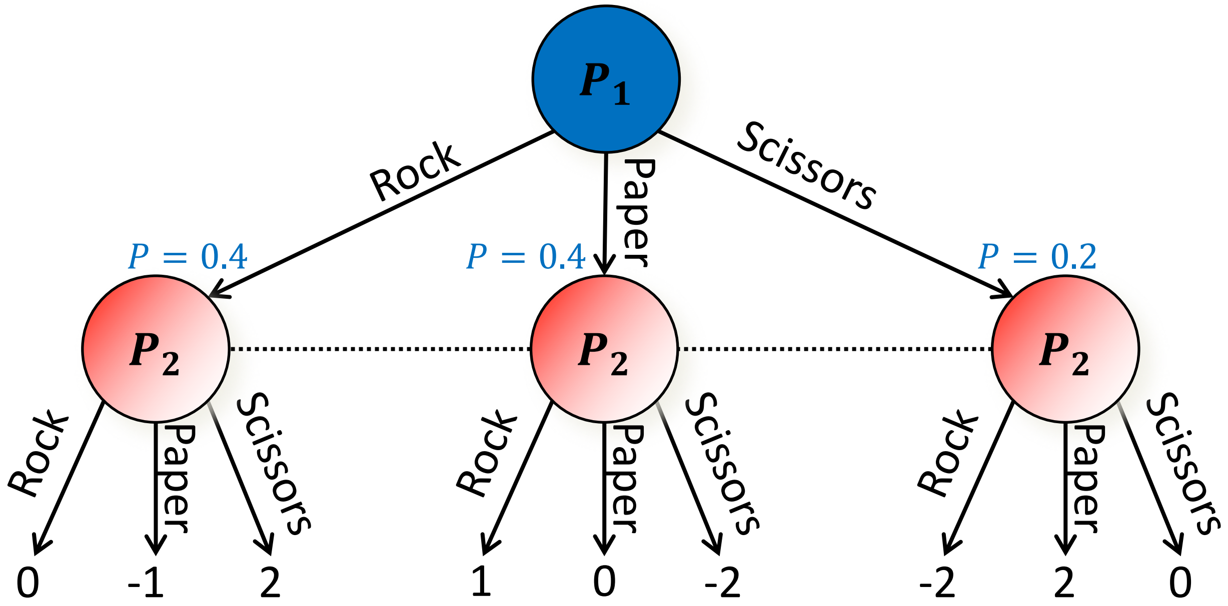
As with AlphaZero, ReBeL chooses a random action with ϵ probability to ensure proper exploration

Theorem: With tabular tracking of PBS values, ReBeL will converge to a $\frac{1}{\sqrt{T}}$-Nash equilibrium in finite time, where T is the number of CFR iterations



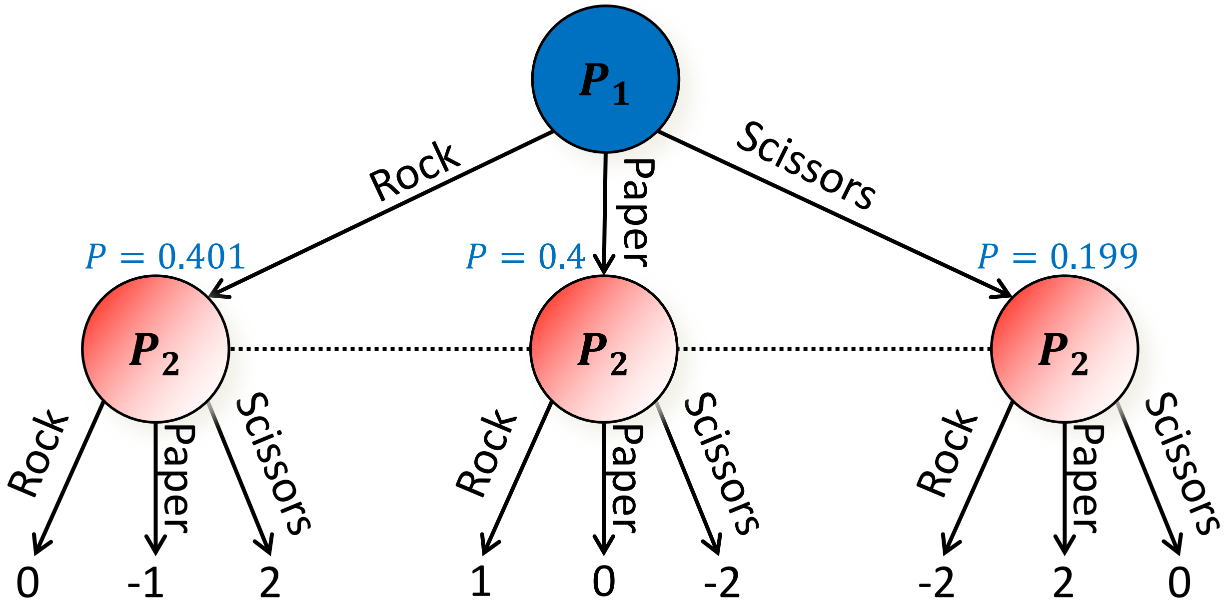
Playing Nash at Test Time

Rock-Paper-Scissors+

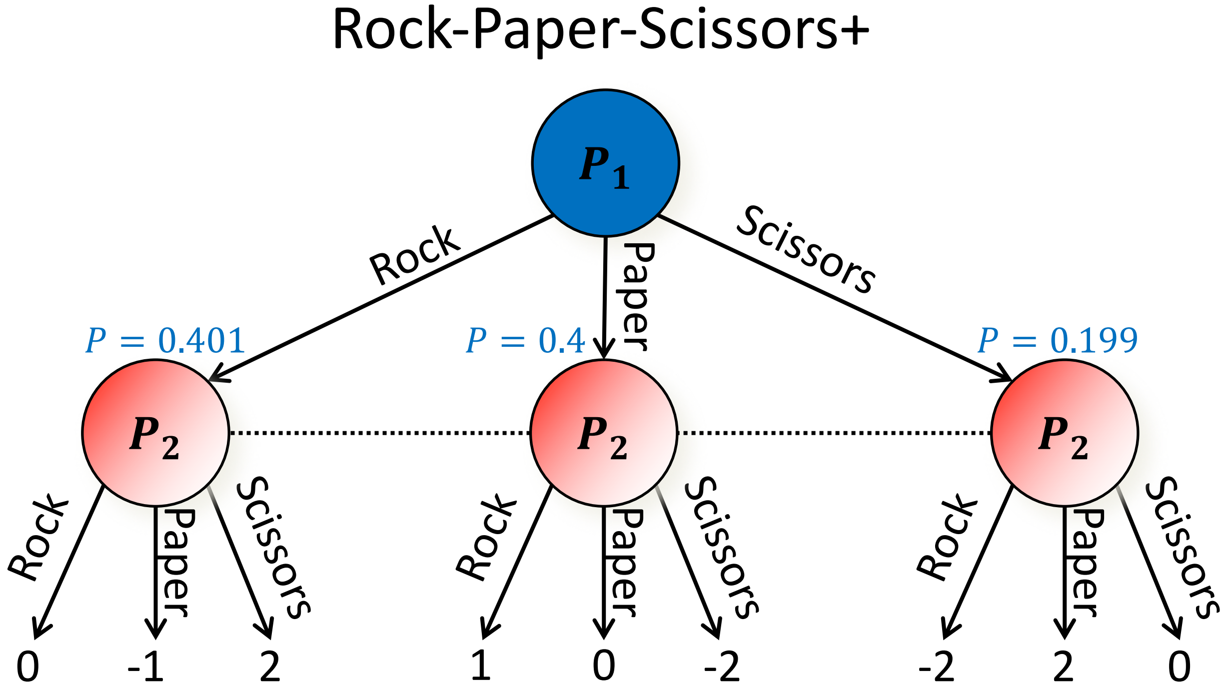


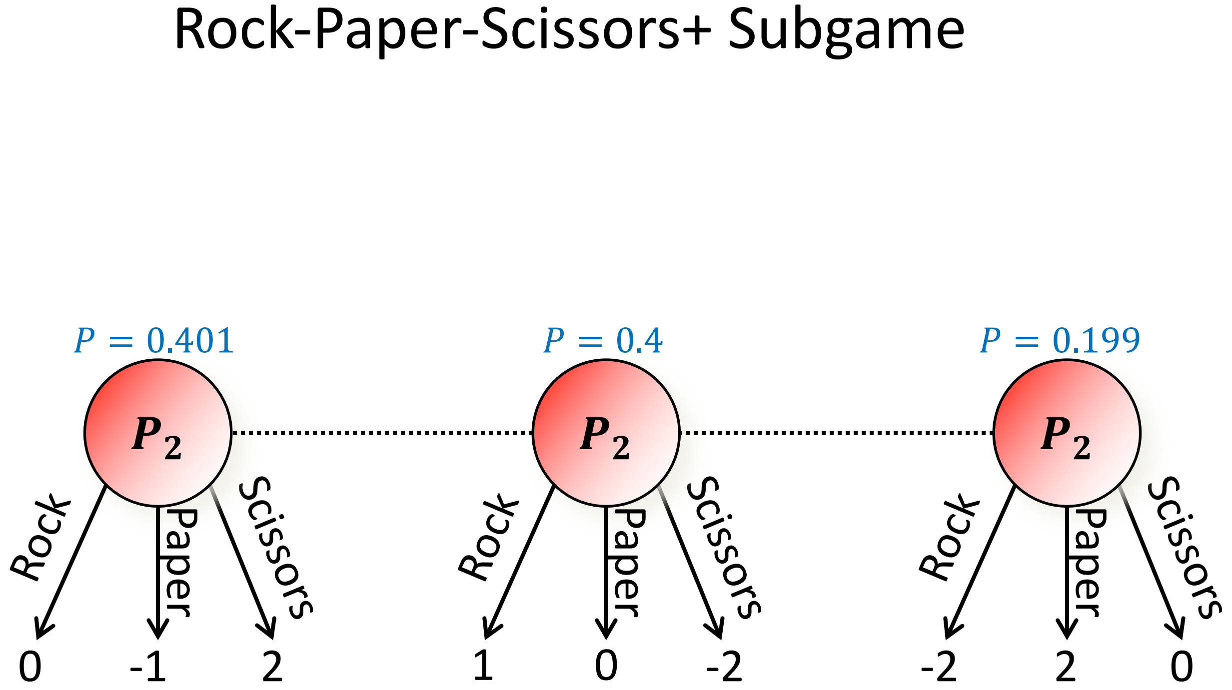
Playing Nash at Test Time

Rock-Paper-Scissors+



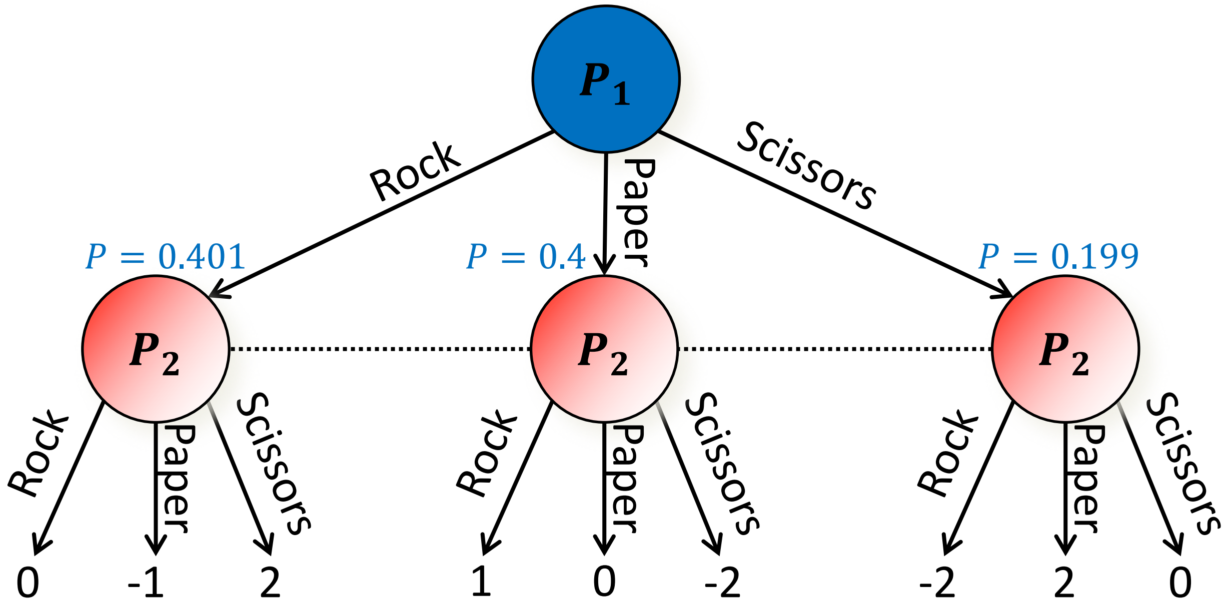
Playing Nash at Test Time



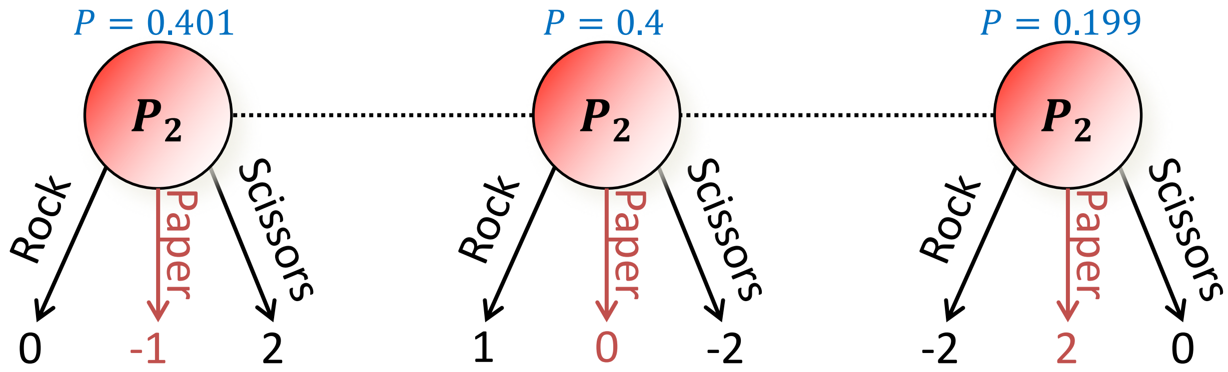


Playing Nash at Test Time

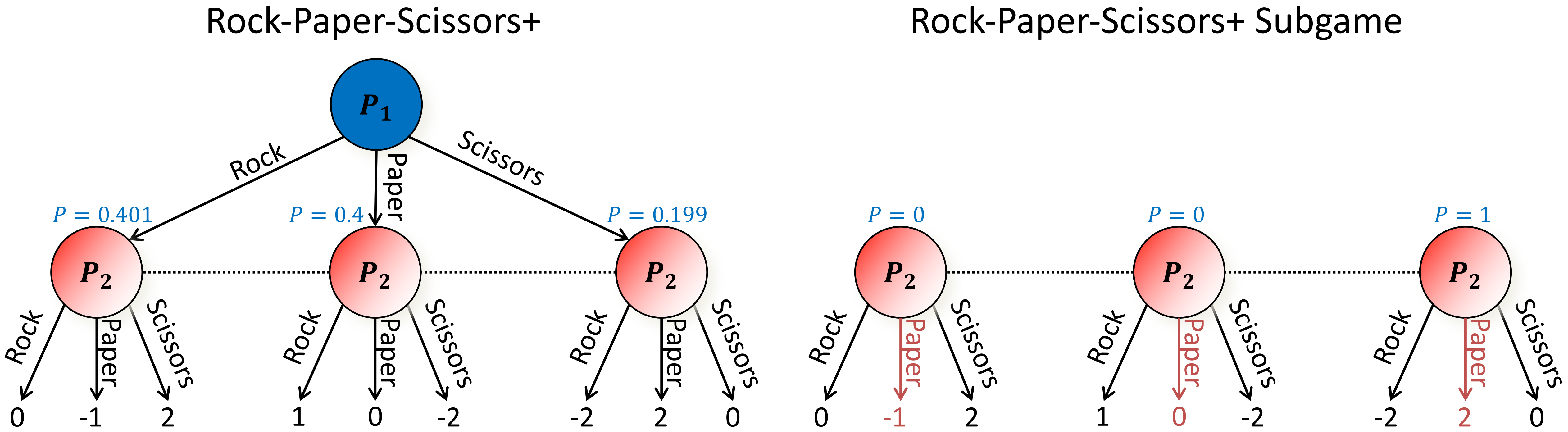
Rock-Paper-Scissors+



Rock-Paper-Scissors+ Subgame



Playing Nash at Test Time



- Our solution: Stop CFR on a **random** iteration and assume beliefs from that iteration
  - Opponent will not know our beliefs, so cannot predict in what way our policy will be pure
  - The **algorithm** will be a Nash equilibrium **in expectation**
  - This is the **exact same algorithm that is used during training**

Results in Two-Player No-Limit Texas Hold'em

	Slumbot	Baby Tartanian8	Local Best Response	Top Humans
DeepStack			383 ± 112	
Libratus		63 ± 14		147 ± 39
Modicum	11 ± 5	6 ± 3		
ReBeL	$\mathbf{45 \pm 5}$	$\mathbf{9 \pm 4}$	$\mathbf{881 \pm 94}$	$\mathbf{165 \pm 69}$

Results in Two-Player Liar's Dice

	1 die, 4 faces	1 die, 5 faces	1 die, 6 faces	2 dice, 3 faces
Tabular Full-Game FP	0.012	0.024	0.039	0.057
Tabular Full-Game CFR	0.001	0.001	0.002	0.002
ReBeL with FP	0.041	0.020	0.040	0.020
ReBeL with CFR	**0.017**	**0.015**	**0.024**	**0.017**

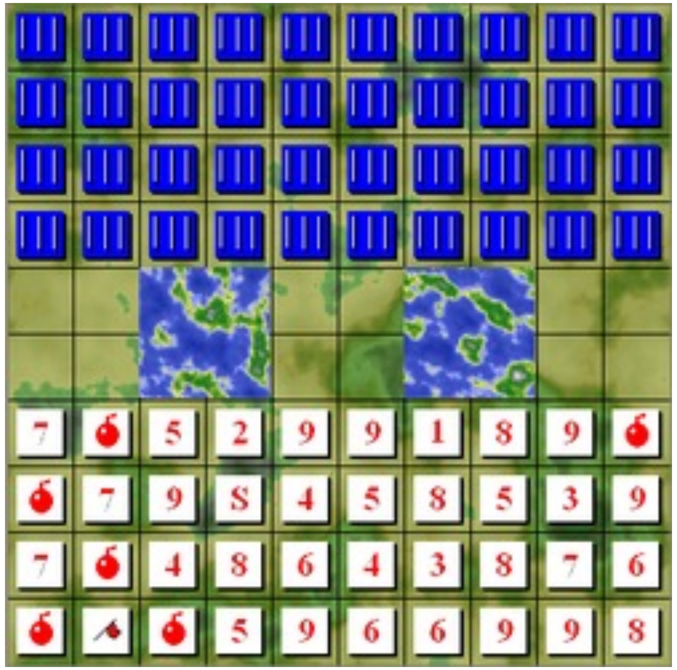
Source code available at github.com/facebookresearch/rebel

Key takeaways

- ReBeL **provably converges to a Nash equilibrium** in two-player zero-sum games (both perfect-info and imperfect-info)
- ReBeL achieves **superhuman performance in poker** while using far **less domain knowledge** than any prior poker bot
- ReBeL reduces to an algorithm similar to **AlphaZero in perfect-information games**

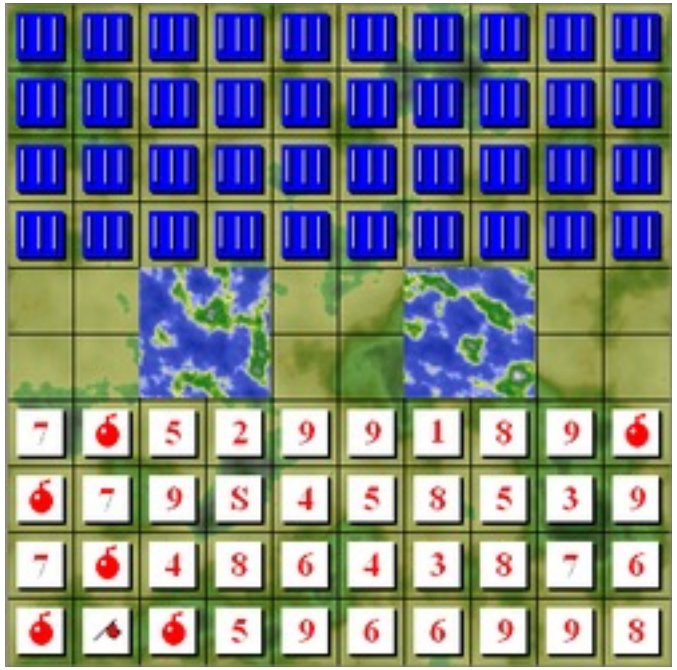
Remaining Challenges: More Hidden Information

- The input to ReBeL's state value function is all the possible action-observation histories
- In Texas hold'em poker there are 1,326 possible hands, so the input is 2,652 probabilities
- What if there is far more hidden information?



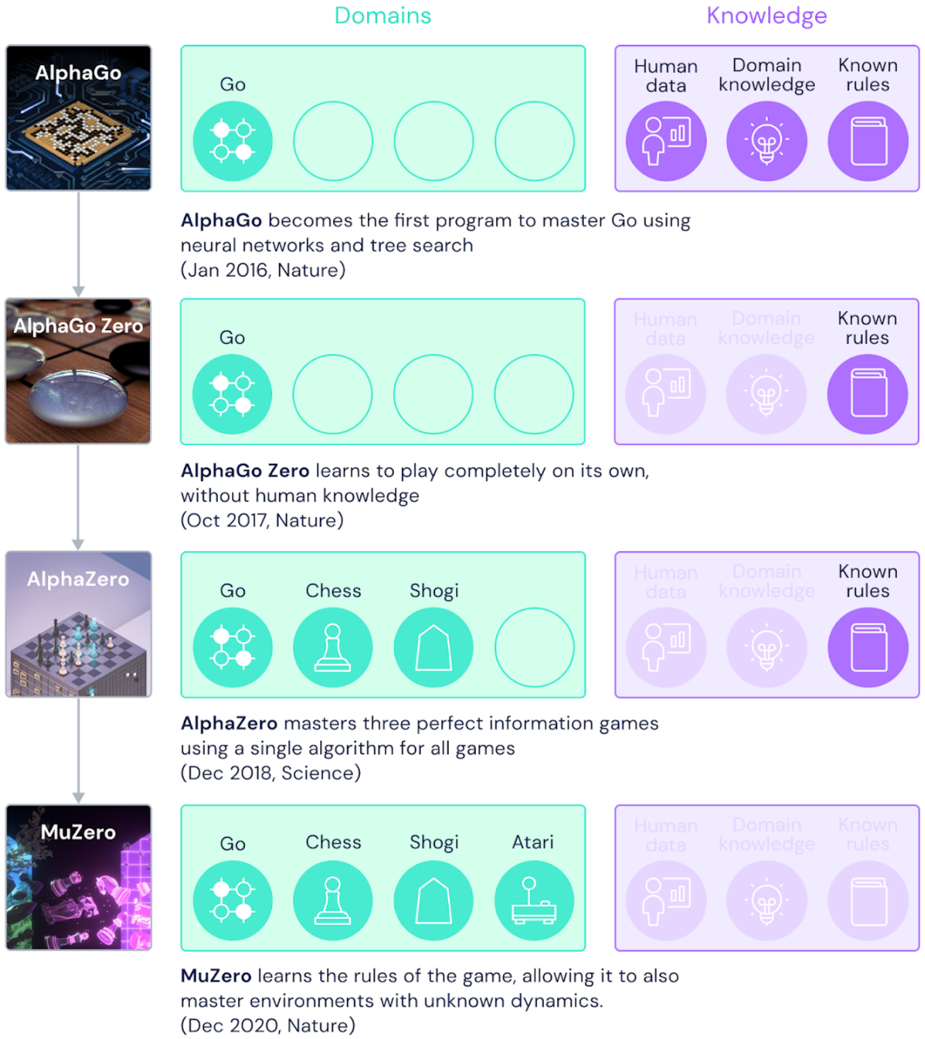
Remaining Challenges: More Hidden Information

- Two recent papers addressing this:
  - “**Scalable Online Planning via Reinforcement Learning Fine-Tuning**” Fickinger, Hu, Amos, Russell, Brown NeurIPS-21
  - “**A Fine-Tuning Approach to Belief State Modeling**” Sokota, Hu, Wu, Kolter, Foerster, Brown (Under Review)



Remaining Challenges: Learning Without a Simulator

- MuZero extends AlphaZero to work without a known simulator
- Can we extend ReBeL to work without a simulator as well?
- Can we make a single algorithm that can play all two-player zero-sum games without a simulator?



Remaining Challenges: Going Beyond Two-Player Zero-Sum

- **Life isn't zero sum:** AIs are still bad at cooperation, negotiation, and coalition formation

- Self play isn't enough!
  - Given infinite time and resources, a self-play chess bot will learn the Sicilian Defense
  - Given infinite time and resources, a self-play negotiation bot will **not** learn the English language

Thank You!

Noam Brown

www.noambrown.com