Game Abstraction Lecture 2

Tuomas Sandholm

ACTION ABSTRACTION

Action abstraction

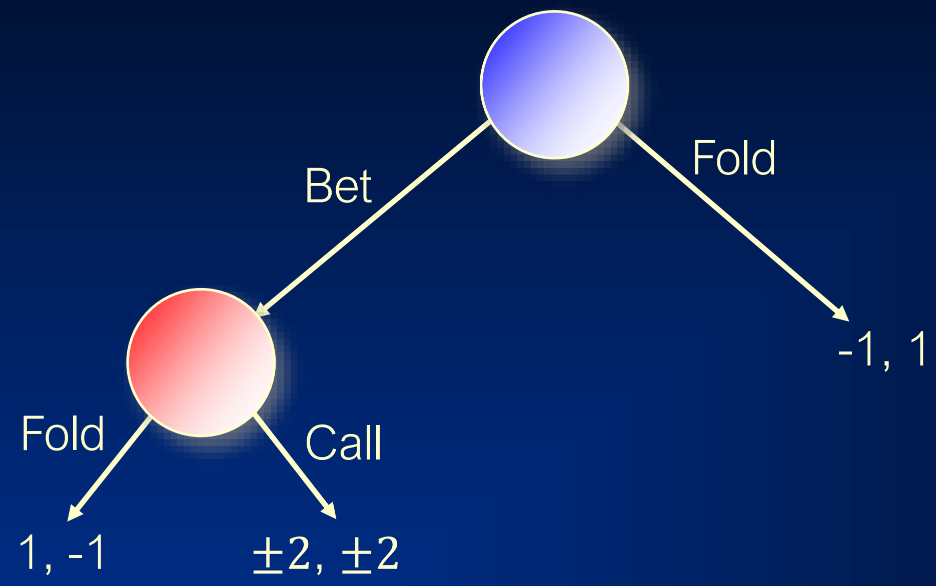
- Typically done manually
- Prior action abstraction algorithms for extensive games (even for just poker) had no guarantees on solution quality [Hawkin et al. AAAI-11, 12]
- For stochastic games there is an action abstraction algorithm with bounds (based on discrete optimization) [Sandholm & Singh EC-12]
- We present the first algorithm for parameter optimization for one player (in 2-player 0-sum games)
  - We use it for action size abstraction
  - Leverage regret matching (or CFR) warm starting by regret transfer

“Regret Transfer and Parameter Optimization *with Application to Optimal* **Action** *Abstraction*”

[Brown & Sandholm, *AAAI*-14]

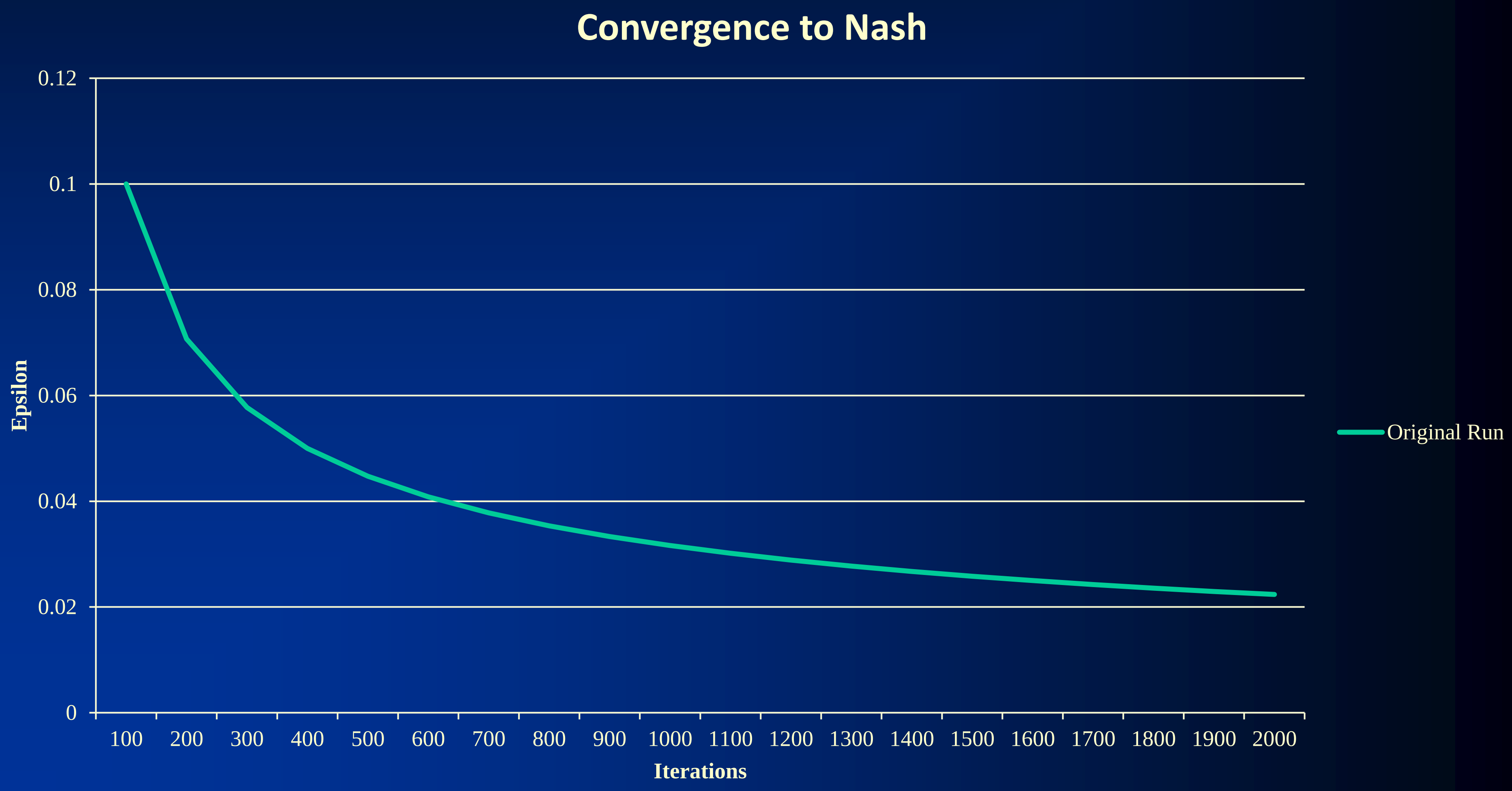
Setting: game payoffs change as we change the actions
(e.g., bet sizes in poker or bid sizes in auctions),
but the game topology doesn’t change

Motivation: A Simple Game

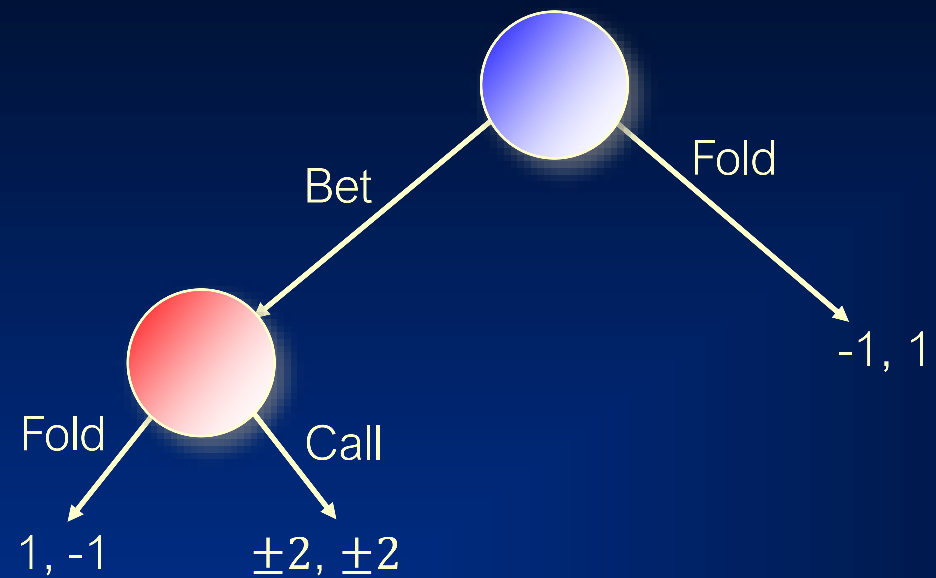


We solve with **No-Regret Learning**

Convergence to ϵ-Nash equilibrium

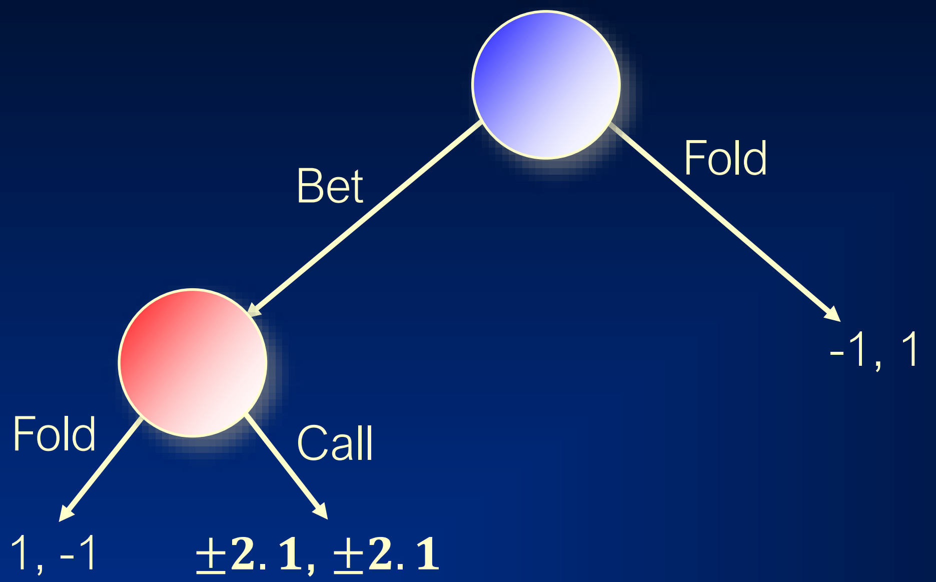


Motivation: A Simple Game



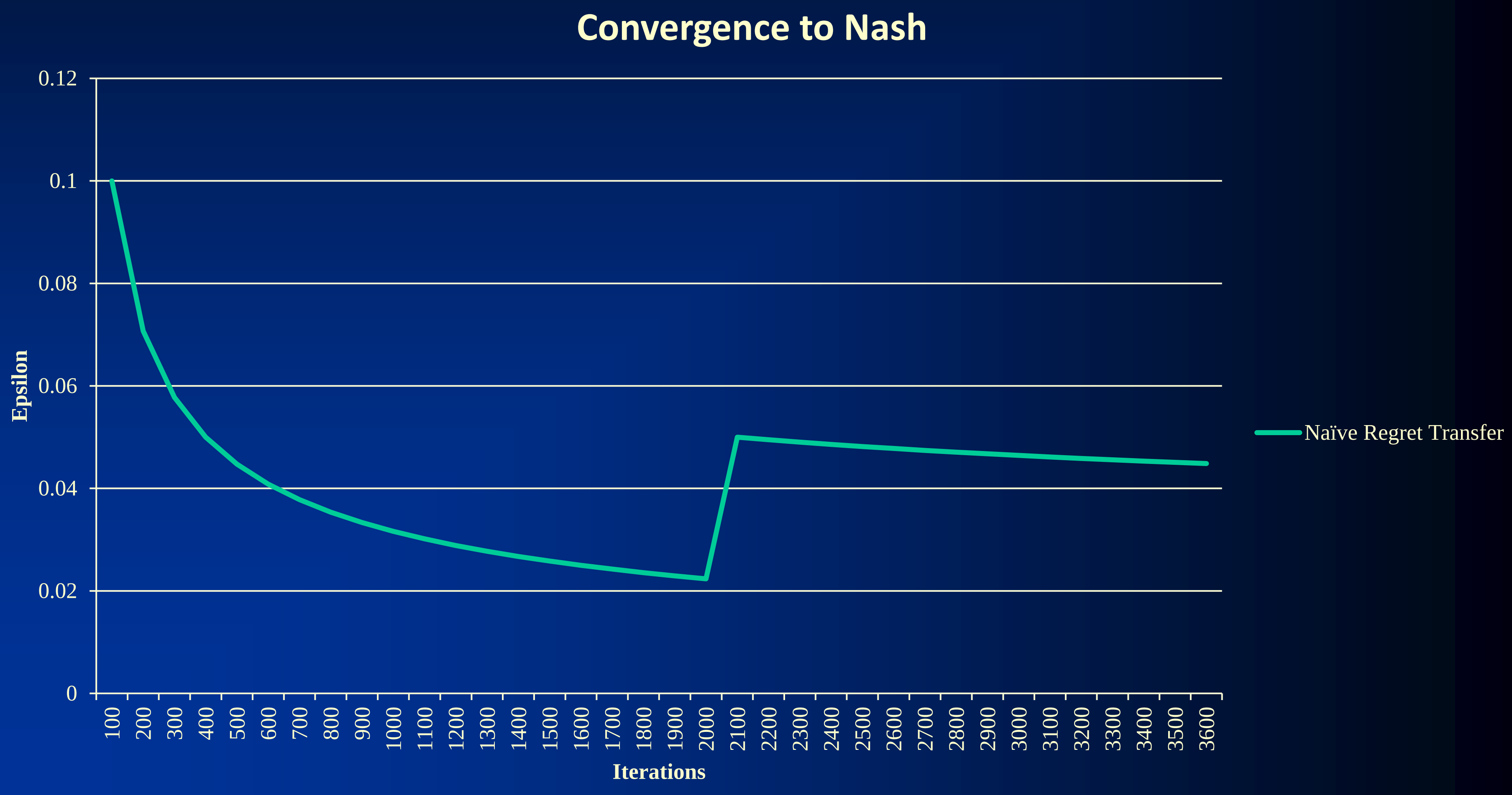
Suppose we change the Bet-Call payoff part-way through our run

Motivation: A Simple Game

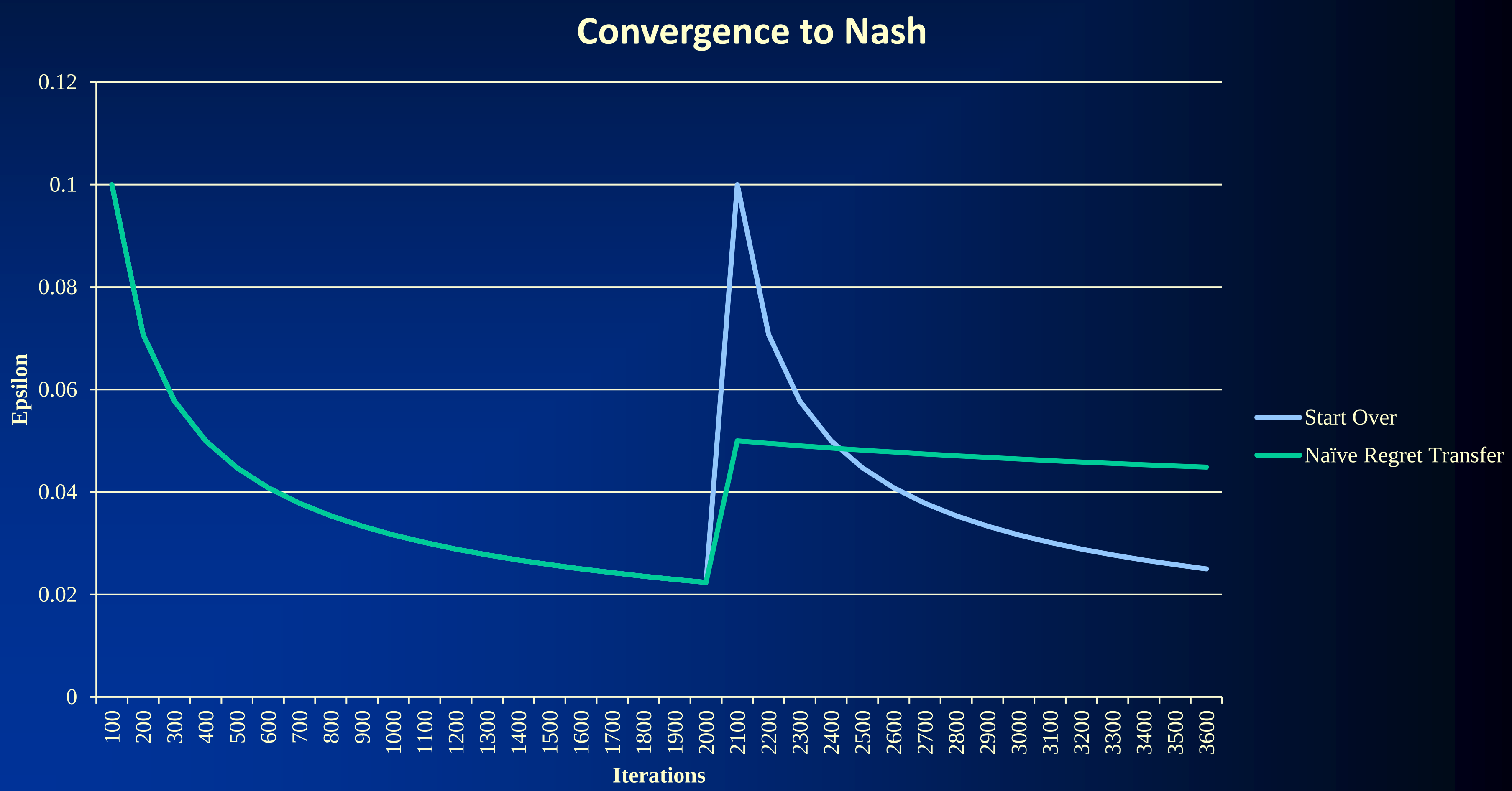


$$\delta = 2.1 - 2 = 0.1$$

Convergence to ϵ-Nash equilibrium

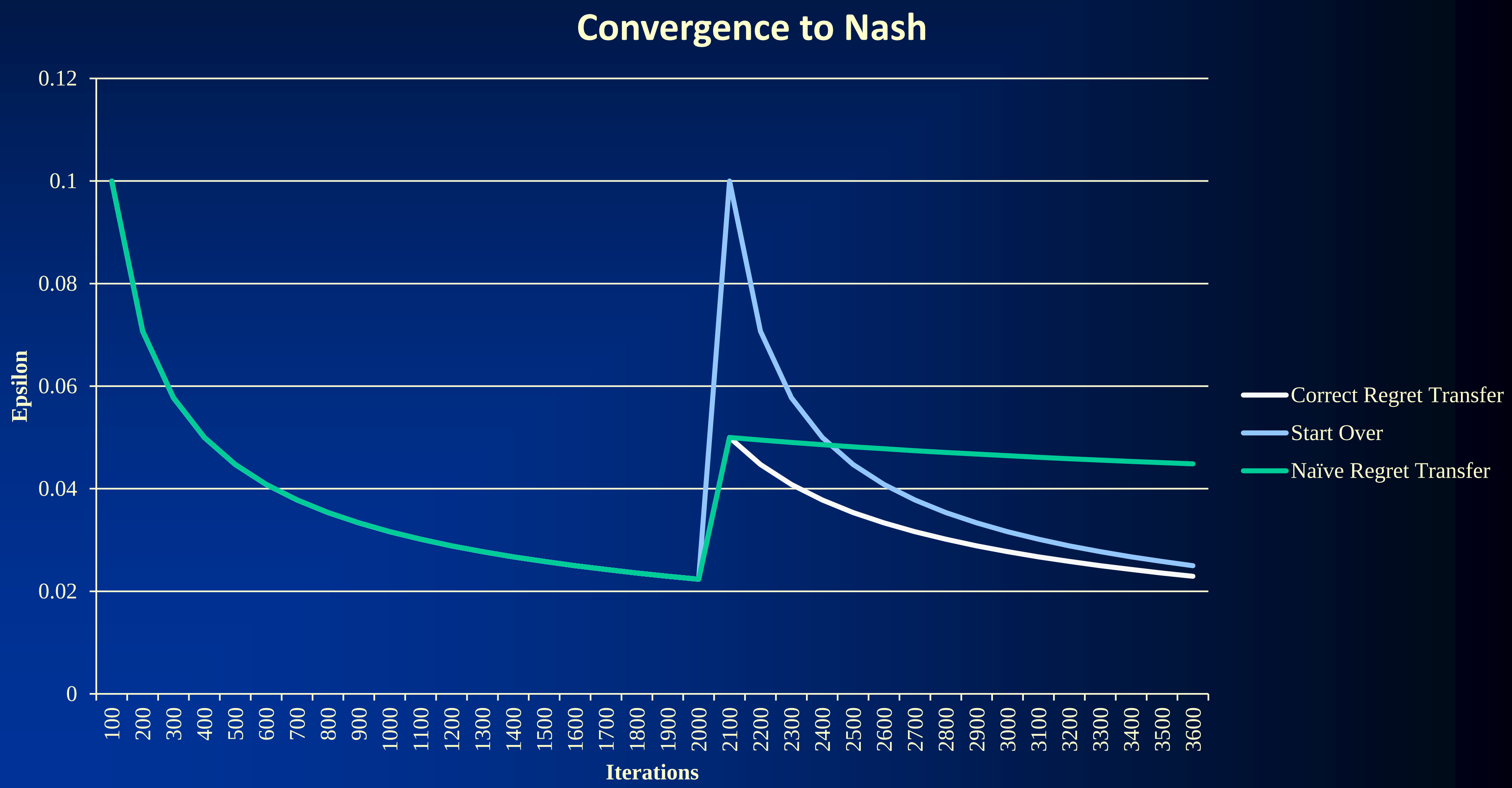


Convergence to ϵ-Nash equilibrium



Convergence to ϵ-Nash equilibrium

$$\text{Scale Amount:}O\left(\frac{1}{\left(1+\delta\sqrt{T}\right)^2}\right)$$



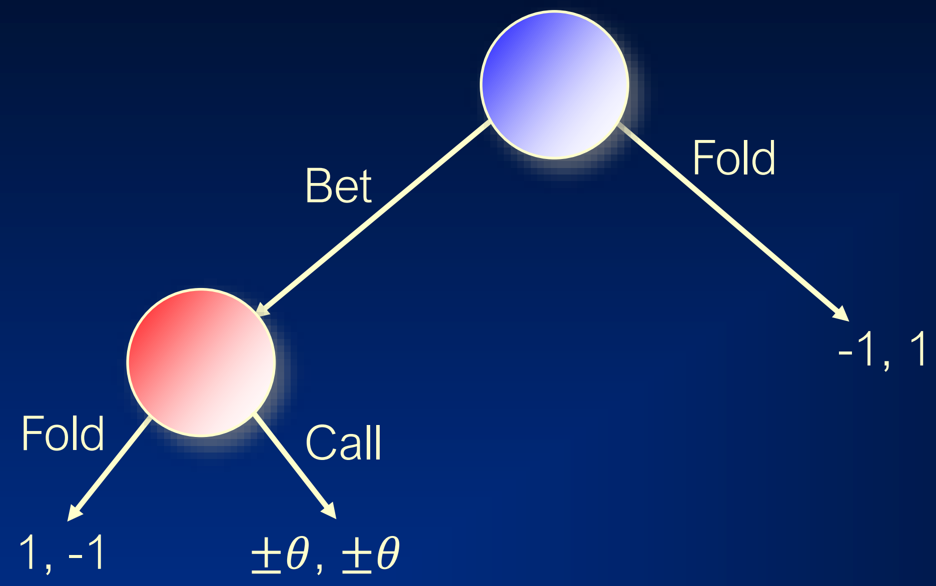
Optimal Parameter Selection

- Action abstraction: action size selection
  - *(Optimizing together with probabilities would be quadratic)*

- Each abstraction has a Nash equilibrium value that isn't known until we solve it

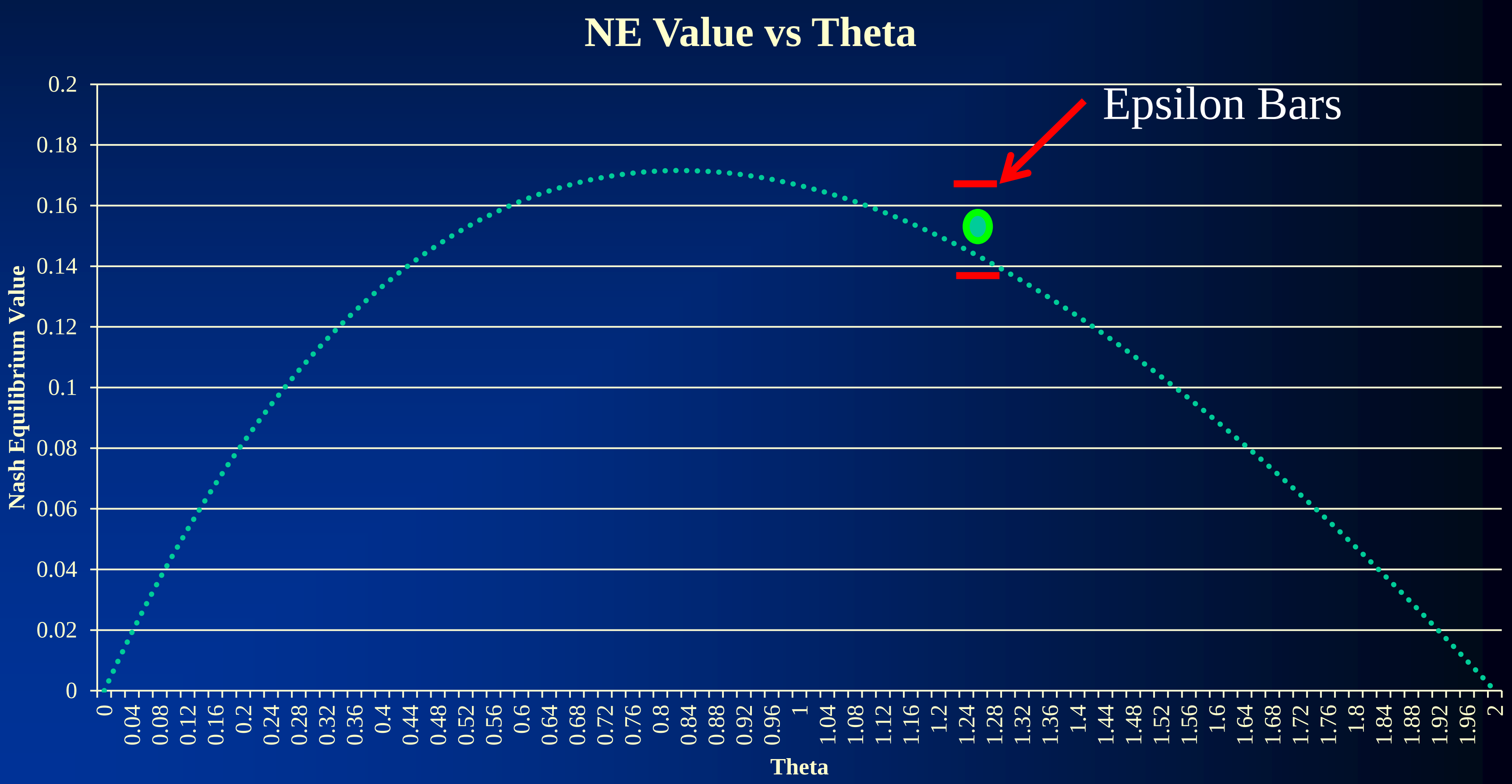
- We want to pick the optimal action abstraction (one with highest equilibrium value for us)

Optimizing A Simple Game

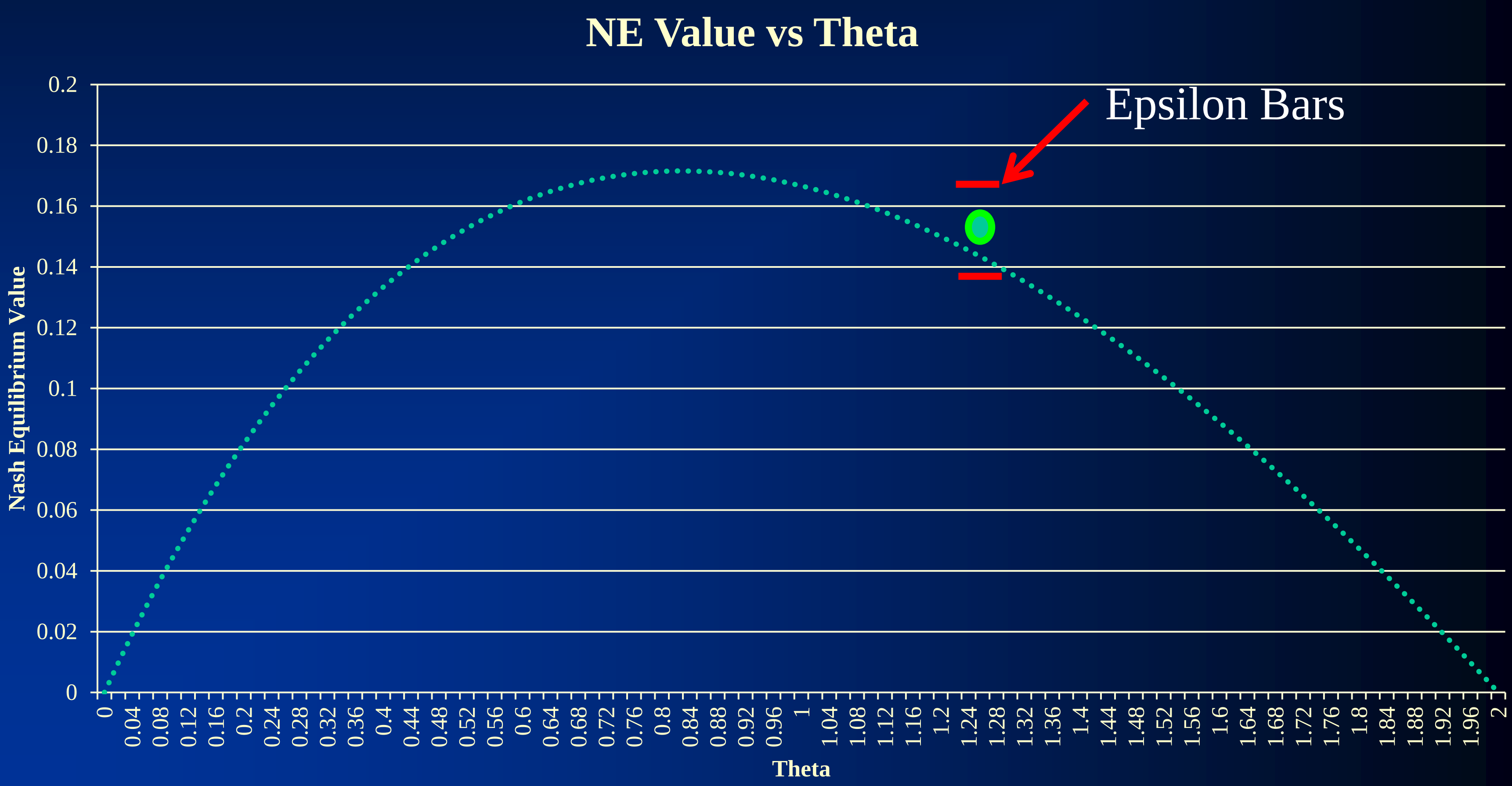


What is the optimal value of θ for P1?

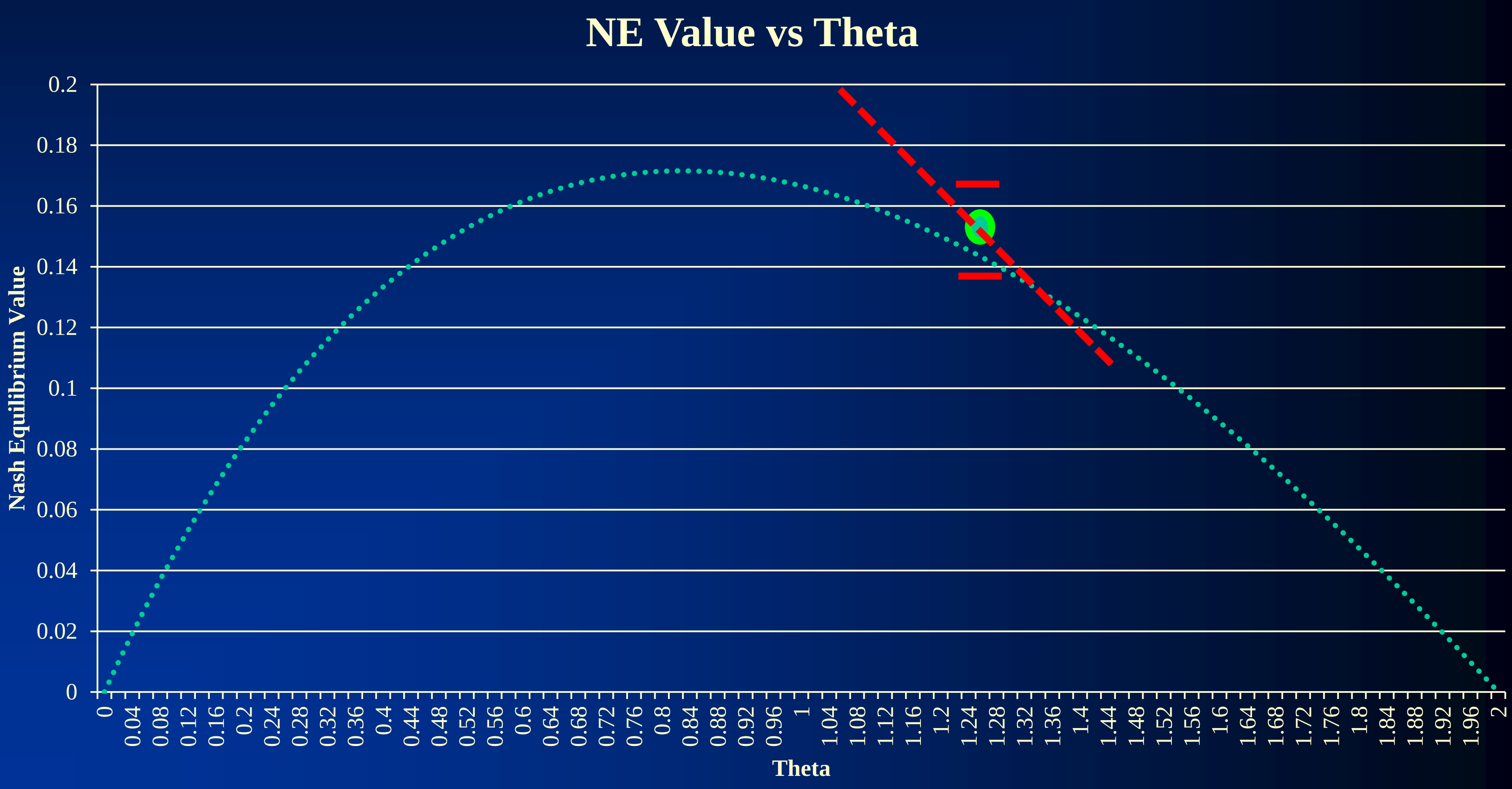
Step 1:Do K_1 iters of No-Regret Learning



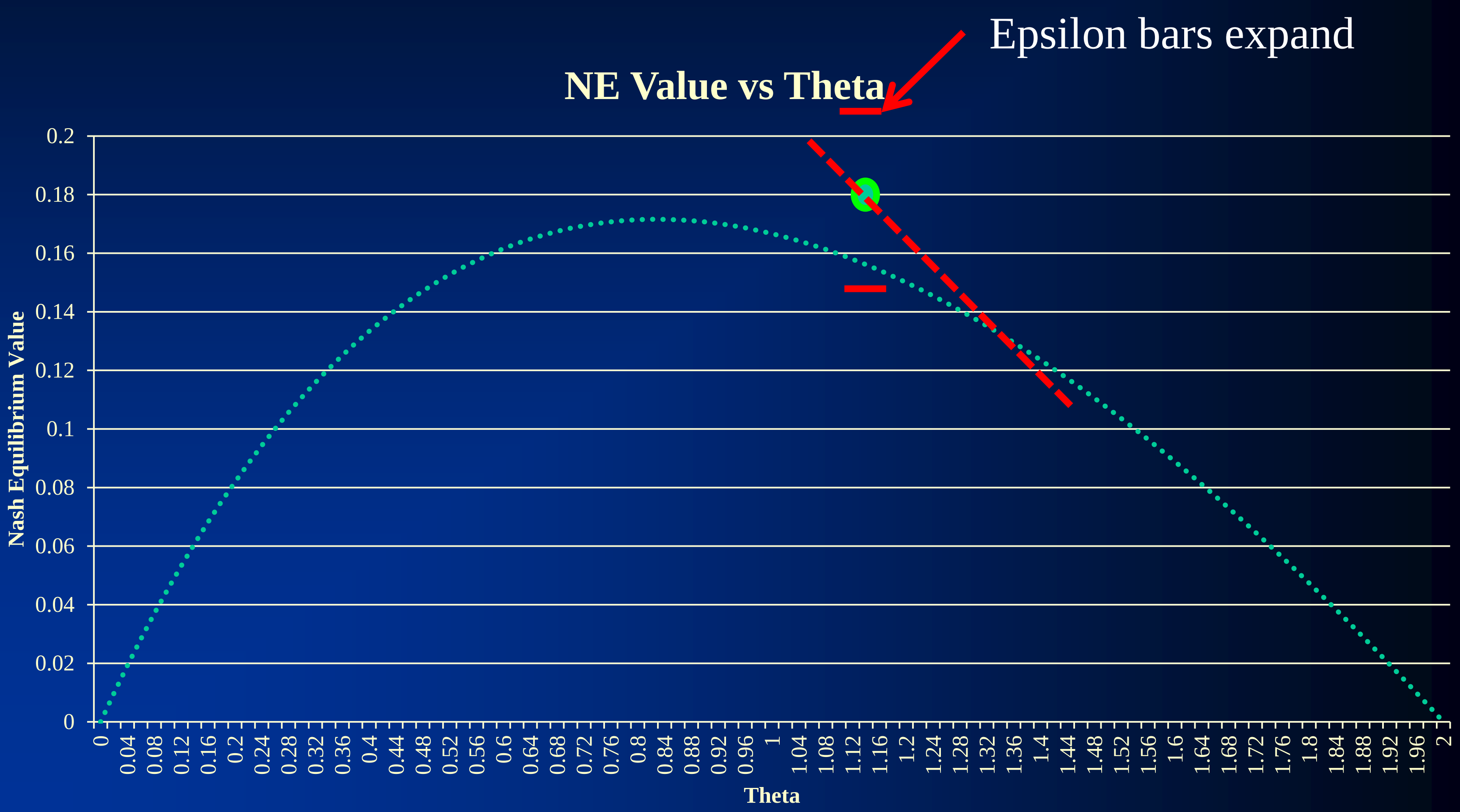
Step 2:Estimate Gradient



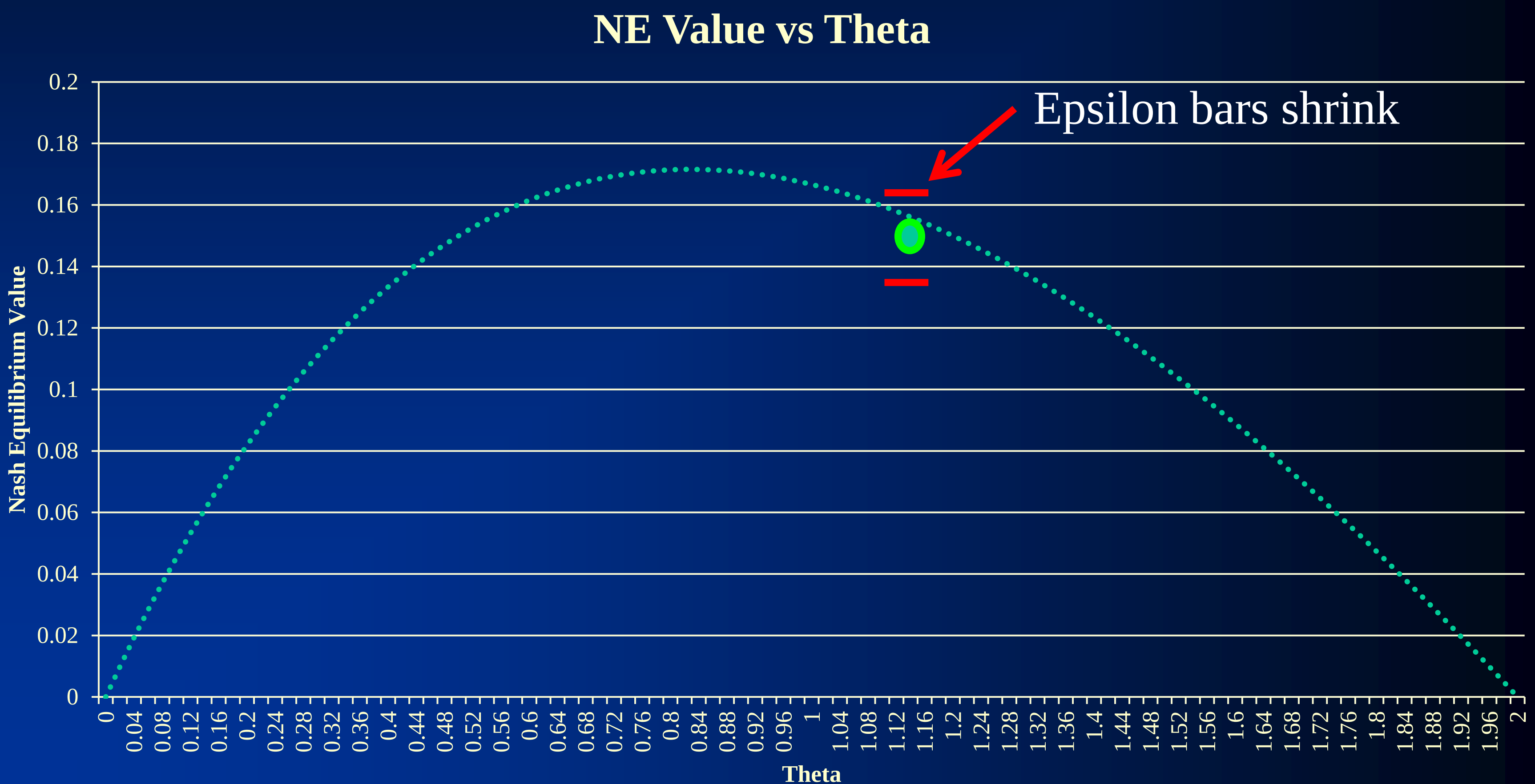
Step 3:Move Theta, Transfer Regret (deweight regrets and strategies for averaging)



Step 4: Do K_2 iters of No-Regret Learning



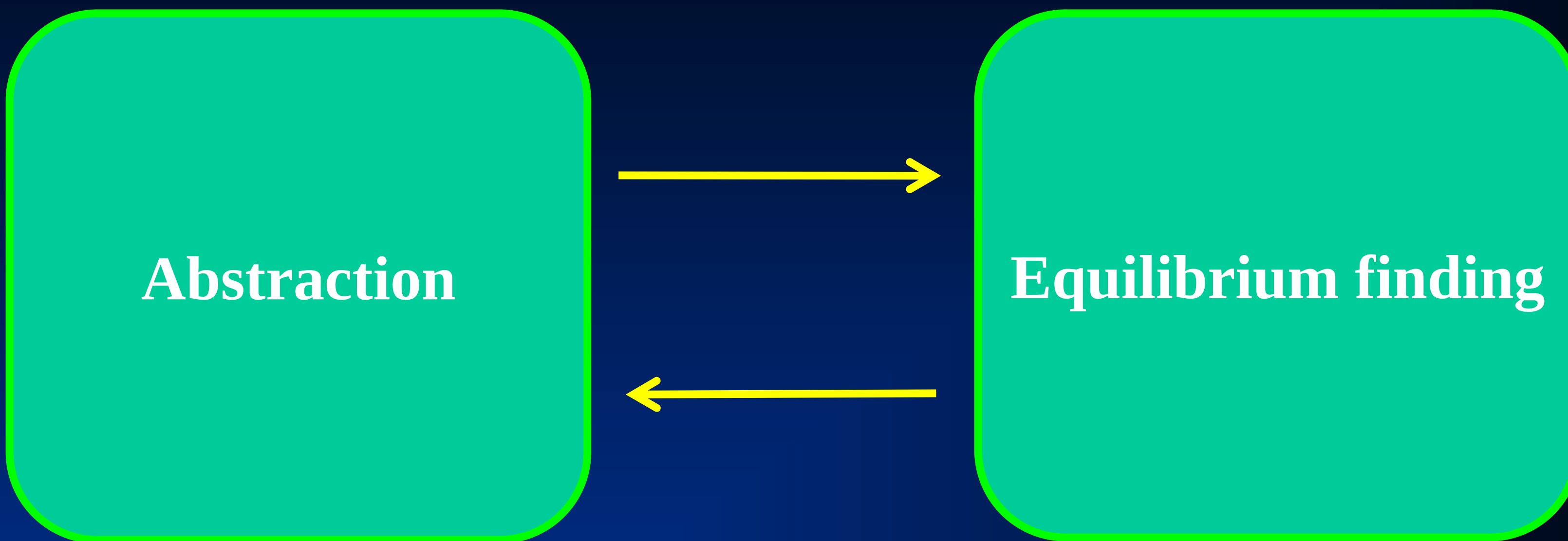
Repeat to convergence



- We have applied this to
  - No-Limit Texas Hold’em (1 bet being sized in that experiment), and
  - Leduc Hold’em (2 bet sizes being sized simultaneously in that experiment)

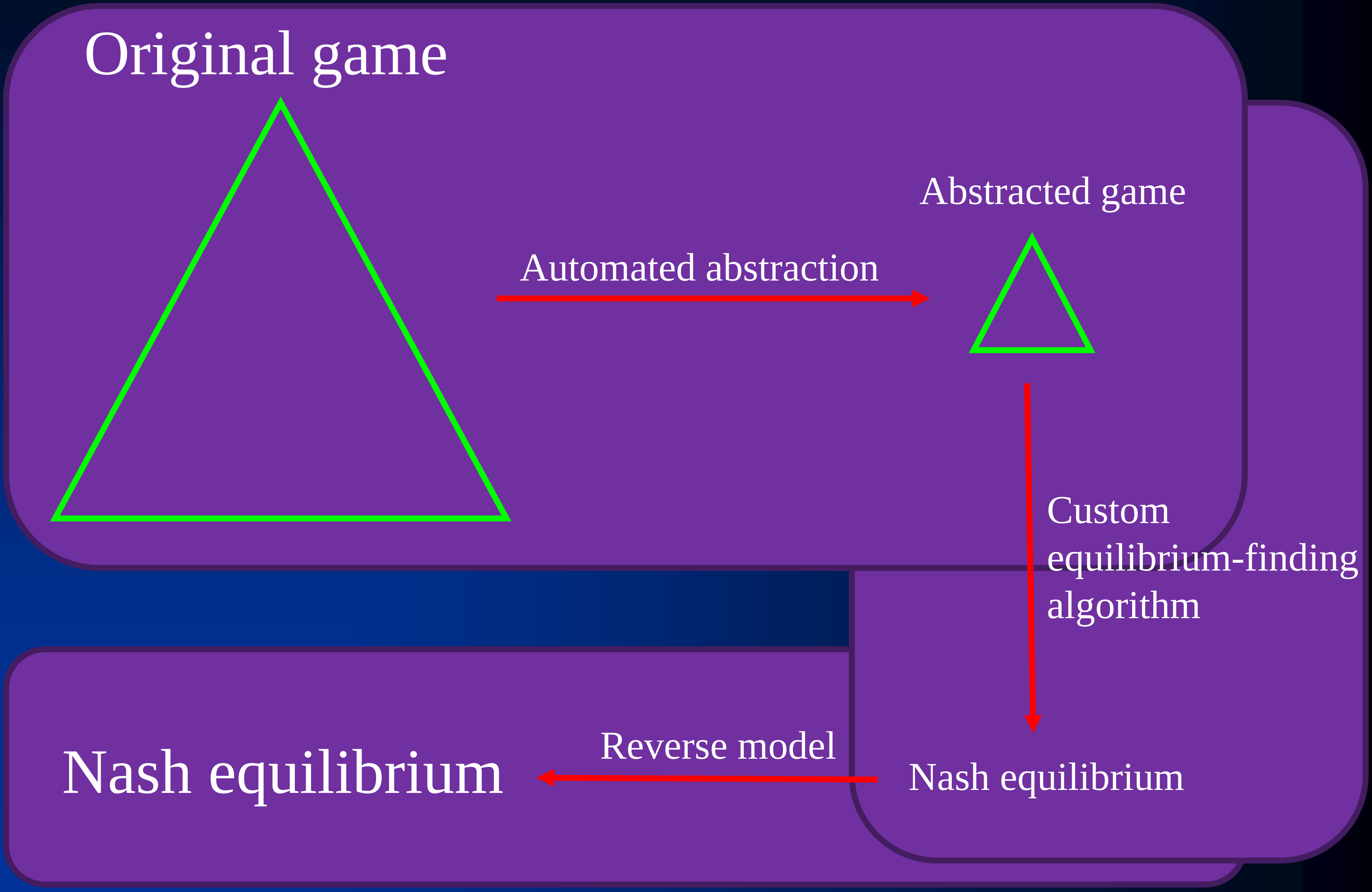
SIMULTANEOUS ABSTRACTION AND EQUILIBRIUM FINDING

Strategy-based abstraction

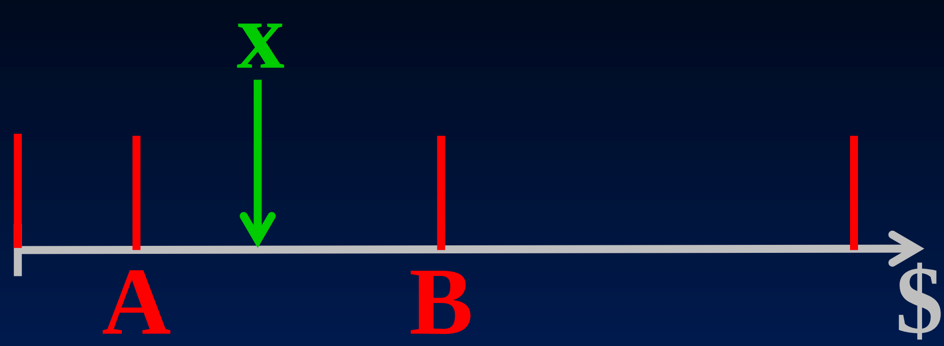


- So far, we have done this for adding actions into the abstraction (and warm starting via discounting) ["Simultaneous Abstraction and Equilibrium Finding in Games", Brown & Sandholm, *IJCAI*-15]

REVERSE MAPPING


Original game
Abstracted game
Automated abstraction
Custom equilibrium-finding algorithm
Reverse model
Nash equilibrium
Nash equilibrium

Action translation



f(x) ≡ probability we map x to A

<u>Desiderata about f</u>

1. f(A) = 1, f(B) = 0
2. Monotonicity
3. Scale invariance
4. Small change in x doesn't lead to large change in f
5. Small change in A or B doesn't lead to large change in f

[Ganzfried & Sandholm, *IJCAI*-13]

<u>"Pseudo-harmonic mapping"</u>

- f(x) = [(B-x)(1+A)] / [(B-A)(1+x)]
- Derived from Nash equilibrium of a simplified no-limit poker game
- Satisfies the desiderata
- Much less exploitable than prior mappings in simplified domains
- Performs well in practice in no-limit Texas Hold'em
  - Significantly outperforms best prior reverse mapping, randomized geometric

LOSSY ABSTRACTION
WITH EXPLOITABILITY BOUNDS

Game abstraction is nonmonotonic

Defender

Attacker		A	Between	B
	A	0, 2	1, 1	2, 0
	B	2, 0	1, 1	0, 2

In each equilibrium:

- Attacker randomizes 50-50 between A and B
- Defender plays A w.p. p, B w.p. p, and Between w.p. $1-2p$
- There is an equilibrium for each $p \in [0, ½]$

An abstraction:

	A	Between	B
A	0, 2	1, 1	2, 0

Defender would choose A, but that is far from equilibrium in the original game where attacker would choose B

Coarser abstraction:

	Between	B
A	1, 1	2, 0

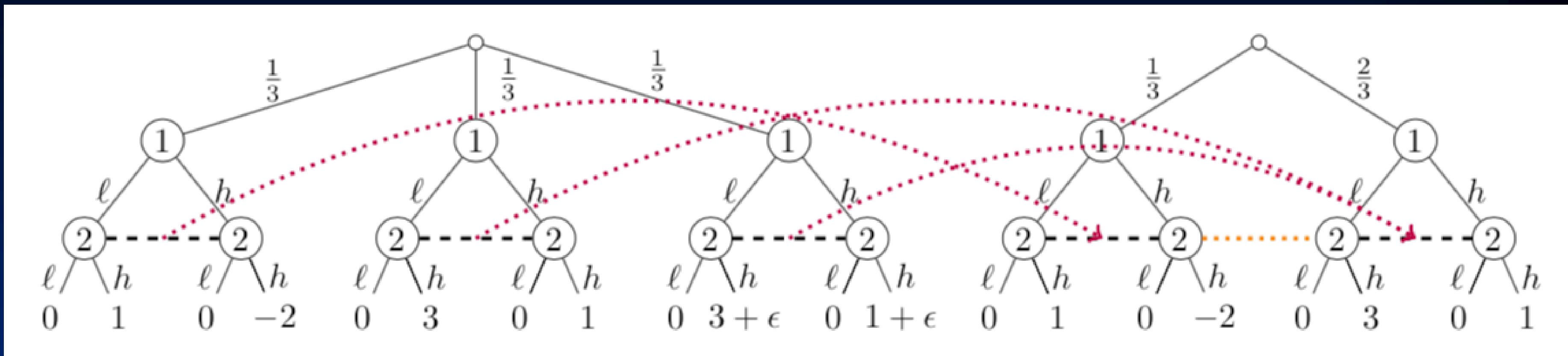
Defender would choose Between. That is an equilibrium in the original game

- Such "abstraction pathologies" also in small poker games [Waugh *et al.*, *AAMAS*-09]

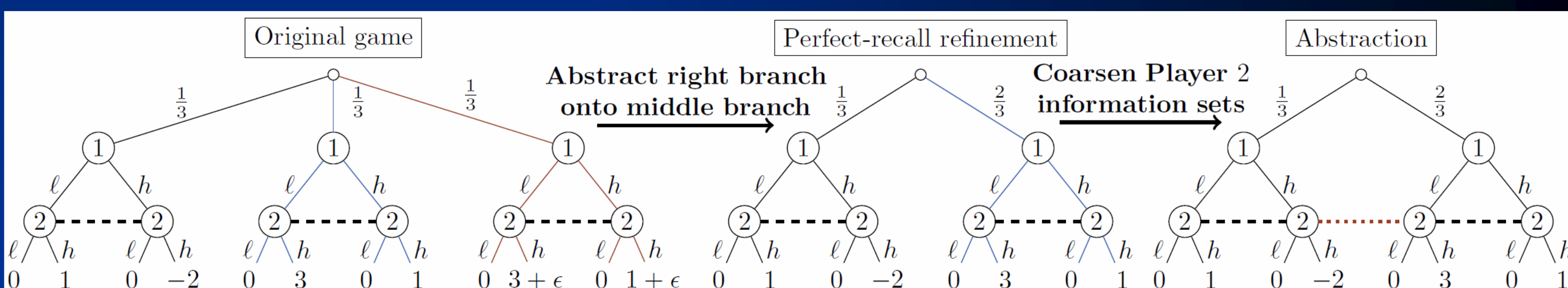
Can we get bounds on exploitability despite abstraction pathologies?

- First answer: Yes, in stochastic games [Sandholm & Singh, *EC*-12]
- I'll present a unified abstraction framework for extensive-form games [Kroer & Sandholm, *NeurIPS*-18]
  - n-player, general-sum game
  - Generalizes and improves over prior work [Lanctot *et al.*, *ICML*-12; Kroer & Sandholm, *EC*-14, *EC*-16]
- Applies to modeling also

Abstraction example



We think of this as two steps, which can be analyzed separately:



Lifted strategies

- Given a strategy profile σ' for the abstraction, a lifted strategy is a profile σ s.t. for each abstract I' and corresponding I:
  - Probability mass on abstract action is spread any way across the set of actions that map to it
  - Formally, $\sigma'(I', a') = \sum_{a \in g^{-1}(a')} \sigma(I, a)$

Abstraction theorem

[Kroer & Sandholm, *NeurIPS*-18]

- Given:
  - a perfect-recall game,
  - an acyclic abstract game,
  - a mapping between them that satisfies our mild, natural assumptions, and
  - an ϵ-Nash equilibrium in the abstract game
- Then: Any lifted strategy is an ϵ'-Nash equilibrium in the original game, where $\epsilon' = \max_i \epsilon_i'$ and

$$\epsilon_i' = \epsilon + mapping\ errori + refinement\ errori$$

Error from mapping real game onto perfect-recall refinement of abstract game

Error between perfect-recall refinement of abstract game and abstract game

- Advantages over prior work:
  - Exact decomposition of error
  - Equilibrium in abstract game doesn't have to be exact
  - Doesn't make restrictive assumption of prior work
  - Exponentially better bound than Lanctot *et al.* [*ICML*-12]
  - We also derive a similar result for solution to abstract game with bounded counterfactual regret (gain at most ϵ_a by switching to any action a)

Mapping error$_i$

Sum of

- Payoff error:
  - Expectation over leaf nodes in real game

    of utility difference between real leaf and the node it maps onto
- Distribution error:
  - Sum over leaf nodes in abstraction

    of difference in probability of reaching abstract leaf and sum of reach probabilities on real leaves that map to it

Refinement error$_i$

- Sum over infosets I_p in the perfect-recall refinement of the abstraction (let I' be the corresponding abstract infoset):

  Sum of:

  - Payoff error:
    - Expectation over leaves under I'

      of utility difference compared to corresponding leaf under I_p
  - Distribution error:
    - Sum over leaves under I_p

      of difference in probability of reaching refinement leaf from I_p
      versus sum of reach probabilities on abstract leaves from I'

Future research on lossy abstraction with exploitability bounds

- The distribution error terms in our decomposition are in general not computable *ex ante* (i.e., before running a solver on the abstract game)
  - Because they can depend on players' strategies
    - Prior approaches required that for pairs of leaves mapped to each other, the leaves have the same sequence of information-set-action pairs leading to them in the abstraction
    - Under that assumption, we can compute *ex ante* bounds (take max's)
- Idea: Find other specialized but practical game classes where game structure can be leveraged to give computable *ex ante* bounds
    - One approach: Our decomposition relies on utility differences (not absolute value thereof as prior approaches did), so structured game classes could potentially even cancel out error terms

Conclusions on this lecture

- Domain-independent techniques
- First action abstraction algorithm with optimality guarantees: iterative action size vector changing
- Simultaneous abstraction and equilibrium finding
- Reverse mapping: “pseudoharmonic”
- Lossy abstraction with exploitability bounds
- Future research
  - Applying these techniques to other domains
  - Better algorithms within our lossy-abstraction-with-bounds framework (or different such framework to be developed in the future)