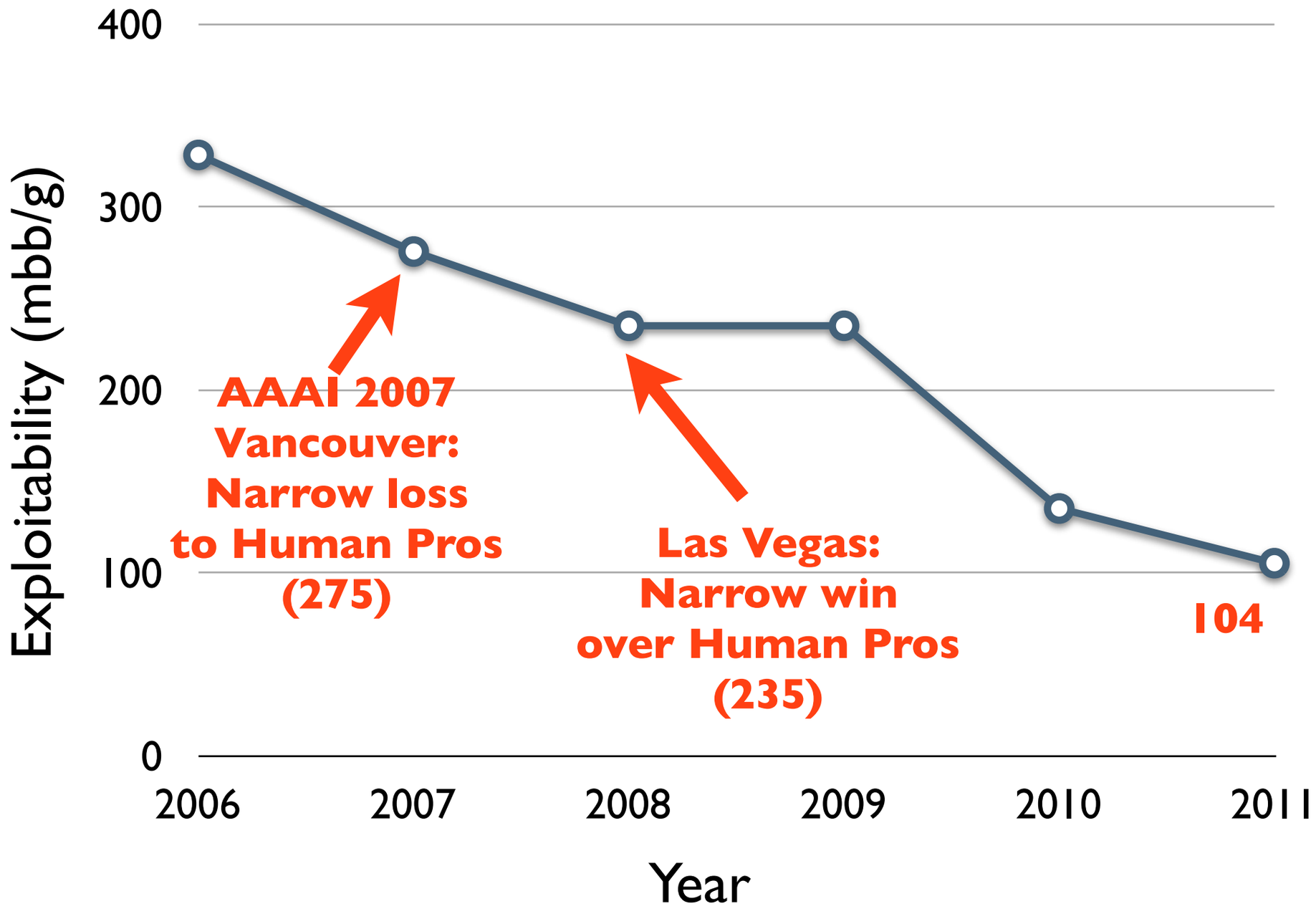
Finding Optimal Abstract Strategies in Extensive-Form Games

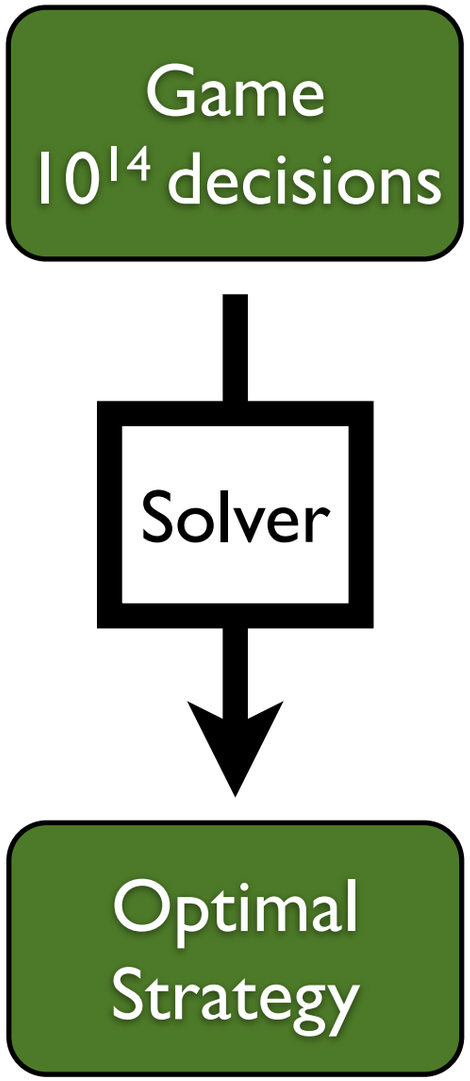
Mike Johanson, Nolan Bard,
Neil Burch, Michael Bowling
University of Alberta, Canada

July 25th, 2012 :: AAAI 2012, Toronto

2-Player Limit Texas Hold'em Poker: Distance from Perfect Play



Abstraction-Solving-Translation



Goal:
We want to learn a strategy **σ** (or, in RL, a policy **π**) that chooses actions.

Exploitability:
Expected loss against a perfect adversary.

Nash Equilibrium:
Unexploitable - expected loss of $0 per game. An **optimal strategy**. We want to approximate this.

Abstraction-Solving-Translation

Problem:
The game has 10^{14} information sets. Far too large to solve!

With current techniques, this would take 4 petabytes of RAM and thousands of CPU-years!

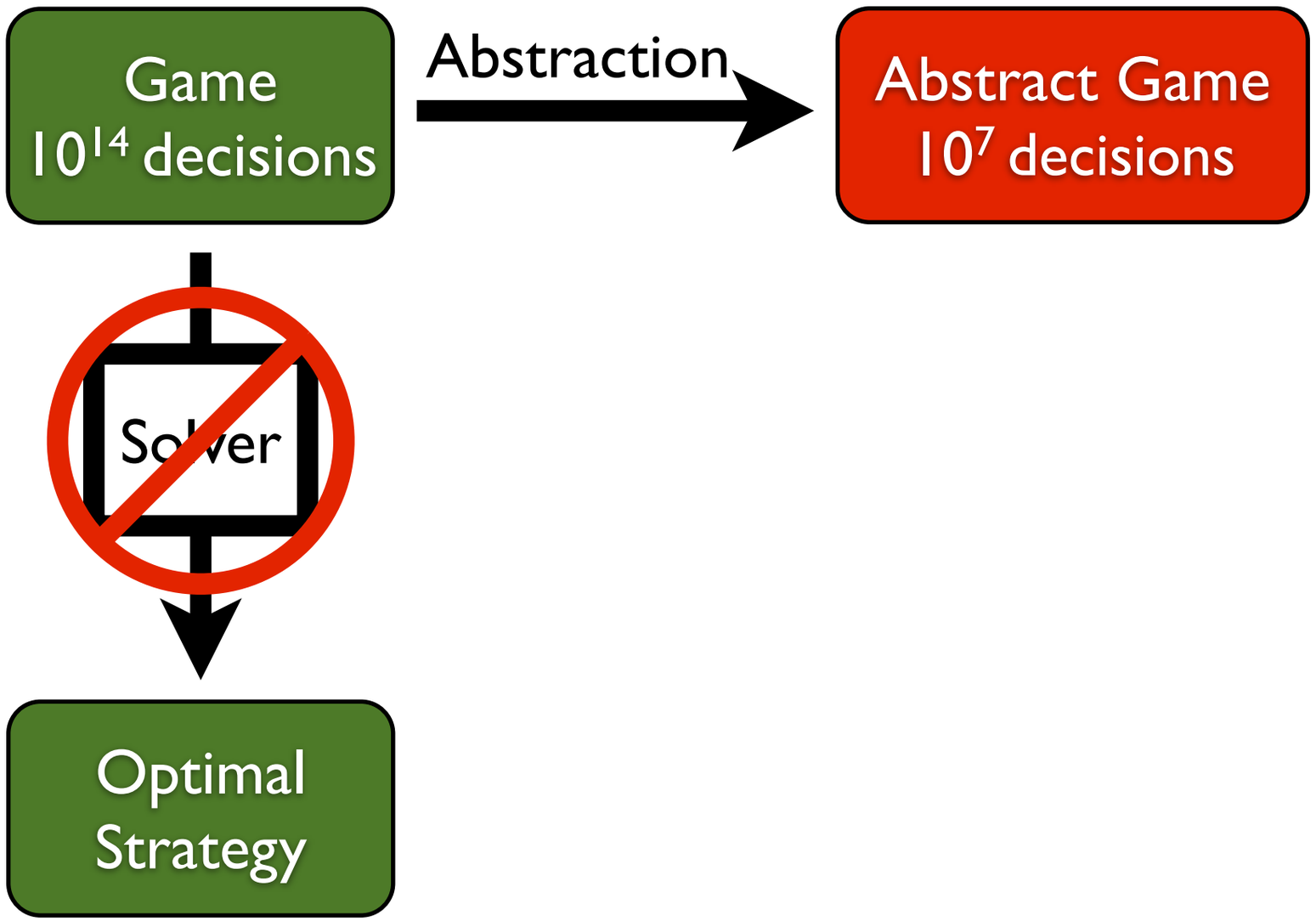
Abstraction-Solving-Translation

Problem:
The game has 10^{14} information sets. Far too large to solve!

With current techniques, this would take 4 petabytes of RAM and thousands of CPU-years!

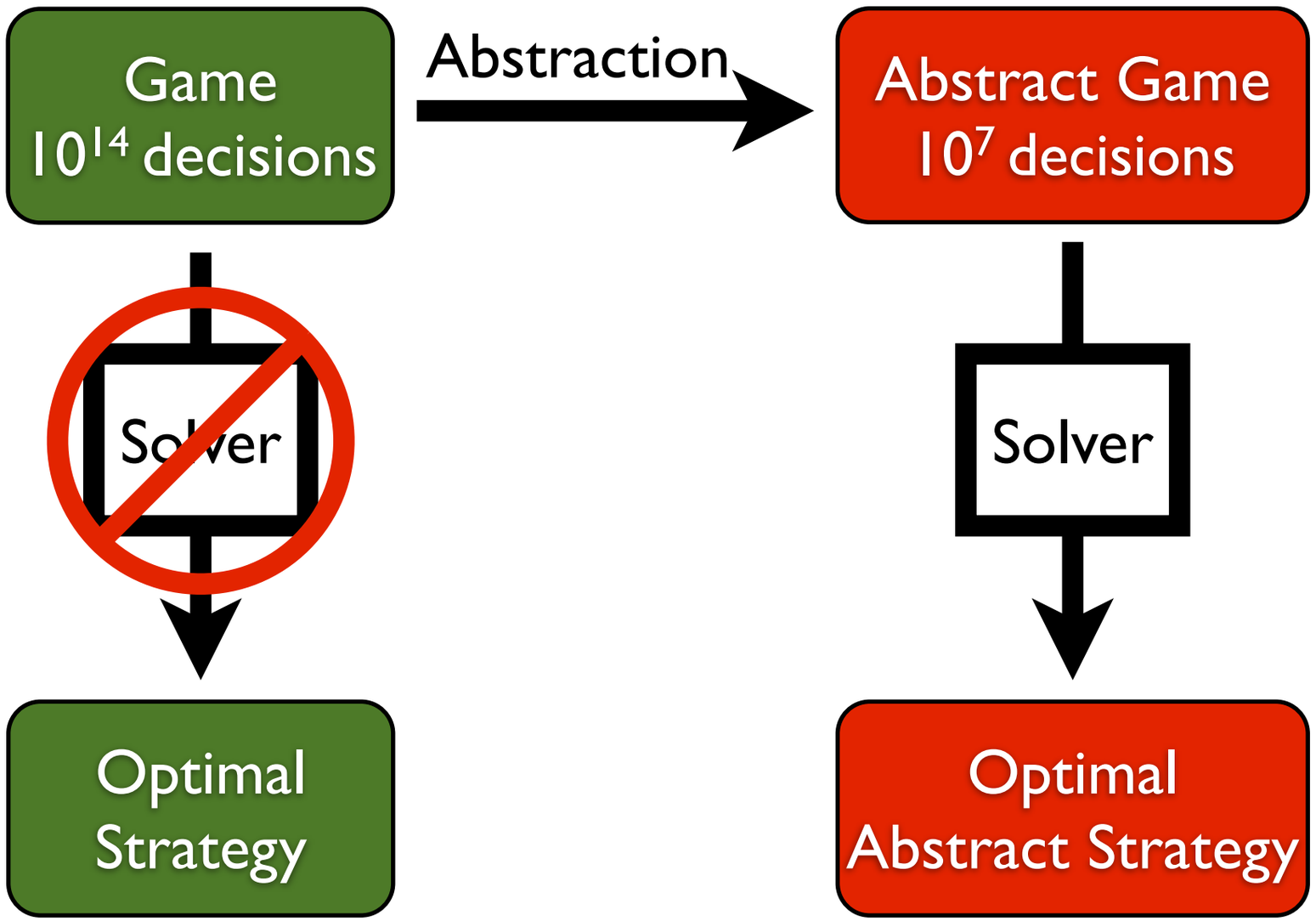
If you have four petabytes of RAM, we should talk!

Abstraction-Solving-Translation



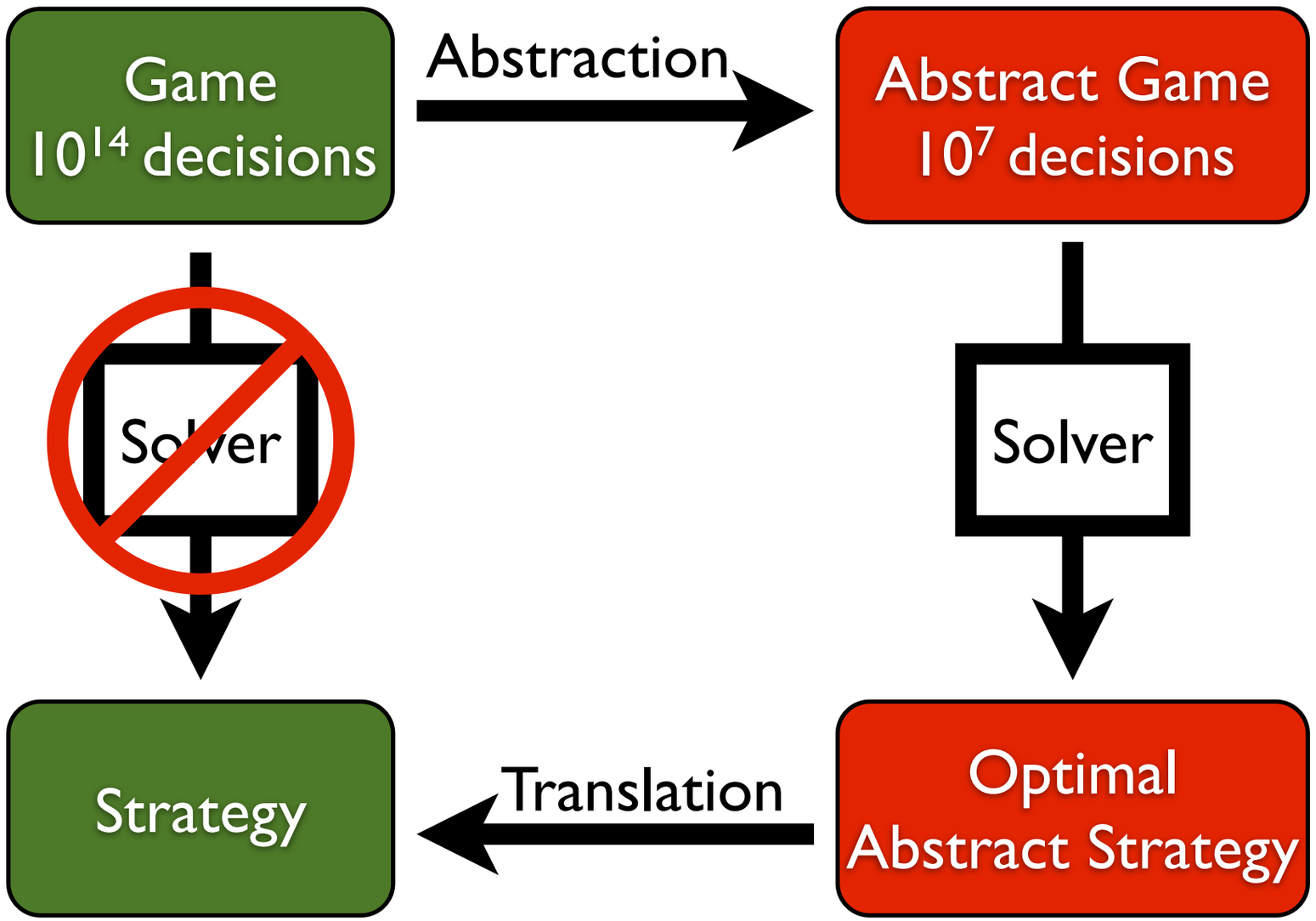
Workaround:
Use state-space abstraction to make a smaller game that we can solve.

Abstraction-Solving-Translation



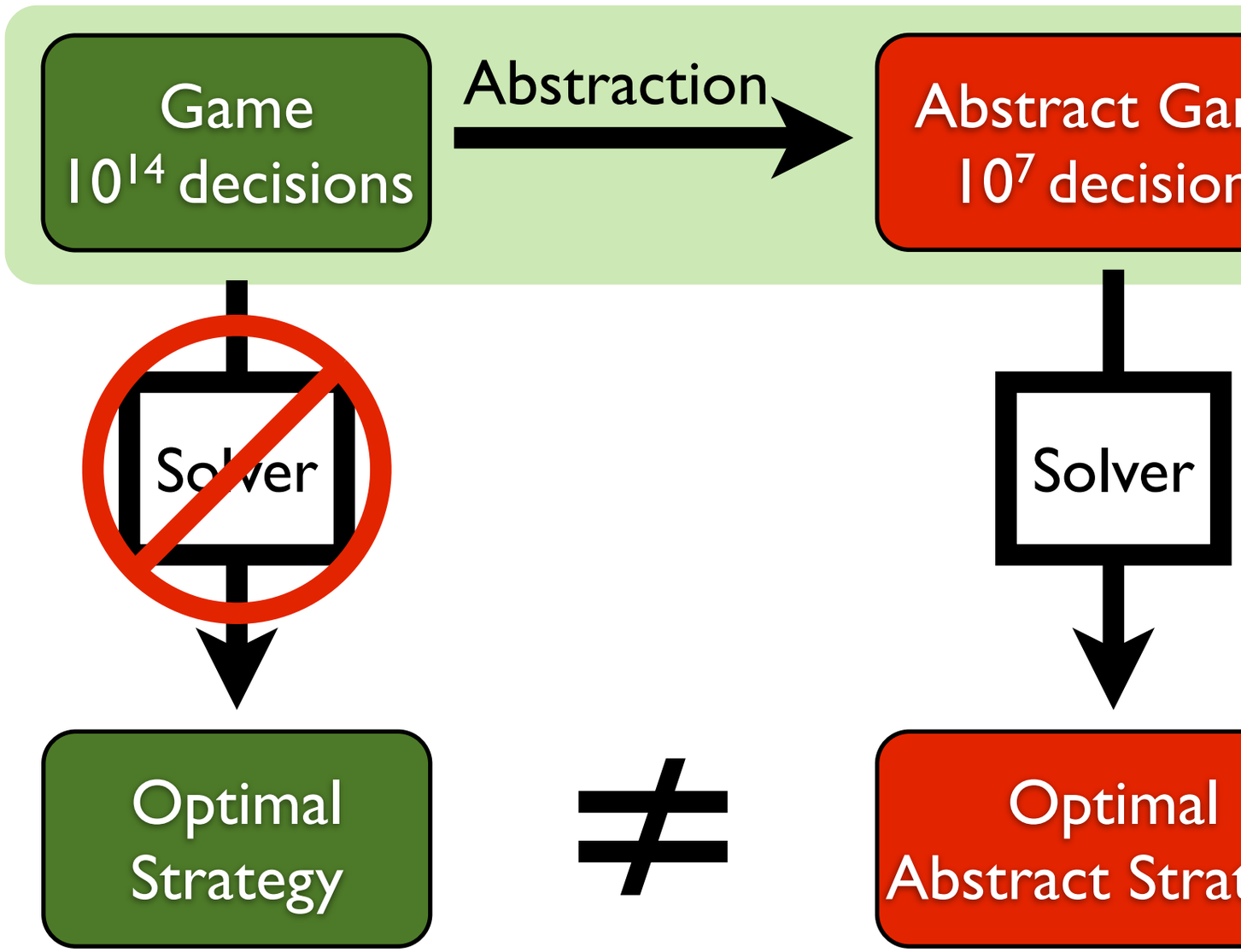
Solving:
Use a game-solving algorithm to find an optimal strategy for the abstract game.

Abstraction-Solving-Translation



Solving:
Use a game-solving algorithm to find an optimal strategy for the abstract game.

Abstraction-Solving-Translation

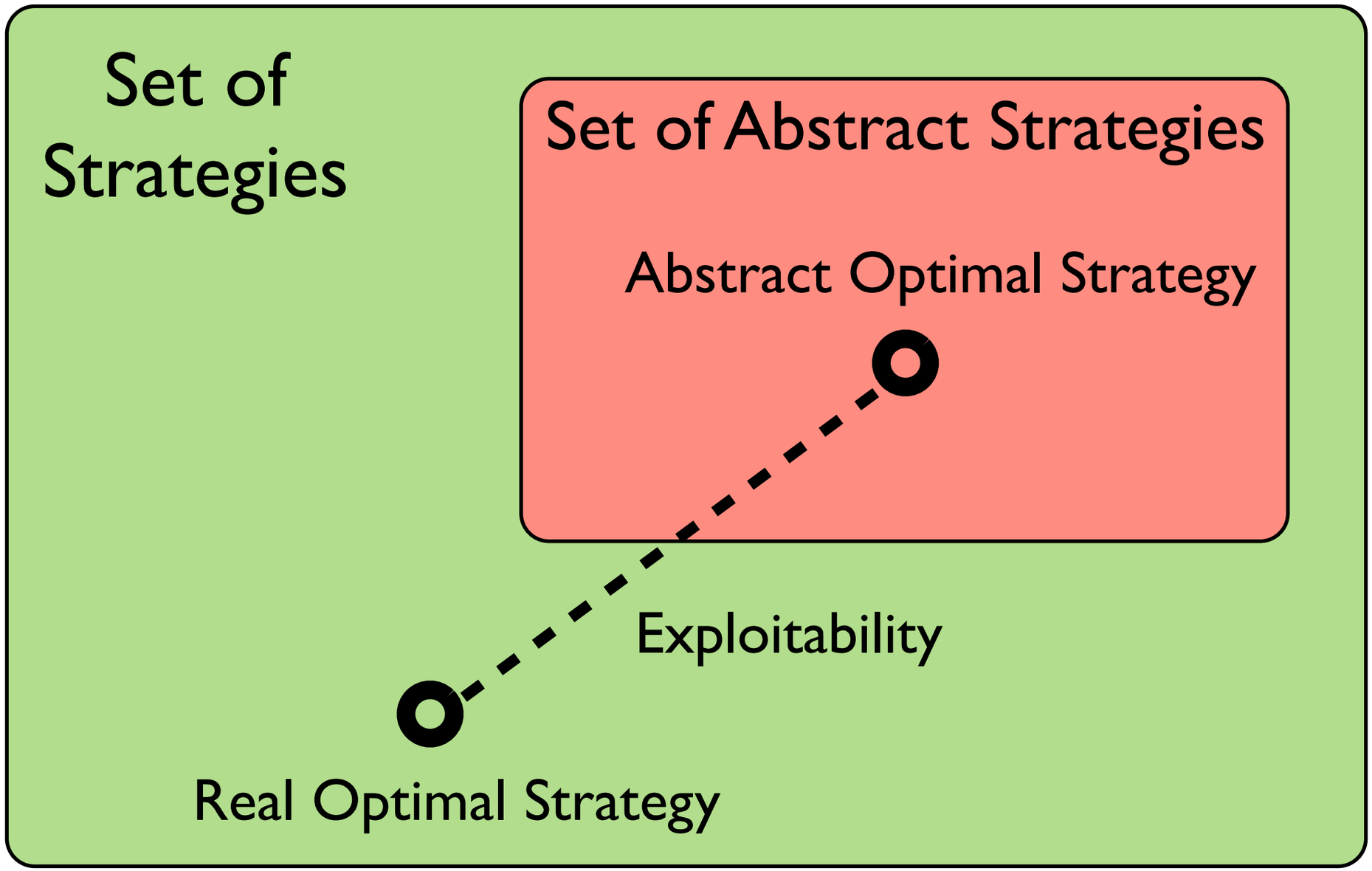


Two Types of Loss:

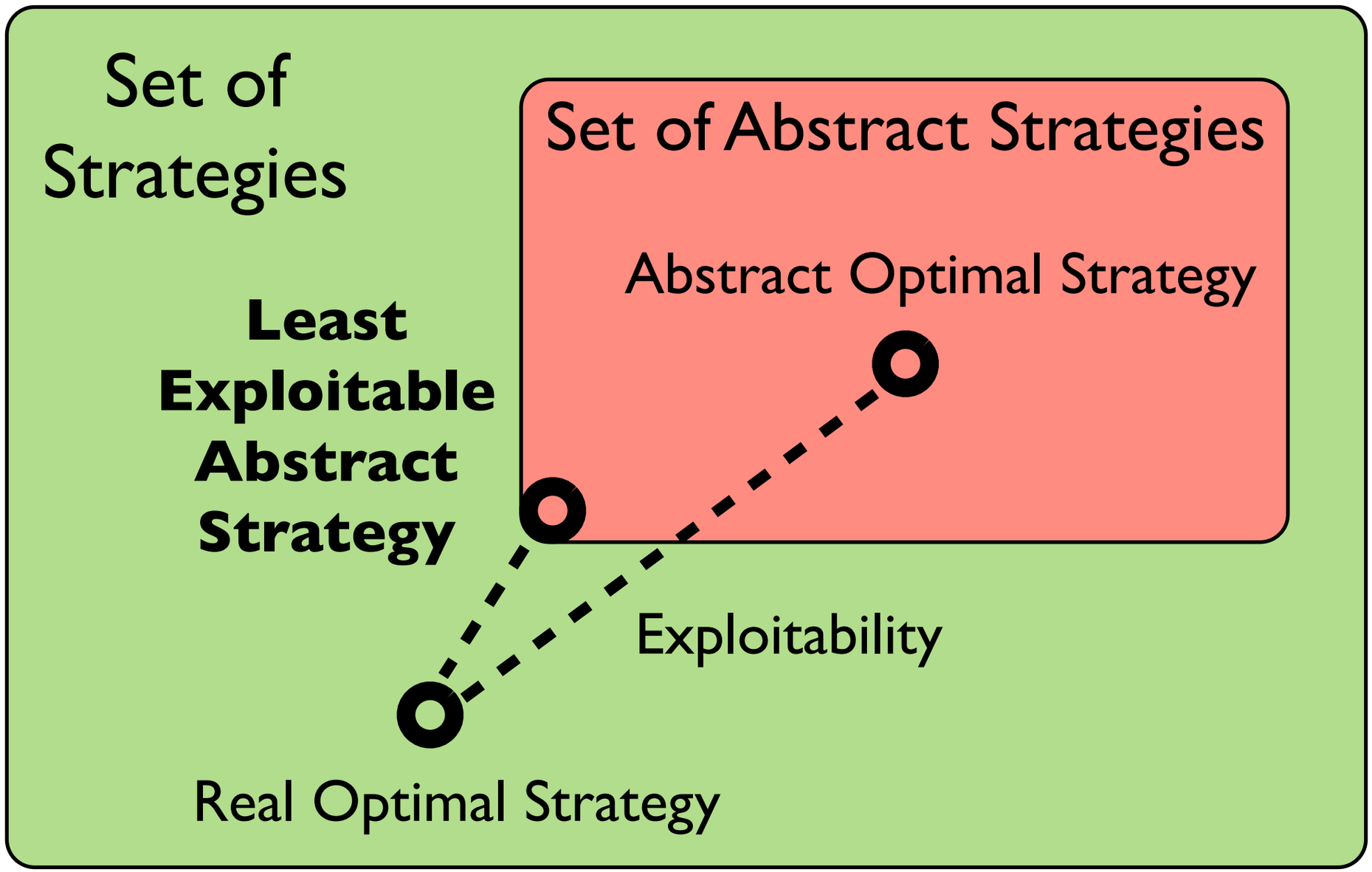
Lossy abstraction. May not be possible to represent an optimal strategy.

Other abstract strategies might be better in the real game!

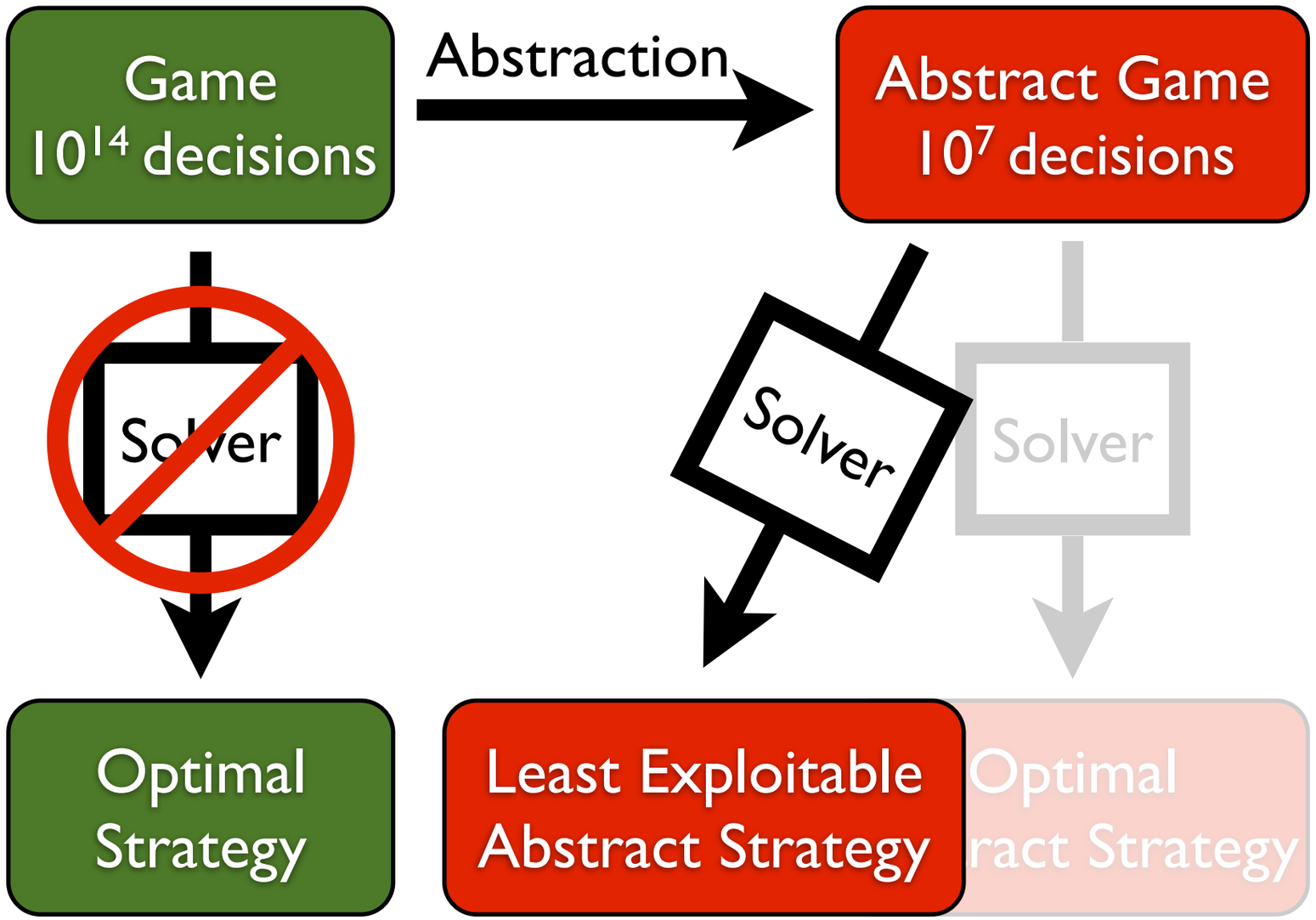
Abstract Equilibrium might not be optimal in the real game.



Abstract Equilibrium might not be optimal in the real game.



Abstraction-Solving-Translation



This Talk:

Efficiently finding an abstract strategy with the **lowest exploitability** in the real game.

Counterfactual Regret Minimization (CFR)
NIPS 2007

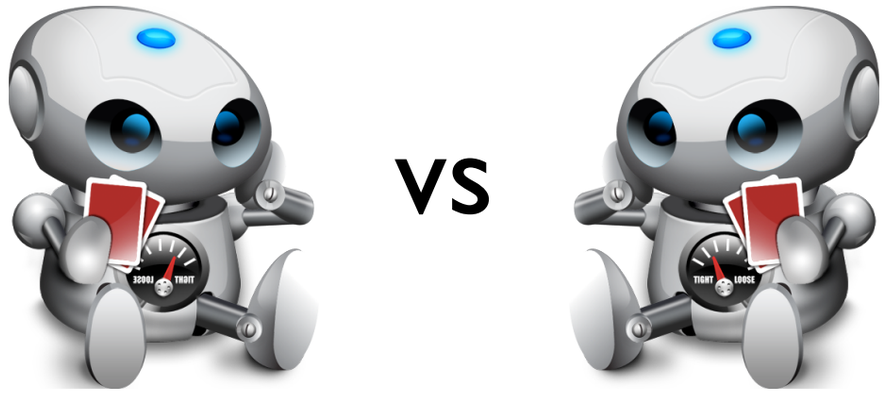
vs

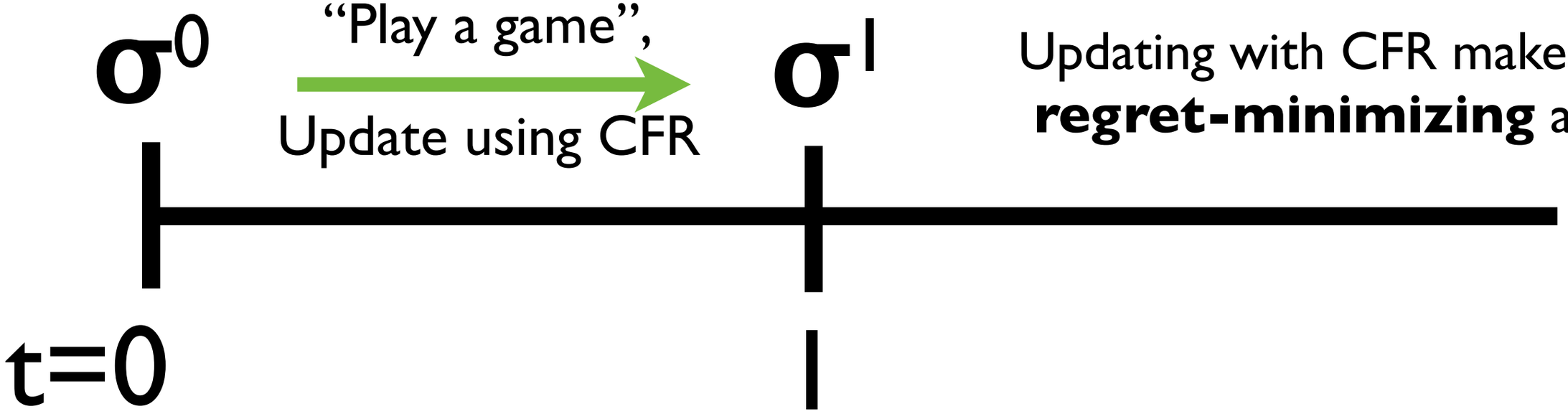
σ^0 = uniform random

|

t=0

Counterfactual Regret Minimization (CFR)
NIPS 2007

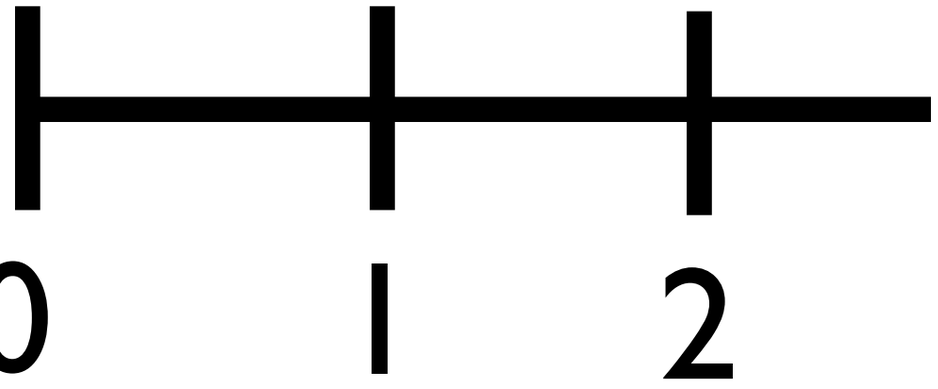




Counterfactual Regret Minimization (CFR)
NIPS 2007

vs

σ^0 σ^1 σ^2 The "Current" strategy



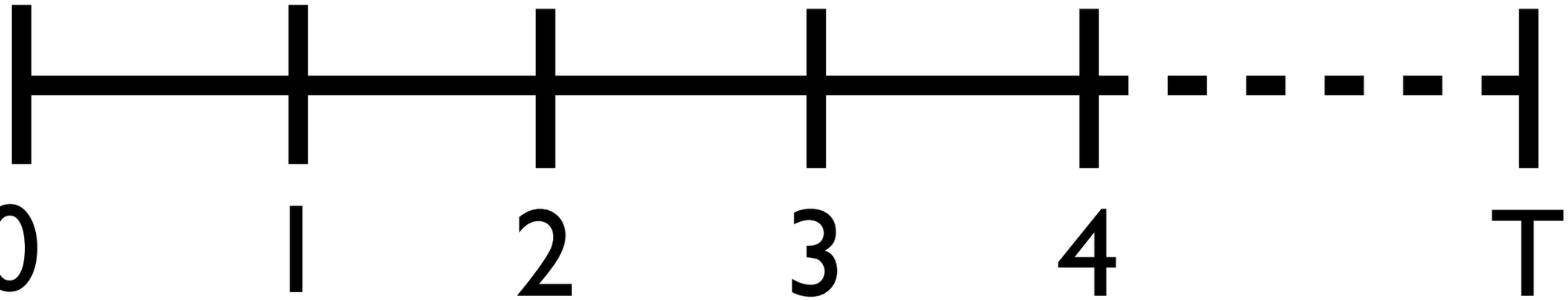
$$\hat{\sigma} = \frac{\sigma^0 + \sigma^1 + \ldots + \sigma^t}{t}$$

The "Average" strategy

Counterfactual Regret Minimization (CFR)
NIPS 2007

vs

σ^0 σ^1 σ^2 σ^3 σ^4 σ^T

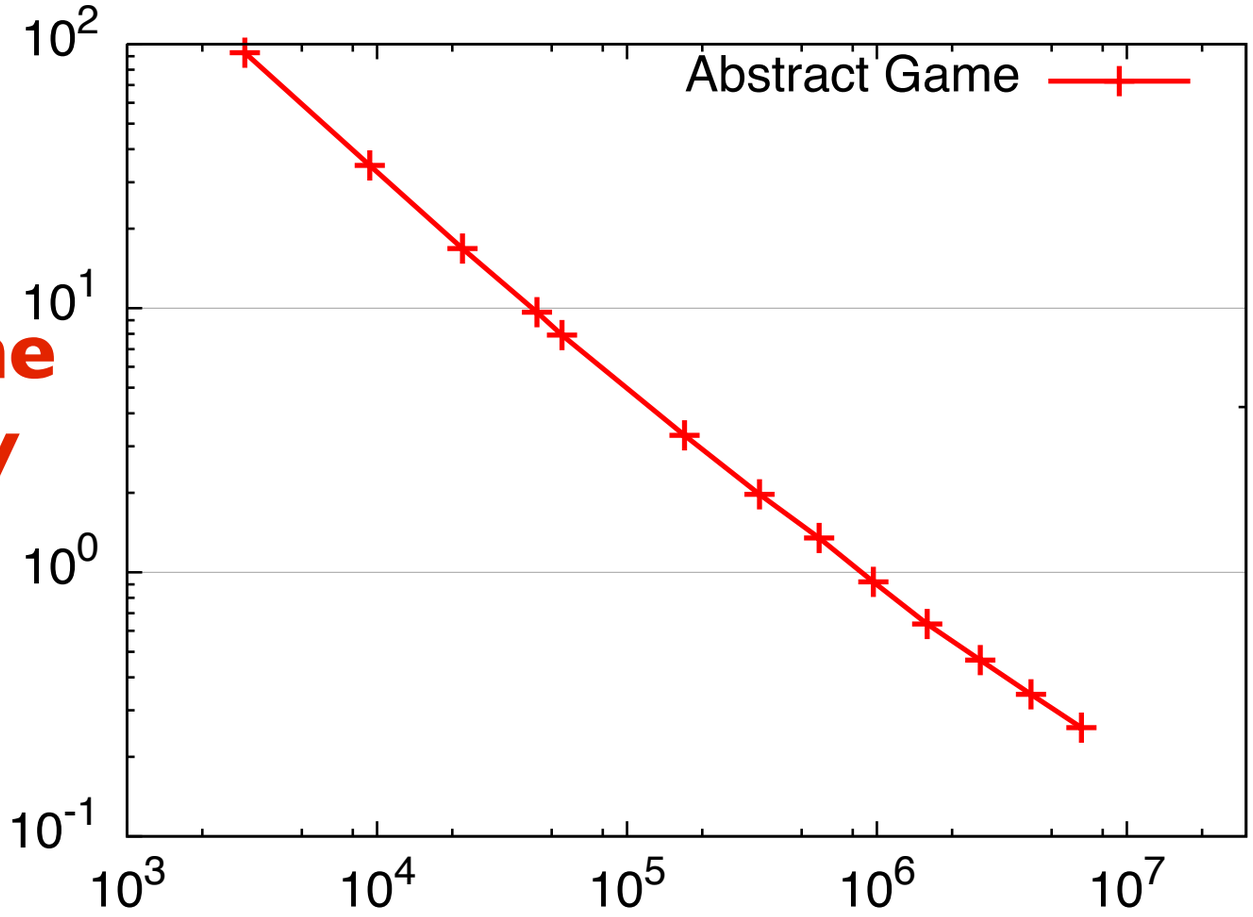


Key Theorem: If both players are **regret-minimizing**, then their **average strategy** converges towards an optimal strategy.

Counterfactual Regret Minimization (CFR)
NIPS 2007

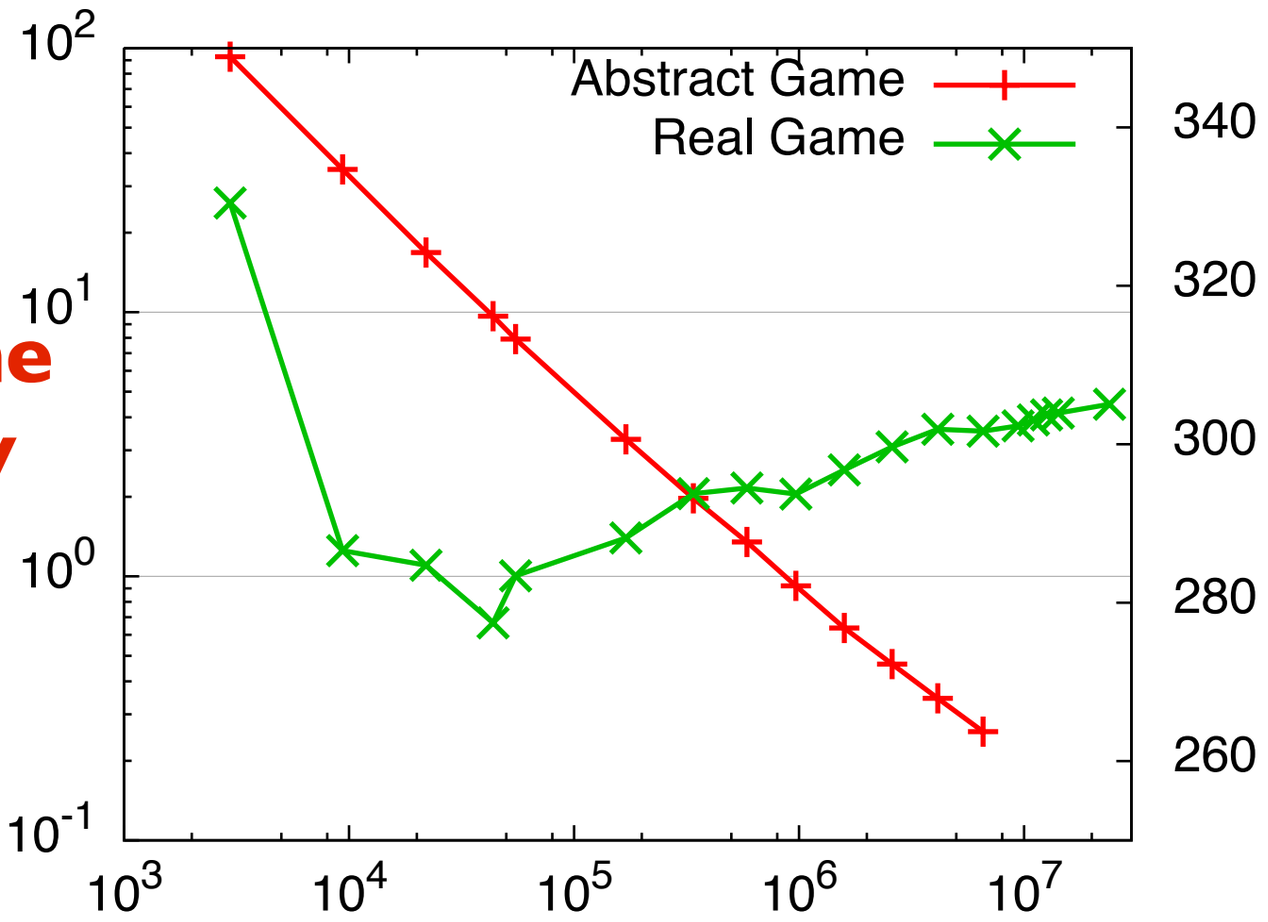
Abstract Game Exploitability

CFR in an abstract
10-Bucket Perfect Recall Game



Counterfactual Regret Minimization (CFR)
NIPS 2007

CFR in an abstract
10-Bucket Perfect Recall Game



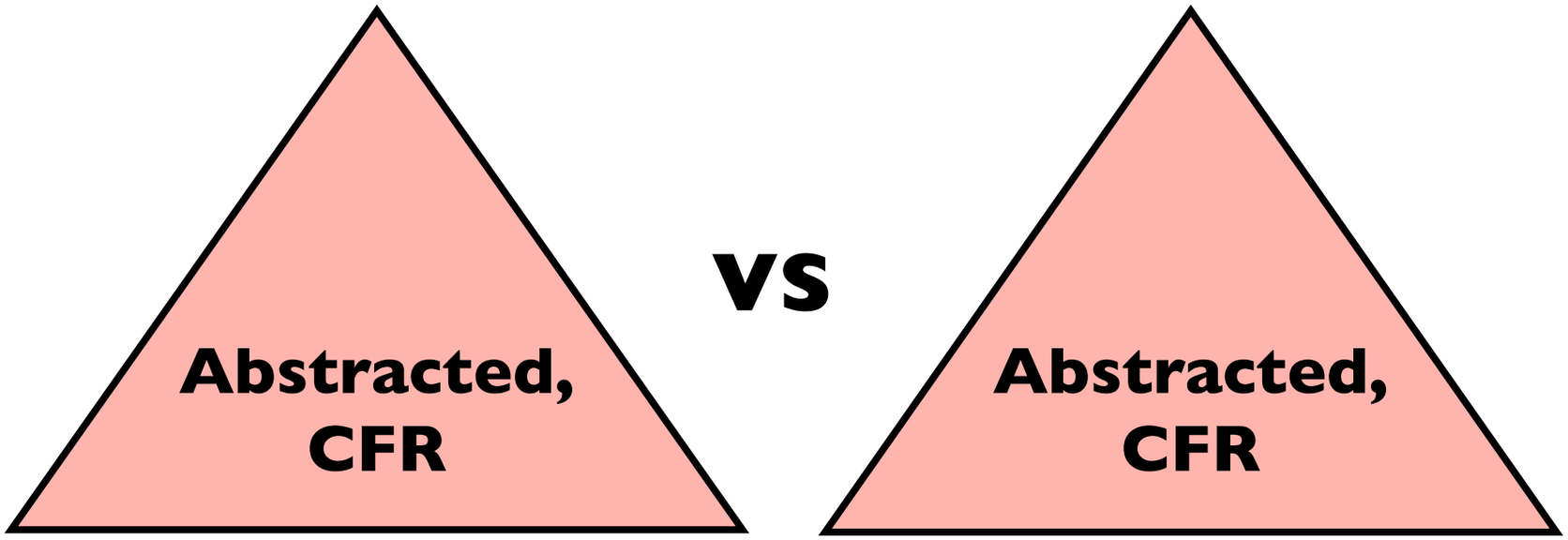
Abstract Game Exploitability

Real Game Exploitability

CFR Iterations

Moving from CFR to CFR-BR in six easy steps.

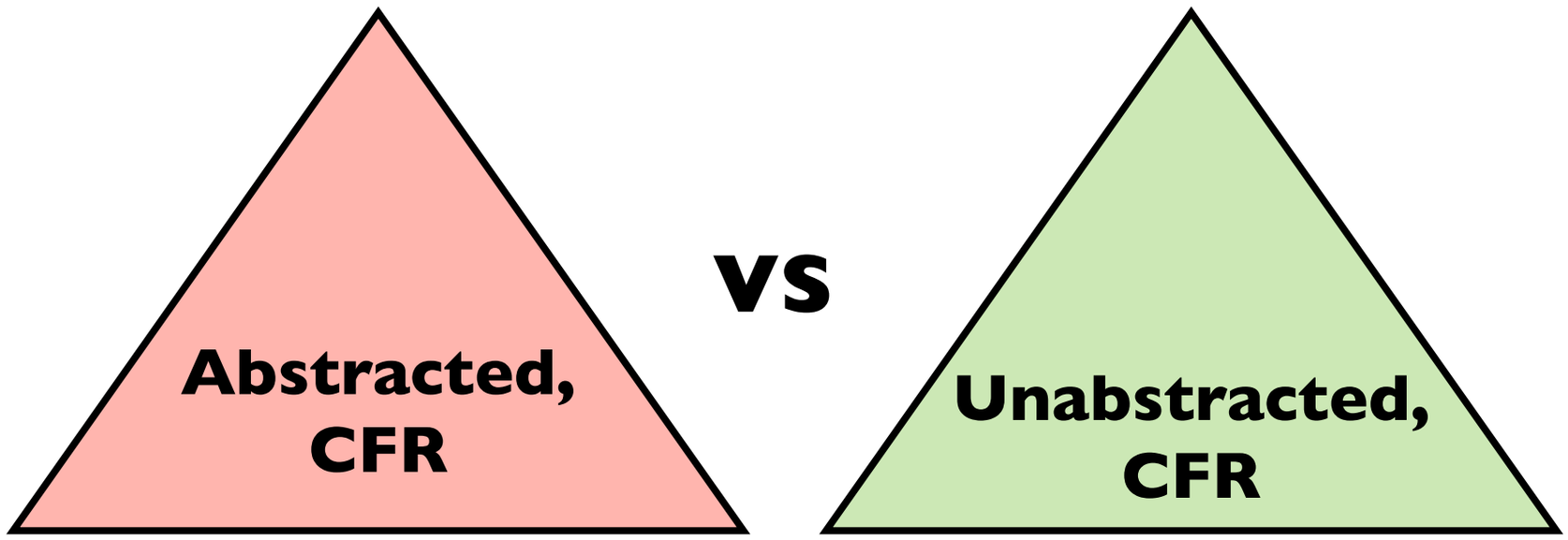
1 Both players abstracted.



X GB RAM X GB RAM

Both players are abstracted.
Computation is efficient,
Solution is suboptimal.
X is typically 1 to 100, depending on size of abstraction.

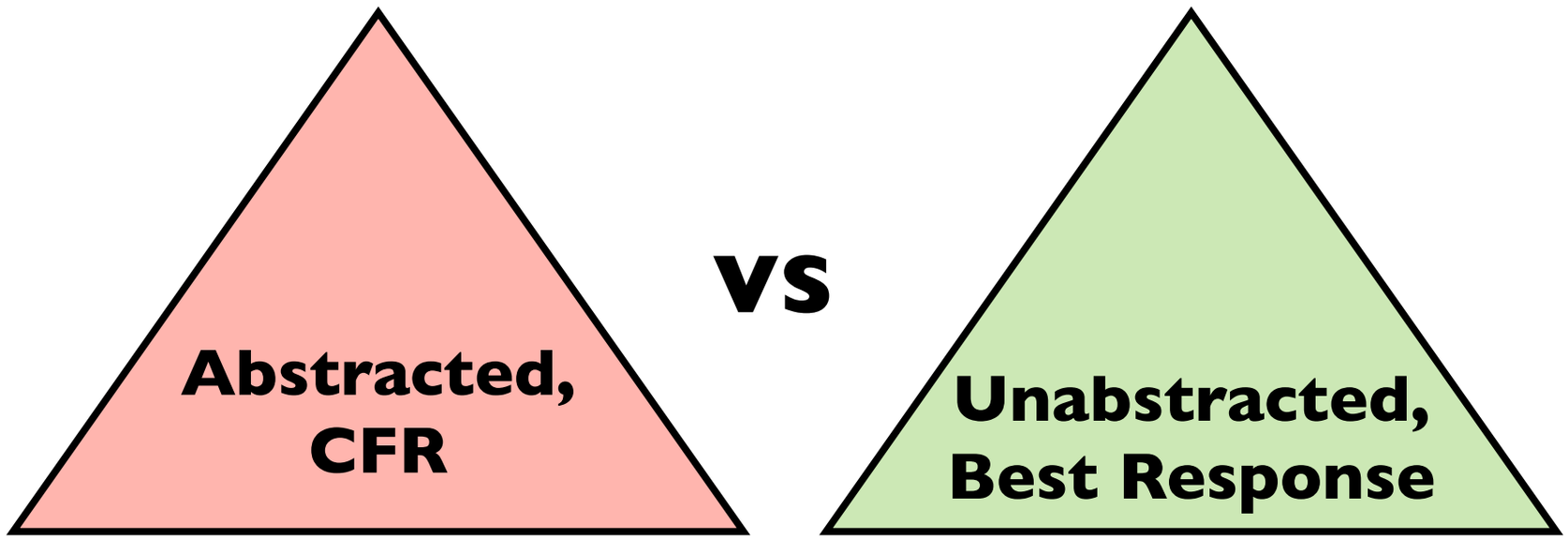
2 Opponent is unabstracted.



100 GB RAM 140 TB RAM

[Waugh et al., 2009]:
Opponent is unabstracted.
Abstracted player minimizes exploitability!
Requires far too much RAM and computation.

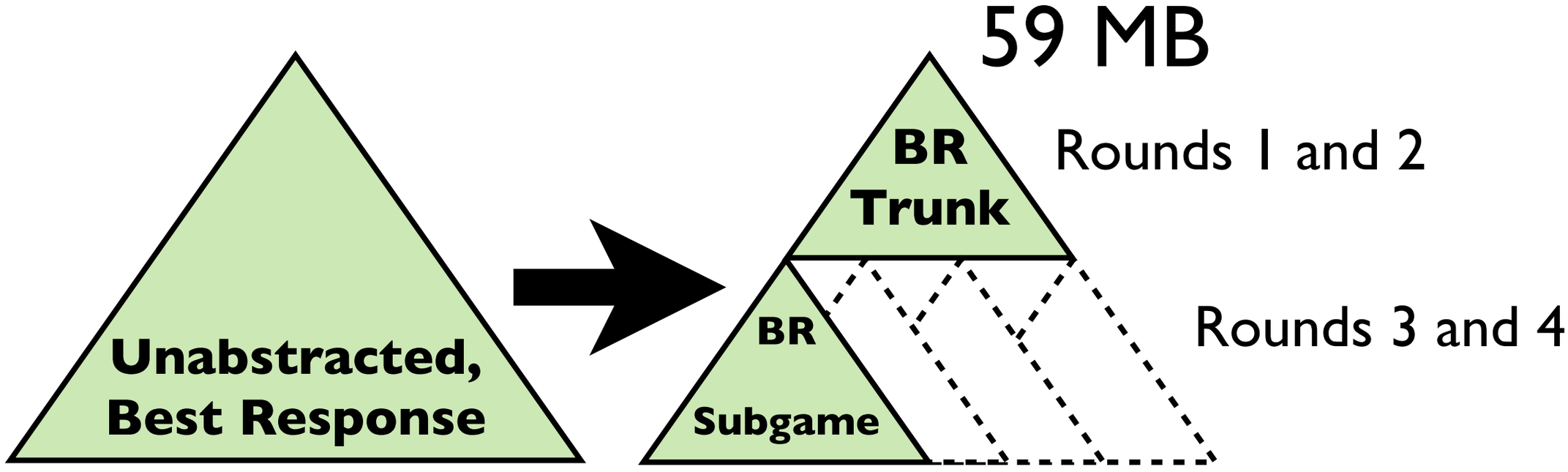
3 Play against a Best Response.



100 GB RAM 8.75 TB RAM

A Best Response is also **regret-minimizing**,
so **average** CFR strategy converges.
Current CFR strategy converges, too!
Takes 76 CPU-days to compute a BR.

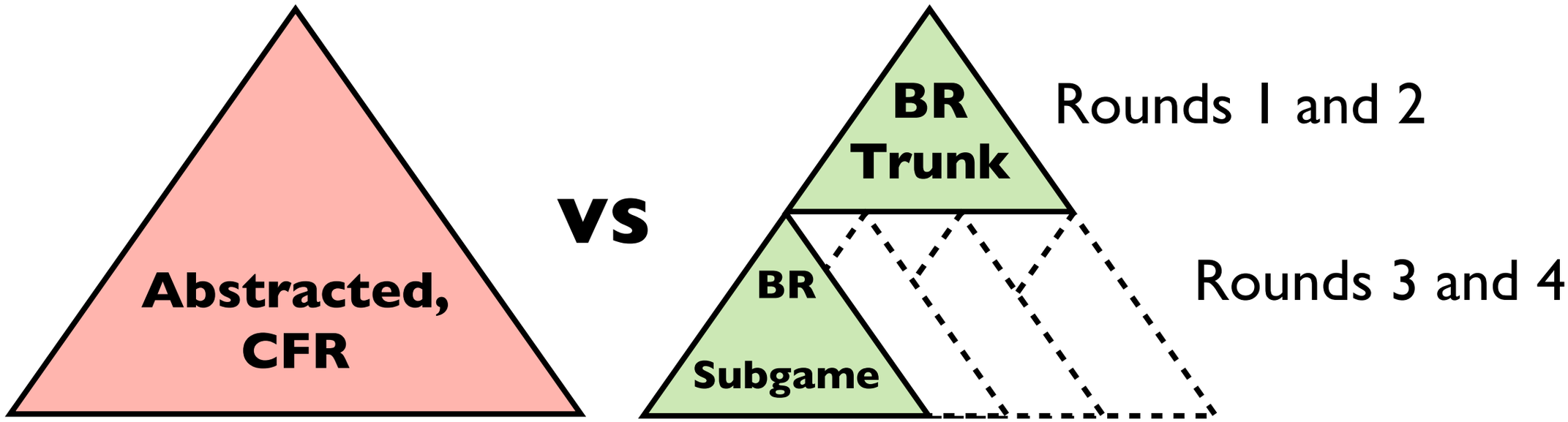
4 Split Best Response into pieces.



Split strategy into a **Trunk** and many **Subgames**.

Big advantage of Best Response:
Can compute subgames independently as needed!
Never need to store all of it at once!

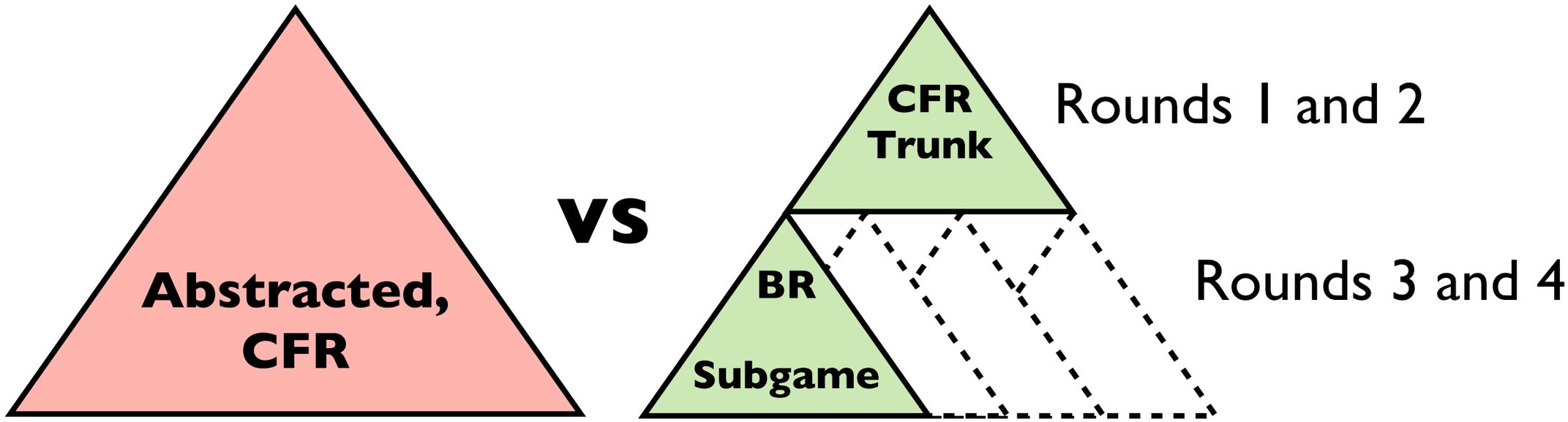
4 Split Best Response into pieces.



100 GB RAM 59+3 = 62 MB RAM

Compute subgames as needed, then discard. Memory problem solved! Takes 2x76 CPU-days, though: first pass to compute Trunk, second pass to play the game.

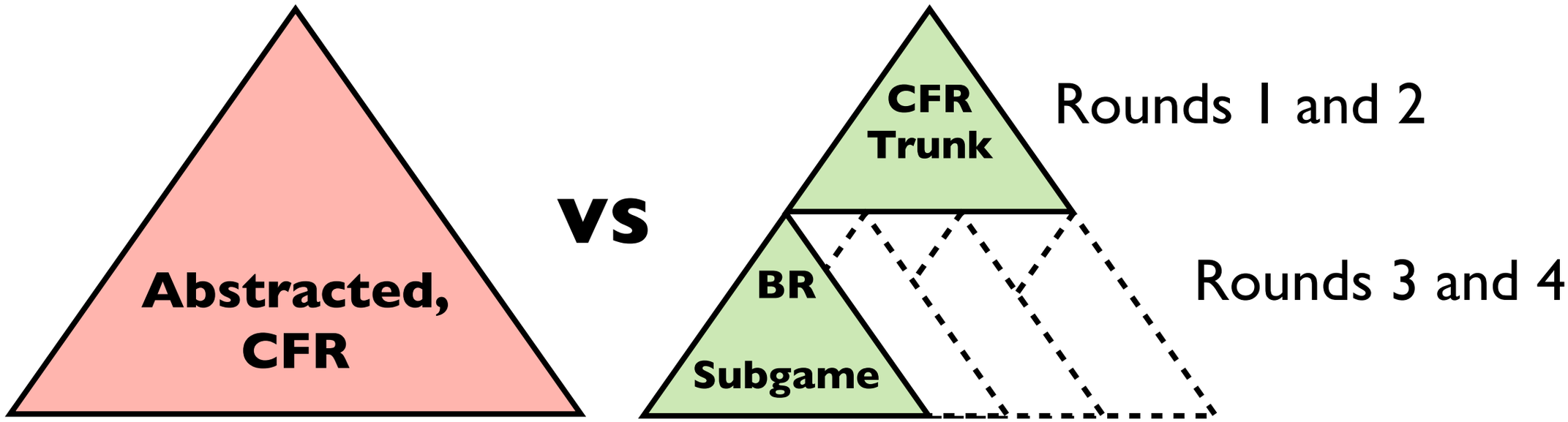
5 Play against a **CFR-BR** Hybrid.



100 GB RAM 936+3 = 940 MB RAM

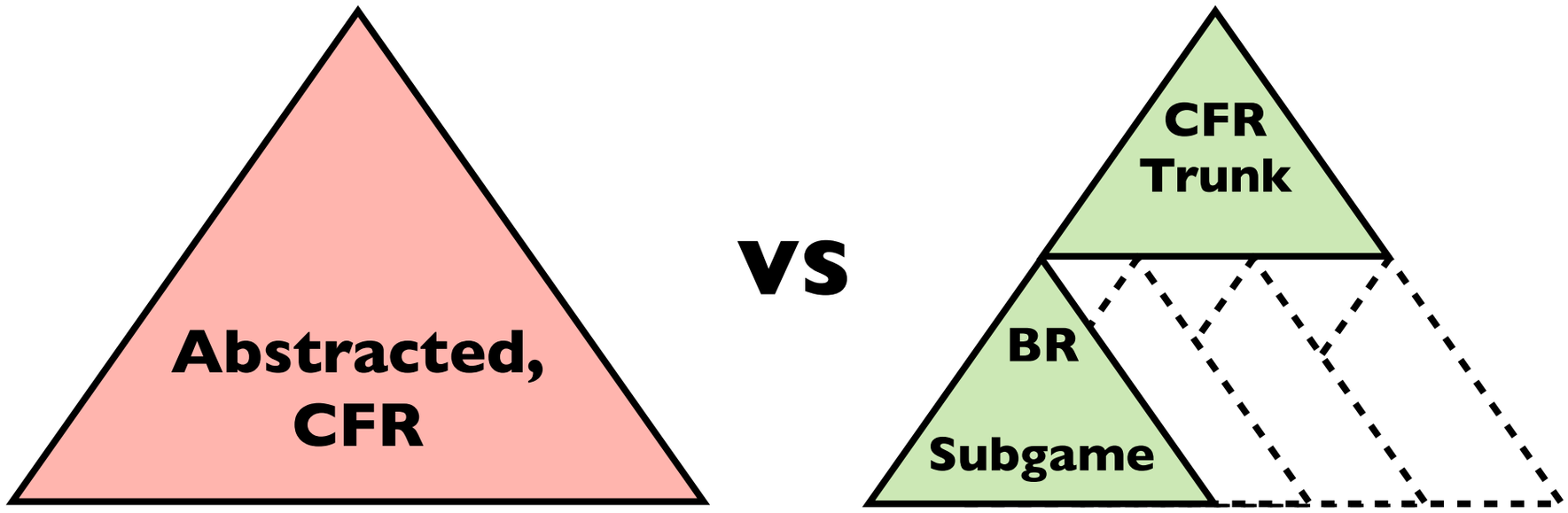
Use CFR to update Trunk strategy. This is also **regret-minimizing**, so CFR converges. Can query Trunk strategy any time, and compute Subgame strategy as needed.

6 Use Sampling to converge faster.



100 GB RAM 940 MB RAM

Sample one subgame, compute BR, update players.
Takes **50 CPU-seconds** per iteration and
940 MB RAM, and still converges!



100 GB RAM

940 MB RAM

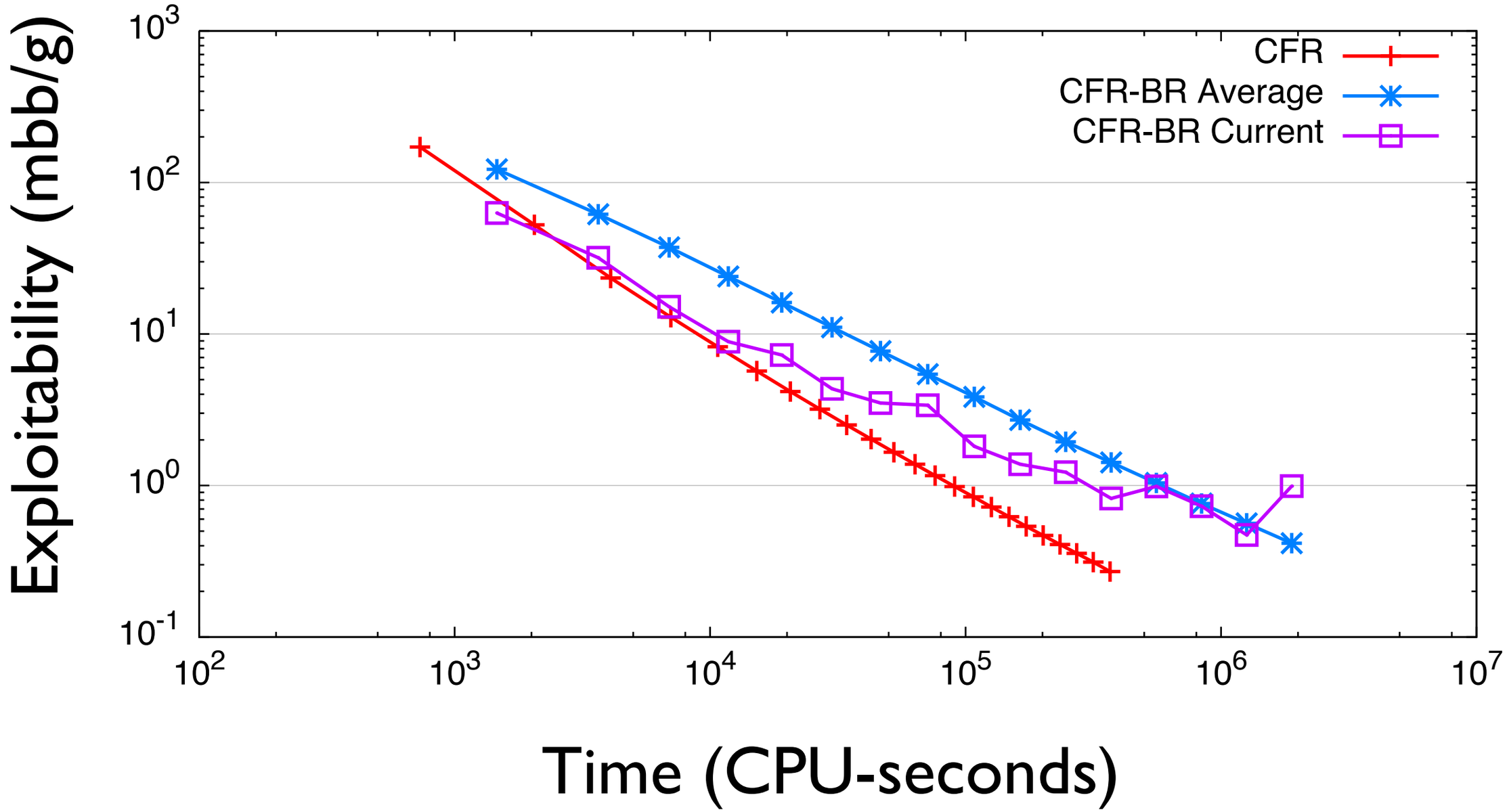
CFR-BR:

Finds the least exploitable abstract strategy,
while using *less* RAM than CFR did!

Average Strategy: Guaranteed to converge.
Current Strategy: Not guaranteed, but
converges faster in practice.

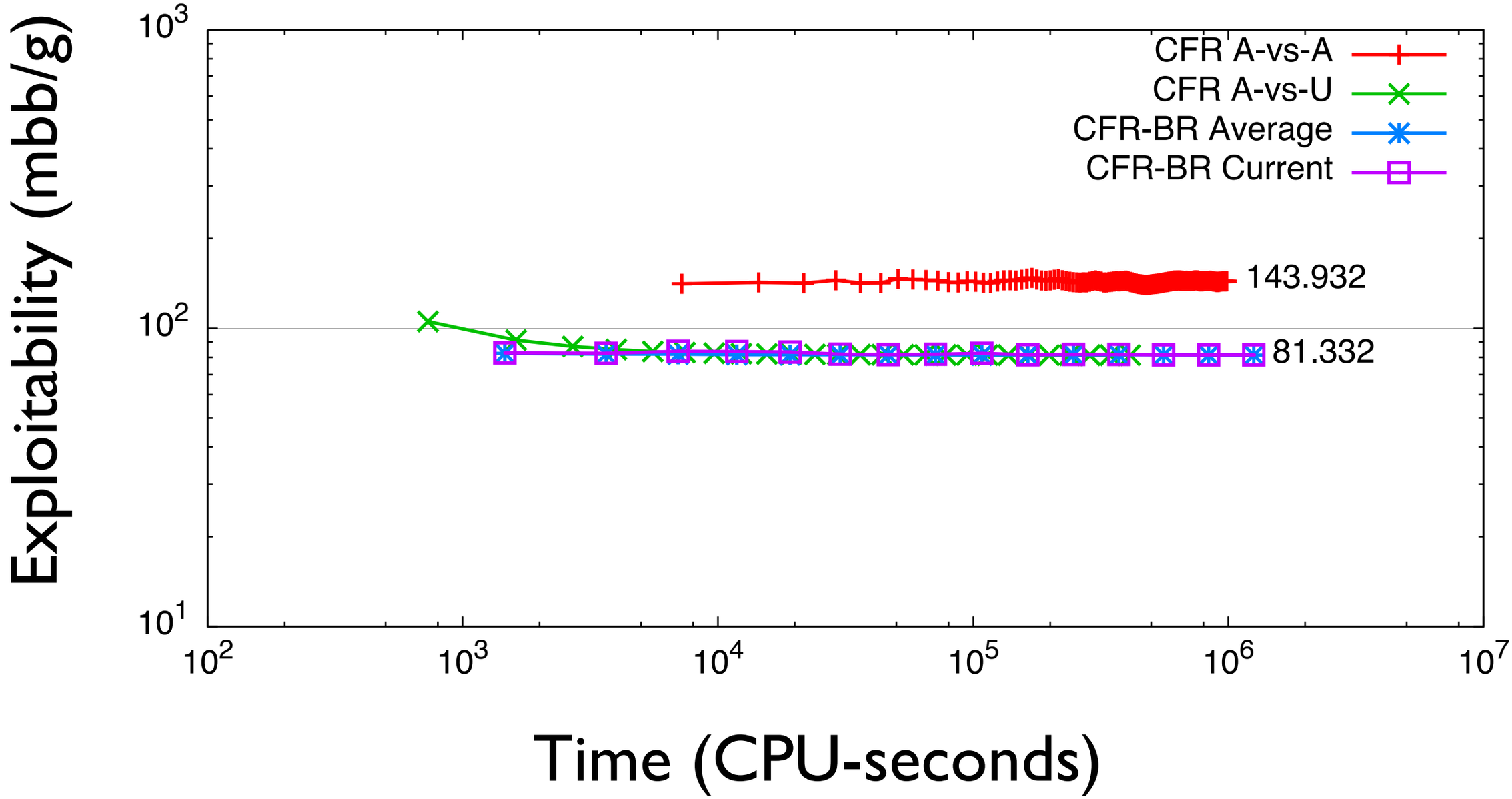
Testing in a small poker game

Unabstracted [2-4] Hold'em Poker: 94 million information sets



Testing in a small poker game

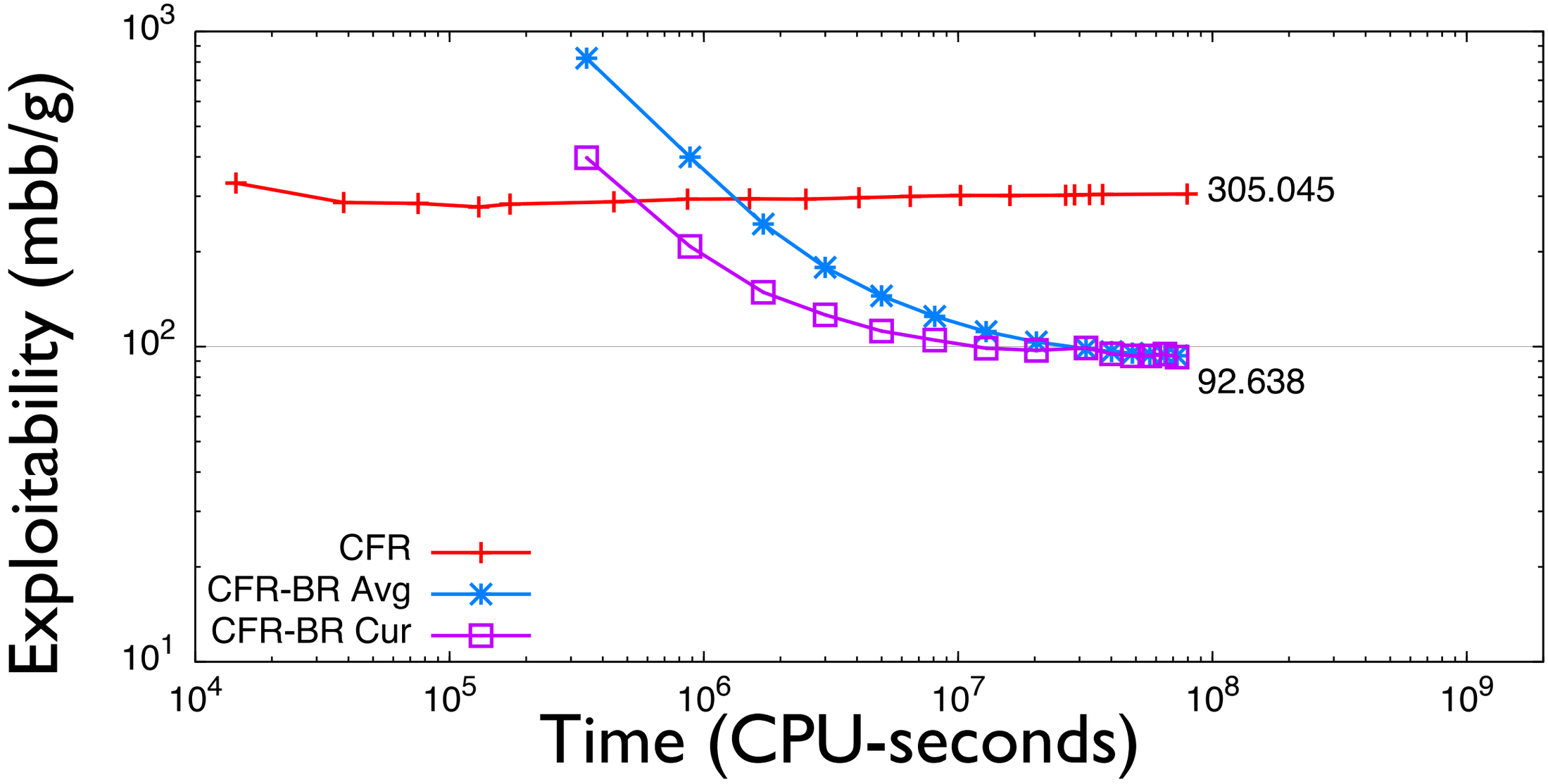
Abstracted [2-4] Hold'em: 1790 information sets



Texas Hold'em Poker: Small Abstractions

2007 Computer Poker Competition Abstraction
57 million information sets

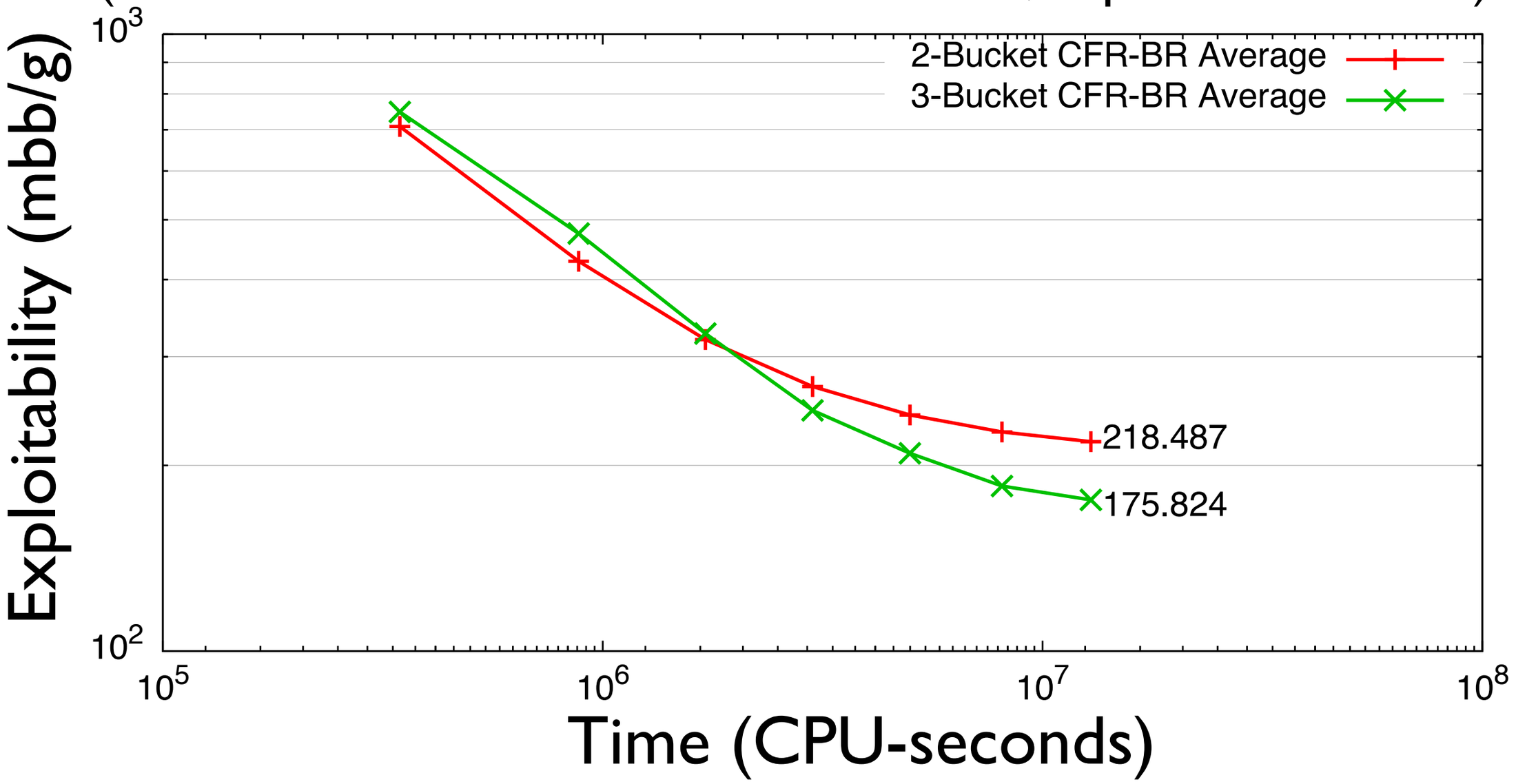
(Previous best strategy: 100x larger abstraction, exploitable for 104)



Texas Hold'em Poker: Tiny Abstractions

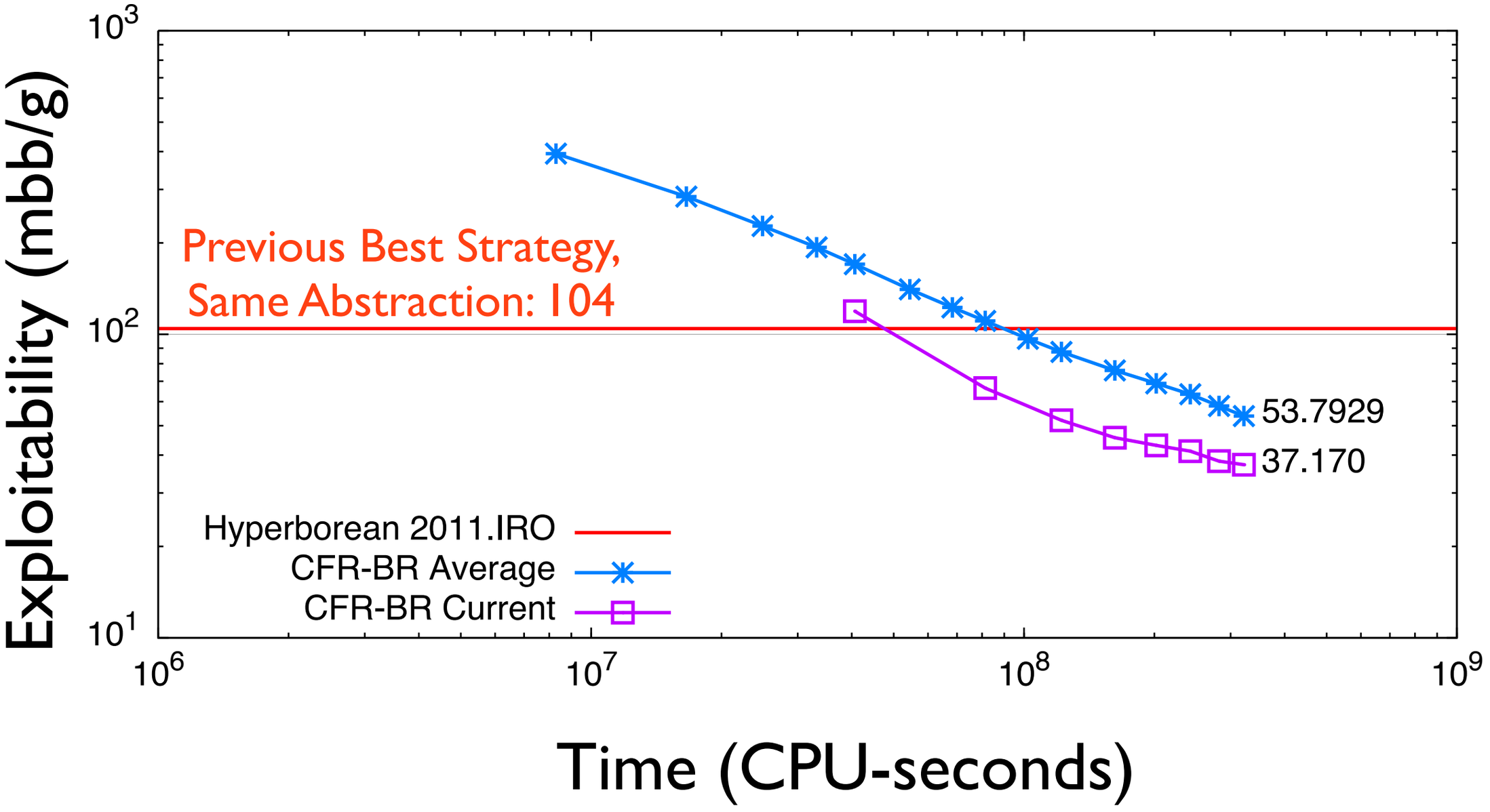
2-Bucket and 3-Bucket Abstractions: These fit on a **1.44 MB Floppy Disk**!

(2008 Man-vs-Machine Winner: **1.25 GB**, exploitable for 235)

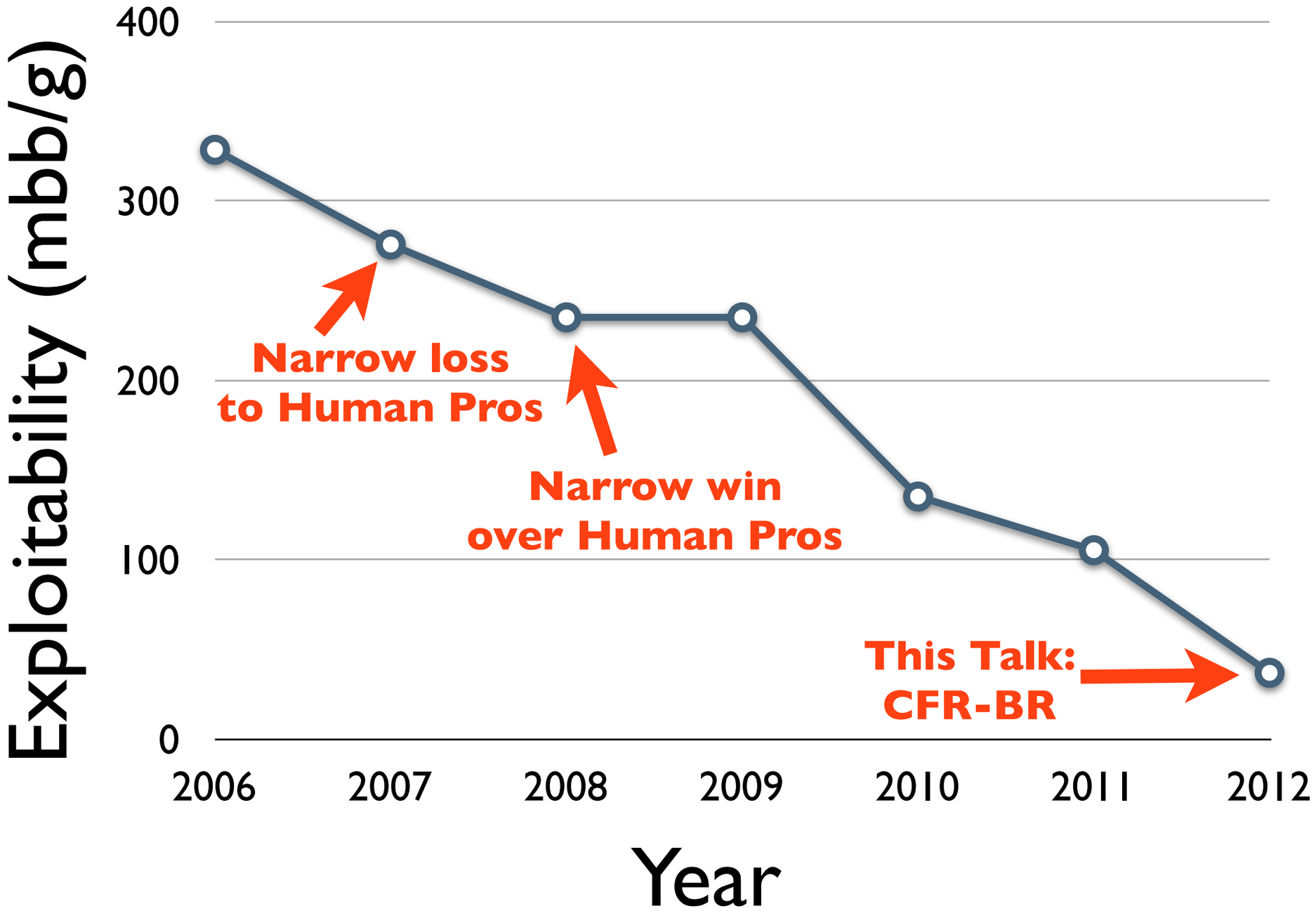


Texas Hold'em Poker: Small Abstractions

Least Exploitable Strategy Ever Made: 5.8 Billion information sets

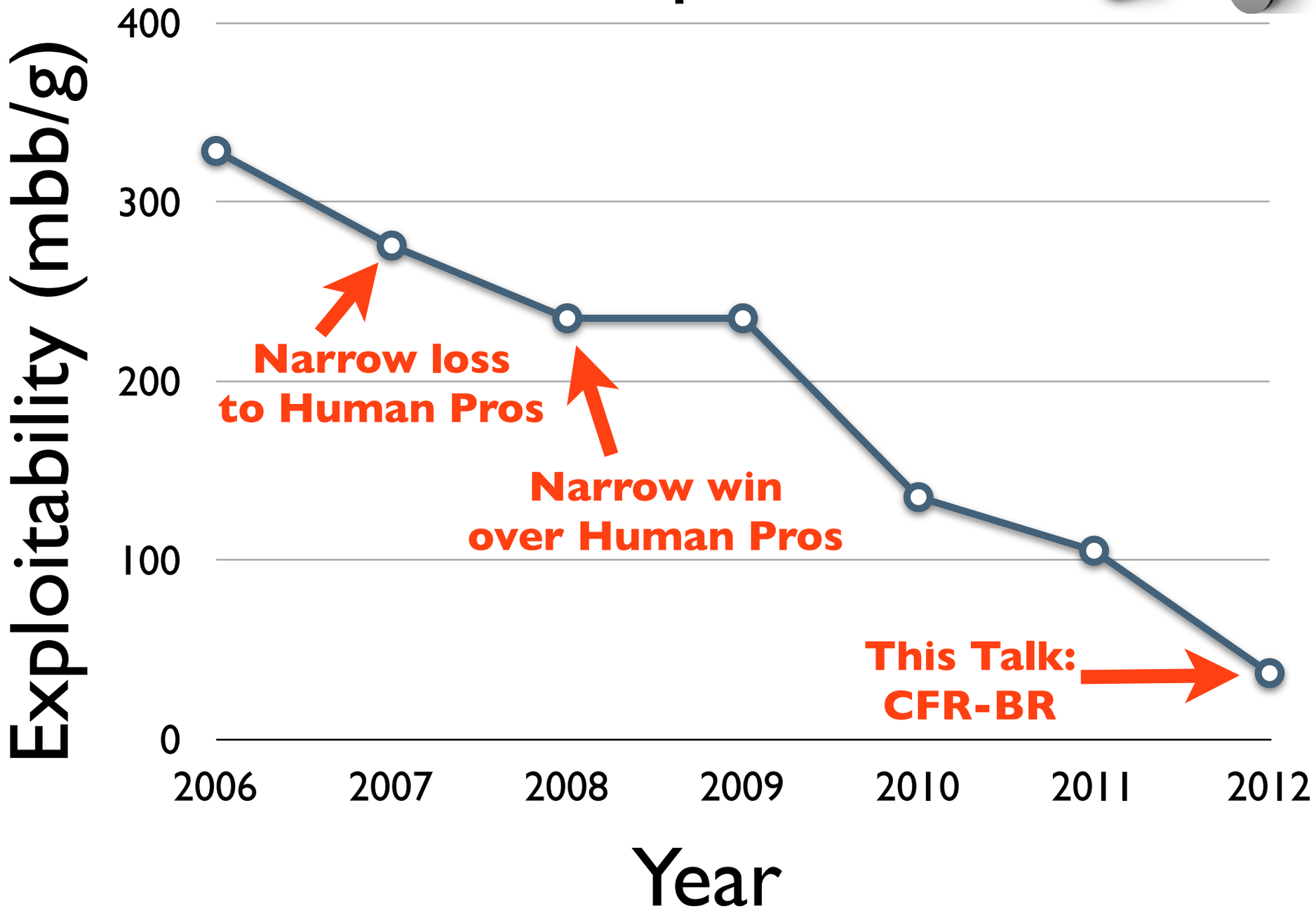


2-Player Limit Texas Hold'em Poker: Distance from Perfect Play



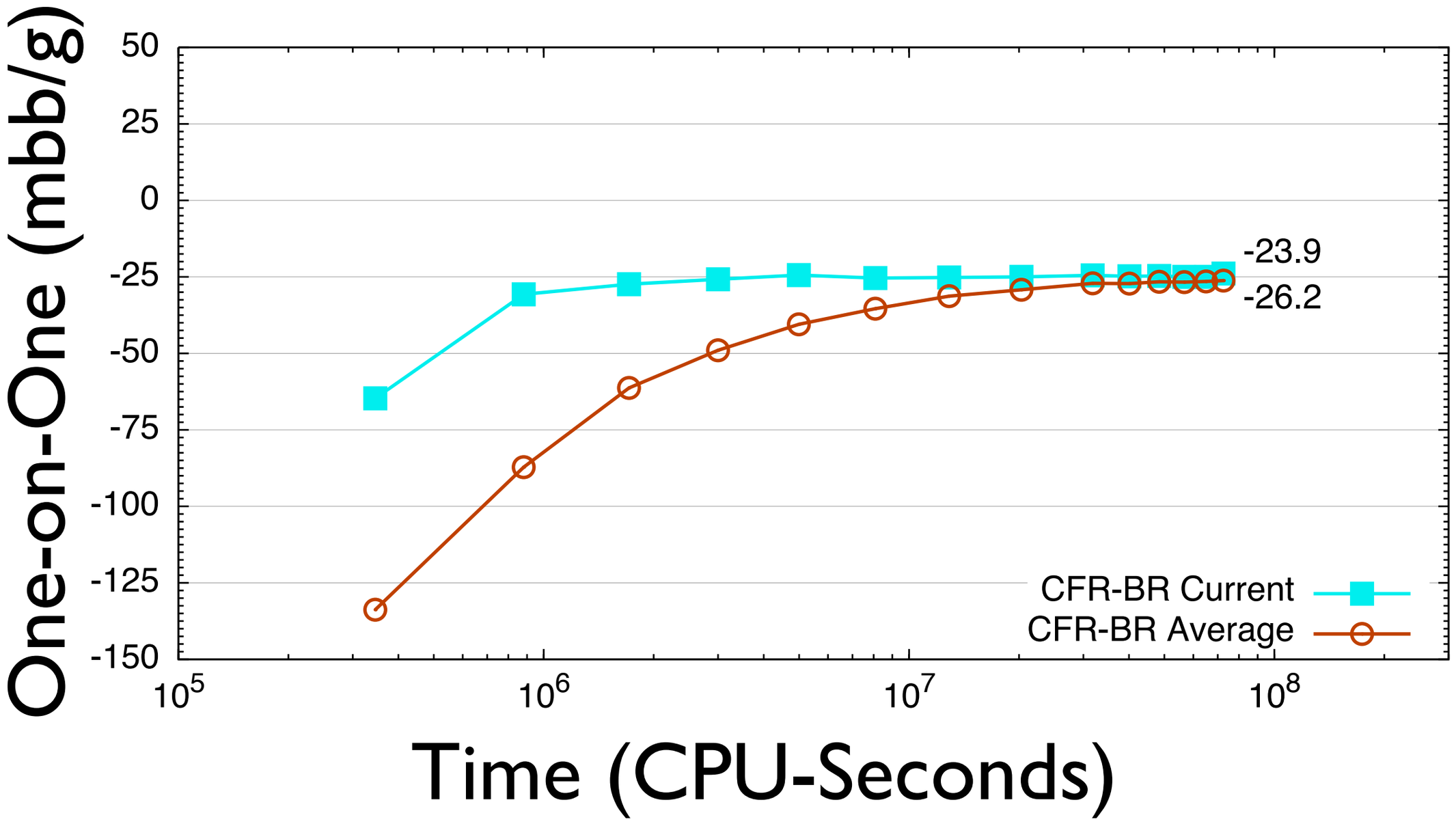
University of Alberta
Computer Poker Research Group

Thanks!
More results at the poster!

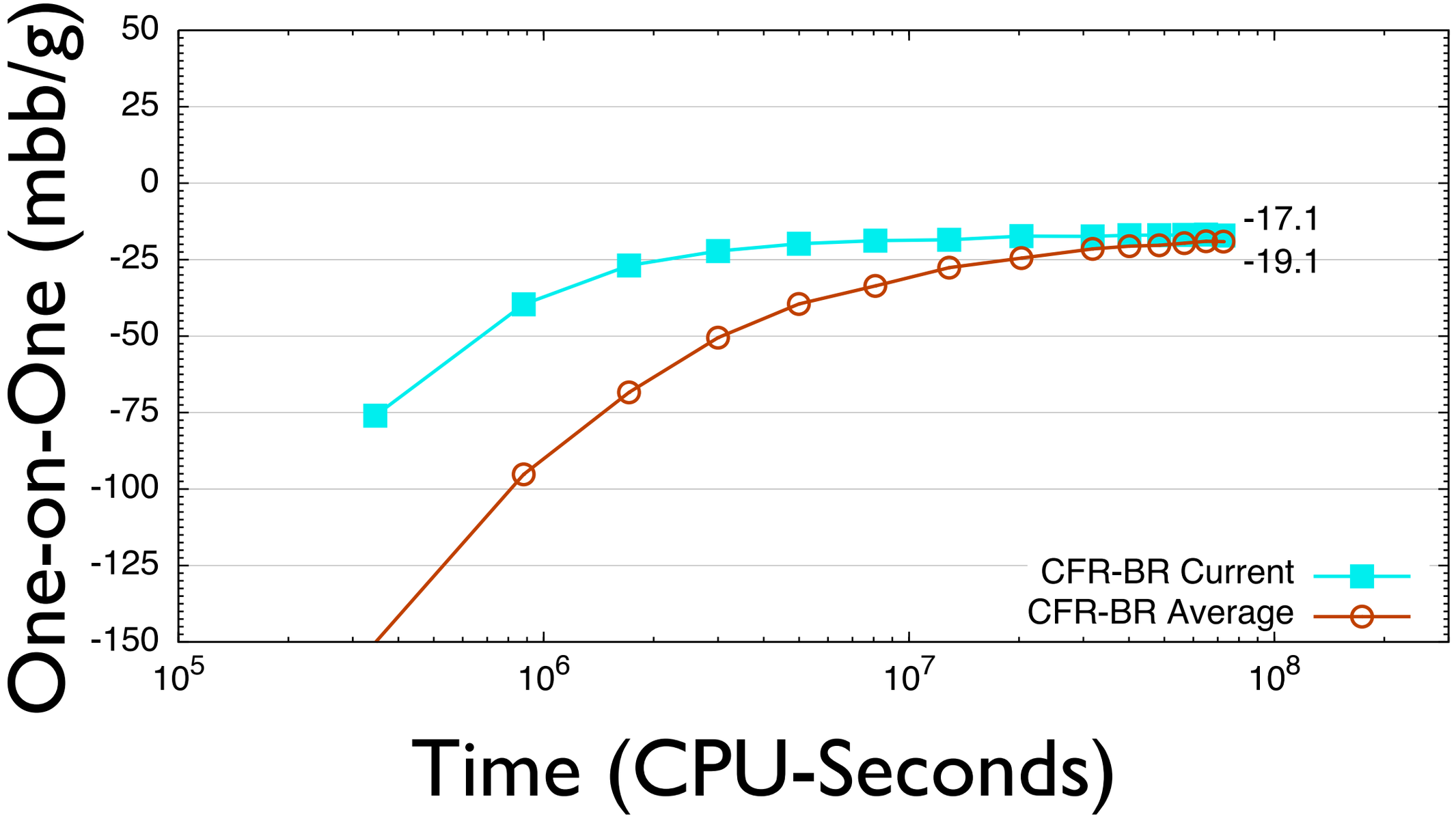


Bonus Slides

One-on-One: PR 10s



One-on-One: IR 9000



One-on-One: vs 2011