Lecture: ReBeL

Recursive Belief-Based Learning for Imperfect-Information Games
CSCE 631 — Intelligent Agents: Computational Game Solving

Alan Kuhnle

Learning Objectives

By the end of this lecture, you will understand:

1. Why AlphaZero-style methods fail in imperfect-information games
2. What a **public belief state (PBS)** is and why it enables search
3. How ReBeL combines RL and search for imperfect-information games
4. The mathematical foundation: PBS values and supergradients
5. The inner loop (subgame solving) and outer loop (blueprint updates)
6. Why beliefs depending on policies isn't circular
7. How ReBeL achieved superhuman poker performance

Big picture: ReBeL = AlphaZero for imperfect-information games [3]

Roadmap

Part 1: The Problem

- Why we can't just use AlphaZero for poker
- The value-depends-on-probability issue

Part 2: The Core Insight

- From hidden states to public belief states
- How beliefs make it "perfect information"

Part 3: Mathematical Foundation

- PBS values and Nash equilibrium
- Infostate values as supergradients
- Convergence theorems

Part 4: The Algorithm & Results

- Inner/outer loops, safe search
- Head-to-head results

The Dream: AlphaZero for Poker

AlphaZero's success in Go, Chess, Shogi:

- Learn value function $V(s)$ through self-play
- Use MCTS with $V(s)$ to guide search
- Train on MCTS-improved policies
- Repeat until superhuman

Question: Can we do the same for poker?

The Dream: AlphaZero for Poker

AlphaZero's success in Go, Chess, Shogi:

- Learn value function $V(s)$ through self-play
- Use MCTS with $V(s)$ to guide search
- Train on MCTS-improved policies
- Repeat until superhuman

Question: Can we do the same for poker?

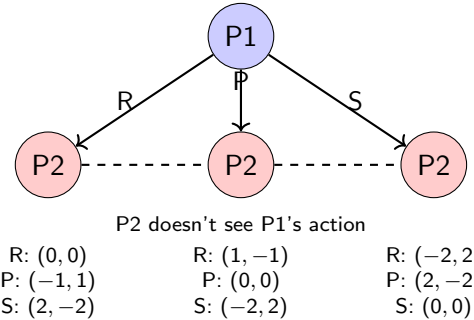
Problem: In poker, states have **hidden information**

- Your cards: private
- Opponent's cards: unknown
- State representation must handle uncertainty

Naive approach: Use information sets (what you observe)
But: This breaks AlphaZero's assumptions [3]

Why Naive Search Fails: Modified Rock-Paper-Scissors

Game variant [3]: Winner gets 2 points when either player chooses Scissors

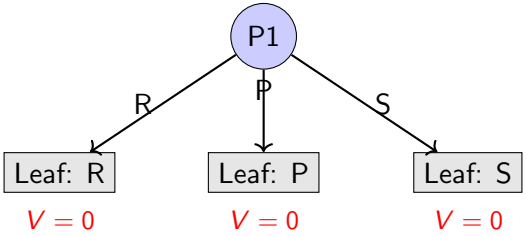


Nash equilibrium: Both play R=0.4, P=0.4, S=0.2

Why? When playing this mixed strategy, each action has EV = 0

The Problem: Values Depend on Probabilities

Suppose we try one-ply search for Player 1:



Problem: All three actions look equally good ($V = 0$)!

- Search gives no signal to prefer R/P over S
- Cannot recover the 40-40-20 distribution
- Leaf values alone are insufficient [3]

Root cause: The *probability* you play an action affects its value
This violates AlphaZero's assumption that states have unique values

Why This Breaks Standard RL+Search

In perfect-information games (Go, Chess):

- Each state s has a unique value $V(s)$
- Value depends only on optimal play from that state
- Value at one state is independent of your strategy at other states
- Search works by backing up these unique values

In imperfect-information games:

- Value of an information set depends on **how often you play** each action there
- Example: Scissors is worth playing 20% of the time, but not more
- So the notion "value of this infoset" is ill-posed without specifying the whole strategy

Implication: Need a different notion of "state" that:

1. Captures the strategic uncertainty (hidden cards, etc.)
2. Is fully observable to all players
3. Has well-defined values we can learn

Analogy: Belief States in Single-Agent Settings

Partially Observable MDPs (POMDPs):

- Agent doesn't see true state x
- Only gets observations o
- Maintains belief: $b(x) = \Pr(x \mid o_{1:t}, a_{1:t})$

Key trick: Treat the *belief* b as the state

- Belief-MDP is a fully observable MDP over beliefs
- Hidden state becomes latent randomness in transitions
- Can apply standard MDP algorithms

ReBeL's insight: Do the same thing for *multi-agent games* [3]

- State = **public belief** about all private information
- Converts imperfect-information game $\rightarrow$ perfect-information game over beliefs

Public vs. Private Information

In poker (and many games), information splits into:

Public information (everyone sees):

- Public cards (flop, turn, river)
- Betting actions taken
- Pot size

Private information (hidden):

- Your hole cards
- Opponent's hole cards

Key observation: Everyone agrees on what's public
But each player has different private info

Public Belief States (PBS)

Fix:

- Prior distribution over private info (e.g., uniform card deal)
- A **blueprint strategy** σ_{blue} for all players

For each public history h, compute belief via Bayes' rule:

$$b_{\sigma_{\text{blue}}}(c_1, c_2 \mid h) \propto \Pr(c_1, c_2) \cdot \Pr(h \mid c_1, c_2, \sigma_{\text{blue}})$$

Public Belief State:

$$\beta = \left(s_{\text{pub}}, \Delta S_1(s_{\text{pub}}), \Delta S_2(s_{\text{pub}})\right)$$

where $\Delta S_i(s_{\text{pub}})$ is a probability distribution over player i's *private* infostates consistent with the public history s_{pub}.

Critical property: This β is **common knowledge**

- All players can compute the same β
- No hidden information at the belief level

How Does This Make It Perfect Information?

Original game:

- True state includes public history h and private cards (c_1, c_2)
- Players don't know which private cards they and opponent have jointly
- $\Rightarrow$ Imperfect information

PBS game:

- State is $\beta = (s_{\text{pub}}, \Delta S_1, \Delta S_2)$
- All players observe the same β
- No information sets at the belief level
- $\Rightarrow$ Perfect information (over beliefs)

Where did hidden cards go?

- Now just *latent randomness* in transitions and payoffs
- Exactly like belief-MDP for POMDPs

Implication: Can use perfect-information search techniques! [3]

Example: Kuhn Poker Setup

Game rules:

- Deck: $\{J, Q, K\}$; two players, each dealt one card
- Ante 1 chip each
- One betting round:
  - P1: check or bet (1 chip)
  - If P1 checks: P2 can check (showdown) or bet (P1 folds/calls)
  - If P1 bets: P2 folds or calls

Underlying states: (c_1, c_2, h)

- $c_1, c_2 \in \{J, Q, K\}$, $c_1 \neq c_2$
- h: action history (public)

Information structure:

- Public history h visible to both
- Private: c_1 known only to P1, c_2 only to P2

Example: Constructing the PBS

Fix simple blueprint for P1:

- With K: always bet $\rightarrow \sigma_{\text{blue}}(B \mid c_1 = K) = 1$
- With Q: bet 30% $\rightarrow \sigma_{\text{blue}}(B \mid c_1 = Q) = 0.3$
- With J: bet 10% $\rightarrow \sigma_{\text{blue}}(B \mid c_1 = J) = 0.1$

Prior: $\Pr(c_1, c_2) = 1/6$ for each distinct pair

After observing $h = B$ (P1 bet), update belief:

c_1	c_2	Prior	$\sigma_{\text{blue}}(B \mid c_1)$	Unnormalized
J	Q	1/6	0.1	0.1/6
J	K	1/6	0.1	0.1/6
Q	J	1/6	0.3	0.3/6
Q	K	1/6	0.3	0.3/6
K	J	1/6	1.0	1.0/6
K	Q	1/6	1.0	1.0/6

Normalize $\rightarrow b_{\sigma_{\text{blue}}}(\cdot \mid h = B)$

PBS: $\beta = (h = B, b)$ — single shared state in PBS game

Wait—Beliefs Depend on Policies. Isn't This Circular?

Concern: $b_\sigma(\cdot \mid h)$ depends on strategy σ
Doesn't this create a circular dependency?

Wait—Beliefs Depend on Policies. Isn't This Circular?

Concern: $b_\sigma(\cdot \mid h)$ depends on strategy σ
Doesn't this create a circular dependency?

ReBeL's solution: Two-loop structure

1. **Inner loop:** *Fix* blueprint σ_{blue}
   - This defines a fixed PBS game $G(\sigma_{\text{blue}})$
   - Learn values/policies in this fixed game
2. **Outer loop:** Update σ_{blue} based on learned policies
   - Use new σ_{blue} to define new PBS game
   - Repeat

Analogy to familiar algorithms:

- Policy iteration: evaluate under π_k, improve $\rightarrow \pi_{k+1}$
- CFR: use current strategies to compute regrets, then update

No contradiction: Beliefs fixed during inner loop; updated between iterations [3]

PBS Values in Two-Player Zero-Sum Games

Key result: In 2p0s games, every PBS has a well-defined minimax value [3]

Value of PBS β when all players play policy profile σ:

$$V_i^{\sigma}(\beta) = \sum_{h \in \mathcal{H}(s_{\text{pub}}(\beta))} p(h \mid \beta)\, v_i^{\sigma}(h)$$

where:

- $\mathcal{H}(s_{\text{pub}})$: set of histories matching the public state $s_{\text{pub}}(\beta)$
- $p(h \mid \beta)$: probability of history h under belief distribution β
- $v_i^{\sigma}(h)$: expected value from history h under policy σ

Nash equilibrium value:

$$V_i(\beta) = V_i^{\sigma^*}(\beta)$$

where σ^* is any Nash equilibrium in the subgame rooted at β.

Critical property (2p0s): All Nash equilibria of that subgame give the same value, and

$$V_1(\beta) = -V_2(\beta).$$

Infostate Values: Definition

For player i in infostate s_i at PBS β, let $\mathcal{H}(s_i)$ be the set of histories consistent with s_i.

Best response value starting from s_i:

$$v_i^{\sigma^*}(s_i \mid \beta) = \max_{\sigma_i} \sum_{h \in \mathcal{H}(s_i)} p(h \mid s_i, \beta_{-i})\, v_i^{\langle \sigma_i, \sigma^*_{-i} \rangle}(h)$$

where:

- $p(h \mid s_i, \beta_{-i})$: probability of h given s_i and opponents' belief β_{-i}
- σ^*_{-i}: opponents' Nash equilibrium policy
- $\langle \sigma_i, \sigma^*_{-i} \rangle$: profile where player i plays σ_i, others play σ^*_{-i}

Interpretation: Maximum value player i can achieve at infostate s_i if opponents play Nash in the subgame rooted at β [3]

Theorem 1: Infostate Values as Supergradients

Theorem (Theorem 1 from [3])

, informal] Extend the PBS value function $V_1(\beta)$ to unnormalized belief distributions. For any PBS β and any Nash equilibrium σ^* in the subgame rooted at β, define a vector g with components

$$g_{s_1} := v_1^{\sigma^*}(s_1 \mid \beta) \quad \text{for each infostate } s_1.$$

Then g is a supergradient of this extended value function at β.

Interpretation:

- Infostate values encode how PBS value changes if we shift probability mass between infostates.
- Increasing probability on higher-value infostates increases $V_1(\beta)$ the fastest.
- "Supergradient" = generalized gradient for possibly non-smooth convex functions.

Why this matters: Lets us use gradient-based methods (like CFR) to optimize over beliefs. [3]

Why Use Infostate Values Instead of PBS Values?

Challenge: PBS β is a continuous probability distribution

- Cannot enumerate all possible PBSs
- MCTS-style search over continuous PBS space is intractable

In 2p0s games: PBS optimization is convex [3]

- Can use iterative gradient-based methods over beliefs
- CFR operates on "supergradients" of the value function
- Infostate values $v(s_i \mid \beta)$ provide exactly this information

ReBeL's value network:

$$\hat{v} : \beta \to \mathbb{R}^{|S_1|+|S_2|}$$

Returns a vector of infostate values, not a single scalar PBS value. [3]

This enables CFR-based search over belief space.

ReBeL: High-Level Structure

Two nested loops:

Inner Loop (for fixed blueprint σ_{blue})

Treat PBS game $G(\sigma_{\text{blue}})$ as environment:

1. Generate self-play data
2. Solve local subgames starting at sampled PBSs
3. Train value network $\hat{v}_\theta(\beta)$ and policy network $\hat{\sigma}_\theta(a \mid \beta)$

Outer Loop

Update blueprint:

- New blueprint = current networks: $\sigma_{\text{blue}}^{\text{new}} := \sigma_\theta$
- Recompute PBSs using new blueprint
- Repeat inner loop

Inner Loop: Subgame Solving

For a PBS β encountered during play:

1. **Construct depth-limited subgame** rooted at β
2. **Boundary conditions:**
   - Outside subgame: assume blueprint σ_{blue}
   - At leaf nodes (subgame frontier): use $\hat{v}_\theta$ for continuation values
3. **Run equilibrium-finding algorithm** (e.g., CFR) in subgame:
   - Computes approximate Nash policy σ^*_β
   - Produces infostate values $v^{\sigma^*}(s_i \mid \beta)$ for all s_i

Training targets:

- $\hat{v}_\theta(s_i \mid \beta) \approx v^{\sigma^*}(s_i \mid \beta)$ (value supervision)
- $\hat{\sigma}_\theta(a \mid \beta) \approx \sigma^*_\beta(a)$ (policy supervision)

Connection: Similar to DeepStack's continual resolving. [3]

CFR-Based Subgame Solving

On iteration t of CFR in subgame rooted at β_r:

1. CFR determines policy profile σ^t
2. For each leaf node z at the frontier, compute:

$$\hat{v}(s_i(z) \mid \beta_z^{\sigma^t})$$

   where $\beta_z^{\sigma^t}$ is the PBS at z when agents play according to σ^t
3. Back up to compute infostate/PBS values at root: $v^{\sigma^t}(\beta_r)$
4. Update CFR regrets and iterate

Key observation: Leaf values change every iteration (they depend on σ^t).

After T iterations:

$$\bar{v}(\beta_r) = \frac{1}{T} \sum_{t=1}^{T} v^{\sigma^t}(\beta_r)$$

This average is added to training data for $\hat{v}_\theta$. [3]

Inner Loop: Self-Play Data Generation

Procedure:

1. Start at initial PBS β_r
2. Solve subgame rooted at β_r using CFR + value network
3. Sample random iteration $t \sim \text{unif}\{1, \ldots, T\}$
4. Sample leaf PBS β' from the subgame according to policy σ^t
5. Add $(\beta_r, \bar{v}(\beta_r))$ and $(\beta_r, \bar{\sigma}(\beta_r))$ to training data
6. Repeat from β' until game ends

Why sample random iteration?

- CFR iterates through many policies $\sigma^1, \sigma^2, \ldots, \sigma^T$
- Want $\hat{v}_\theta$ accurate for PBSs arising under *any* iteration
- Ensures diverse PBS coverage during training

Exploration: One player samples random actions with probability $\varepsilon > 0$. [3]

Theorem 2: Convergence of Value Network

Theorem (Theorem 2 from [3])

, simplified] Consider an idealized value approximator that returns the most recent sample for sampled PBSs, and 0 otherwise. Running Algorithm 1 with T iterations of CFR in each subgame produces a value approximator with error at most

$$O\left(\frac{1}{\sqrt{T}}\right)$$

for any PBS that could be encountered during play, where the hidden constant depends on the game.

Interpretation:

- With perfect function approximation: convergence rate $O(1/\sqrt{T})$
- Same rate as tabular CFR
- More CFR iterations per subgame $\rightarrow$ better value estimates

Why This Converges (Intuition)

Idealized scenario:

- Rich function approximation (networks can represent any function)
- Accurate subgame solving (enough CFR iterations)
- Sufficient coverage (visit all relevant PBSs)

Then:

1. Value network learns accurate PBS values
2. Policy network learns good initial policies for search
3. Outer loop converges to a fixed point σ^*
4. σ^* is approximately Nash in original game

Why Nash in original game?

- PBS game is just a reparametrization of the original game
- Nash equilibrium in PBS game $\Leftrightarrow$ Nash in original [3]
- In 2p0s: guarantees $u_1(\sigma_1^*, \tau_2) \geq v^*$ for any τ_2

Test Time: The Safe Search Problem

Challenge at test time:

- During training: assumed opponent's policy is known (blueprint)
- At test time: don't know opponent's full policy
- Can't compute exact PBS based on *their* strategy

Modified RPS example:

- Run CFR as P1: get policy (R=0.4001, P=0.3999, S=0.2)
- **Unsafe search:** Pass down beliefs assuming you will follow this exact policy in the future
- Then optimize P2's response to those beliefs $\rightarrow$ might play (R=0, P=1, S=0)
- If opponent knew you'd do this conditioning, they'd exploit you

Intuition: conditioning your beliefs on your own future deviations can destroy the minimax guarantee.

Need: "Safe search" that plays Nash *in expectation*

- Not every realized policy must be Nash
- But the distribution over policies produced by search must be Nash

ReBeL's Safe Search Solution

Theorem (Theorem 3 from [3])

, simplified] If Algorithm 1 is run at test time with:

- *No off-policy exploration*
- *Value network with error at most δ (trained per Theorem 2)*
- *T iterations of CFR per subgame*

Then the algorithm plays an $O\left(\delta + \frac{1}{\sqrt{T}}\right)$-Nash equilibrium, with game-dependent constants.

How it works (sketch):

1. When doing search at test time, pick random CFR iteration t.
2. Assume all players follow policies from iteration t.
3. Compute PBS based on those (fixed) policies.
4. Solve subgame, play resulting policy for that episode.

In expectation: Distribution over played policies is Nash (up to approximation error). No additional safety constraints needed. [3]

Comparison: AlphaZero vs. ReBeL

AlphaZero	**ReBeL**
Domain: Perfect-info games	**Domain:** Imperfect-info games
State: World state w	**State:** Public belief state β
Search: MCTS over states	**Search:** CFR over PBS
Networks: $V(w)$, $\sigma(a\|w)$	**Networks:** $\hat{v}(\beta)$, $\hat{\sigma}(a\|\beta)$
Training: Self-play + MCTS	**Training:** Self-play + subgame CFR
Test time: MCTS	**Test time:** CFR with random iteration

Conceptual similarity:

- Both: RL + search at training and test time
- Both: Value network guides search; policy network warm-starts

Key difference: ReBeL operates on belief states to handle hidden information. [3]

Special Case: Perfect-Information Games

Question: If game has no hidden info, does ReBeL become AlphaZero?

Answer: Almost!

In perfect-information games:

- No private information $\rightarrow$ belief is trivial
- PBS β collapses to regular game state w
- ReBeL operates on states directly

Differences:

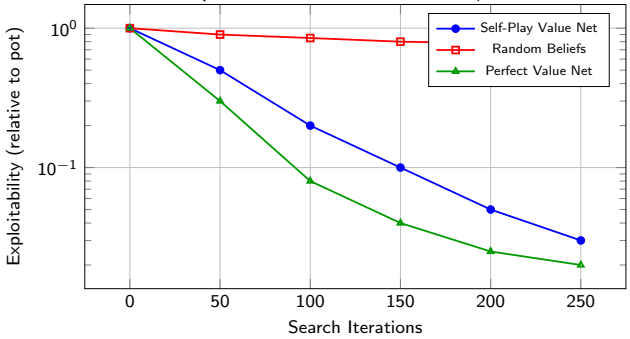
- ReBeL uses CFR for search (gradient-based)
- AlphaZero uses MCTS (tree search)
- Could swap in MCTS inside the PBS framework $\rightarrow$ essentially AlphaZero

Conclusion: ReBeL framework is *general*

- Handles both perfect and imperfect information
- Choice of search algorithm (CFR vs. MCTS) is modular [3]

Convergence: Turn Endgame Hold'em (TEH)

TEH: Simplified poker (first two rounds check/call automatically)



Key observation [3]:

- Learning beliefs from self-play (blue) almost matches having perfect values (green)
- Both dramatically beat a random-belief baseline (red)
- Exploitability is comparable to $\sim$125 iterations of full-game CFR

Context: Top poker AIs use 100–1000 CFR iterations.

Head-to-Head: ReBeL vs. Other Poker AIs

Heads-Up No-Limit Texas Hold'em:

Agent	vs. Slumbot	vs. BabyT8	vs. LBR	vs. Human
DeepStack	—	—	383±112	—
Libratus	—	63±14	—	147±39
Modicum	11±5	6±3	—	—
ReBeL	**45±5**	**9±4**	**881±94**	**165±69**

Units: milli-big-blinds per game (mbb/g); positive = ReBeL wins

Notes [3]:

- LBR: Local Best Response (constrained exploiter)
- Human: Dong Kim, top HUNL professional (7,500 hands)
- ReBeL: ¡ 2 sec/decision (max 5 sec)
- Libratus beat Dong Kim by only 29±78 mbb/g

Convergence: Liar's Dice

Liar's Dice variants: 1 die (4, 5, 6 faces) and 2 dice (3 faces)

Algorithm	**1×4f**	**1×5f**	**1×6f**	**2×3f**
Full-game FP	0.012	0.024	0.039	0.057
Full-game CFR	0.001	0.001	0.002	0.002
ReBeL FP	0.041	0.020	0.040	0.020
ReBeL CFR-D	0.017	0.015	0.024	0.017

Exploitability: Lower is better

Notes [3]:

- Top rows: tabular algorithms on full game (1024 iterations)
- Bottom rows: ReBeL with depth-2 subgames (1024 iterations)
- ReBeL achieves low exploitability without full-game traversal

What Makes These Results Remarkable

Superhuman HUNL performance [3]:

- Beat top human pro (Dong Kim): 165 ± 69 mbb/g
- Decisively beat benchmark AIs
- Fast decisions: ¡ 2 seconds per hand

Far less domain knowledge than prior AIs:

- DeepStack: hand-crafted features, expert-guided PBS sampling
- Libratus/Pluribus: heavy abstraction, game-specific optimizations
- ReBeL: learns through self-play, minimal abstractions

General framework:

- Works in both poker and Liar's Dice
- Provably converges to Nash (with perfect approximation) [3]
- First RL+Search method for imperfect information

Major step toward general-purpose multiagent learning. [3]

Relationship to CFR / Deep CFR / RPG

	Deep CFR	RPG	ReBeL
What's learned	Regret/advantage nets	Actor-critic (policy gradients designed to reduce regret)	Infostate values over PBS
Training	External sampling	On-policy self-play	Self-play + subgame solving
Search at test	None (use avg policy)	None (use current policy)	CFR with value net
Replay buffer	Yes (2 buffers)	No	No
Guarantees	None (approximate)	Tabular only	2p0s (with perfect approx)

Key distinctions:

- Deep CFR: neuralizes CFR's regret tables [1]
- RPG: connects policy gradients to regret minimization [2]
- ReBeL: learns PBS infostate values; uses CFR inside search [3]

Limitations and Open Questions

Computational cost:

- Belief computation + CFR subgame solving = expensive
- ReBeL used 128 machines $\times$ 8 GPUs for data generation [3]

Representation challenges:

- PBS input size grows with number of infostates
- Intractable in games like Recon Chess with huge state spaces [3]

Theoretical guarantees:

- Proven only for 2-player zero-sum
- General-sum / n-player less clear [3]

Practical concerns:

- Networks trained mostly on near-equilibrium play
- Adversarial opponents might exploit rarely-seen PBSs
- No test-time adaptation to specific opponent styles

Summary

ReBeL combines RL and search for imperfect-information games. [3]

Key ideas:

1. **Public belief states:** Convert to perfect-info game over beliefs
2. **Infostate values:** Learn supergradients of PBS value function
3. **Subgame solving:** Use CFR with value net at leaves
4. **Blueprint iteration:** Update blueprint in outer loop
5. **Safe search:** Sample random CFR iteration at test time

Mathematical results:

- Theorem 1: Infostate values form a supergradient of PBS value
- Theorem 2: Value network converges at $O(1/\sqrt{T})$
- Theorem 3: Safe search plays ϵ-Nash with $\epsilon = O(\delta + 1/\sqrt{T})$

Empirical achievements:

- Superhuman HUNL: beat top pro 165 ± 69 mbb/g
- Low exploitability in Liar's Dice
- Far less domain knowledge than prior poker AIs

References

[1] Brown, N., et al. (2019). Deep Counterfactual Regret Minimization. ICML.

[2] Srinivasan, S., Lanctot, M., et al. (2018). Actor-Critic Policy Optimization in Partially Observable Multiagent Environments. NeurIPS.

[3] Brown, N., Bakhtin, A., Lerer, A., & Gong, Q. (2020). Combining Deep Reinforcement Learning and Search for Imperfect-Information Games. NeurIPS.

Code:

- ReBeL (Liar's Dice): `https://github.com/facebookresearch/rebel`
- OpenSpiel: `https://github.com/deepmind/open_spiel`

Thank you!

Questions?