Robust Strategies and Counter-Strategies: From Superhuman to Optimal Play

Mike Johanson

January 14, 2016

Grad Seminar

University of Alberta
Computer Poker Research Group

michael.johanson@gmail.com

@mikebjohanson

Games as a testbed for Artificial Intelligence

Games as a testbed for Artificial Intelligence

Games as a testbed for Artificial Intelligence

Chinook (Checkers):
- Surpassed humans in 1994
- Solved (perfect play) in 2007

Games as a testbed for Artificial Intelligence

Chinook (Checkers):
- Surpassed humans in 1994
- Solved (perfect play) in 2007

Deep Blue (Chess):
- Surpassed humans in 1997

Games as a testbed for Artificial Intelligence

Chinook (Checkers):
- Surpassed humans in 1994
- Solved (perfect play) in 2007

Deep Blue (Chess):
- Surpassed humans in 1997

Watson (Jeopardy!):
- Surpassed humans in 2011

Games as a testbed for Artificial Intelligence

Chinook (Checkers):
- Surpassed humans in 1994
- Solved (perfect play) in 2007

Deep Blue (Chess):
- Surpassed humans in 1997

Watson (Jeopardy!):
- Surpassed humans in 2011

Current challenges (not yet superhuman):
go, Atari 2600 games, General Game Playing,
Starcraft, RoboCup, poker, curling (?!) and so on…

Games as a testbed for Artificial Intelligence

Games as a testbed for Artificial Intelligence

Babbage and Lovelace:
Wanted “Games of Purely Intellectual Skill” to demonstrate their Analytical Engine.
Chess, Tic-Tac-Toe. Horse racing?

Games as a testbed for Artificial Intelligence

Babbage and Lovelace:
Wanted "Games of Purely Intellectual Skill"
to demonstrate their Analytical Engine.
Chess, Tic-Tac-Toe. Horse racing?

Alan Turing:
Wrote a chess program before first computers, and ran it by hand. **Chess** as part of the Turing Test.

Games as a testbed for Artificial Intelligence

Babbage and Lovelace:
Wanted “Games of Purely Intellectual Skill” to demonstrate their Analytical Engine.
Chess, Tic-Tac-Toe. Horse racing?

Alan Turing:
Wrote a chess program before first computers, and ran it by hand. **Chess** as part of the Turing Test.

John von Neumann:
Founded **Game Theory** to study rational decision making.
Needed computational power to drive it, became pioneer in Computing Science.

Core idea in this line of research:

We aspire to create agents that can
achieve their goals in complex real-world domains.

Core idea in this line of research:

We aspire to create agents that can
achieve their goals in complex real-world domains.

Games provide a series of well-defined and
tractable domains that humans find challenging.

Core idea in this line of research:

We aspire to create agents that can
achieve their goals in complex real-world domains.

Games provide a series of well-defined and
tractable domains that humans find challenging.

New games introduce new challenges
that current approaches can’t handle.
This is a gradient we can follow.

Core idea in this line of research:

We aspire to create agents that can
achieve their goals in complex real-world domains.

Games provide a series of well-defined and
tractable domains that humans find challenging.

New games introduce new challenges
that current approaches can’t handle.
This is a gradient we can follow.

Can play against humans, to compare
Artificial Intelligence to Human Intelligence.

John von Neumann pioneered Game Theory.
When asked about real life and **chess**, he said…

John von Neumann pioneered Game Theory.
When asked about real life and **chess**, he said…

Real life is not like that.

Real life consists of bluffing,
of little tactics of deception,
of asking yourself what is the other man
going to think I mean to do.

And that is what games are
about in my theory.

Chess is a..

2-player,

deterministic,

perfect information game,

with win / lose / tie outcomes.

Chess is a..

2-player,

deterministic,

perfect information game,

with win / lose / tie outcomes.

Poker:

2-10 Players (at one table)
Thousands (tournaments)

Chess is a..

2-player,

deterministic,

perfect information game,

with win / lose / tie outcomes.

Poker:

2-10 Players (at one table)
Thousands (tournaments)

Stochastic:
Cards randomly dealt
to players and the
table.

Chess is a..

2-player,

deterministic,

perfect information game,

with win / lose / tie outcomes.

Poker:

2-10 Players (at one table)
Thousands (tournaments)

Stochastic:
Cards randomly dealt to players and the table.

Imperfect Information:
Opponent's cards are hidden.

Chess is a..

2-player,

deterministic,

perfect information game,

with win / lose / tie outcomes.

Poker:

2-10 Players (at one table)
Thousands (tournaments)

Stochastic:
Cards randomly dealt to players and the table.

Imperfect Information:
Opponent’s cards are hidden.

Maximize winnings by exploiting opponent errors.

My Research and This Grad Seminar

Topic: Computing strong strategies
in Imperfect Information Games

2008:
PhD Start

2015:
PhD End

My Research and This Grad Seminar

Two key milestones
in 2-Player limit hold'em poker:

2008:
PhD Start

2015:
PhD End

My Research and This Grad Seminar

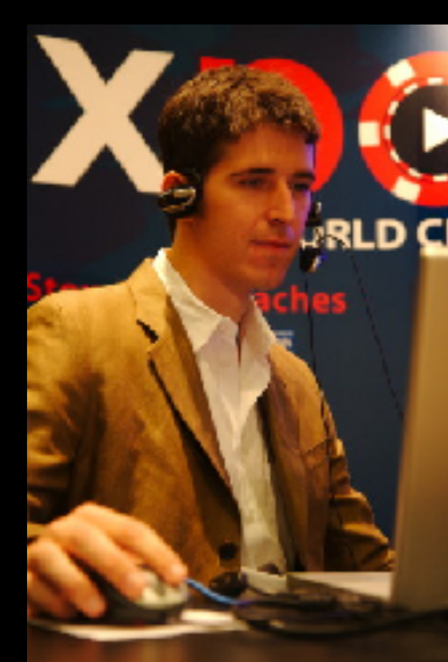
Two key milestones
in 2-Player limit hold'em poker:

2008:
PhD Start

2015:
PhD End

First computer
victory over
human poker
pros.

>



My Research and This Grad Seminar

Two key milestones
in 2-Player limit hold'em poker:

2008:
PhD Start

First computer victory over human poker pros.

2015:
PhD End

Game solved. Computer is now optimal.

>= Everyone, forever.

My Research and This Grad Seminar

Two key milestones
in 2-Player limit hold'em poker:

2008:
PhD Start

First computer victory over human poker pros.

Solving Attempt #1

Solving Attempt #2

Solving Attempt #3…

2015:
PhD End

Game solved. Computer is now optimal.

My Research and This Grad Seminar

Two key milestones
in 2-Player limit hold'em poker:

2008:
PhD Start

First computer victory over human poker pros.

2015:
PhD End

Game solved. Computer is now optimal.

Note: I'll be **very** high-level in this talk.
This is a summary of 7 papers in my thesis,
and 7 more not in my thesis.
Ask questions!

Superhuman Play:

The **Abstraction-Solving-Translation** Procedure.

This is how we beat the pros in 2008.

First used in poker by Shi and Littman in 2002.

Still the dominant approach in large games.

Terminology:

Strategy: A policy for playing a game.
At every decision, a probability
distribution over actions.

Terminology:

Strategy: A policy for playing a game.
At every decision, a probability distribution over actions.

Best Response: A strategy that maximizes utility against a specific target strategy.

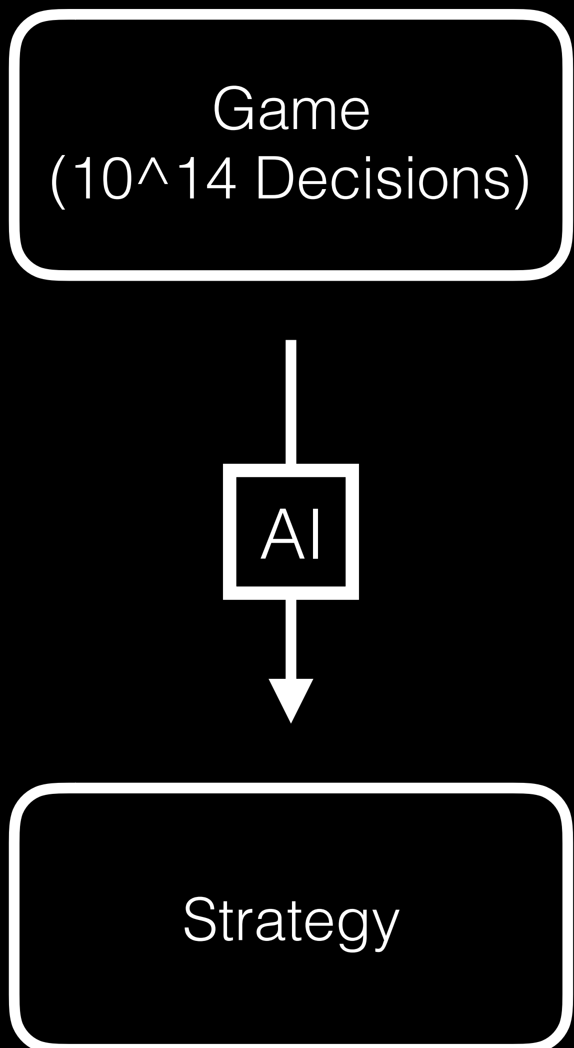
Terminology:

Strategy: A policy for playing a game.
At every decision, a probability distribution over actions.

Best Response: A strategy that maximizes utility against a specific target strategy.

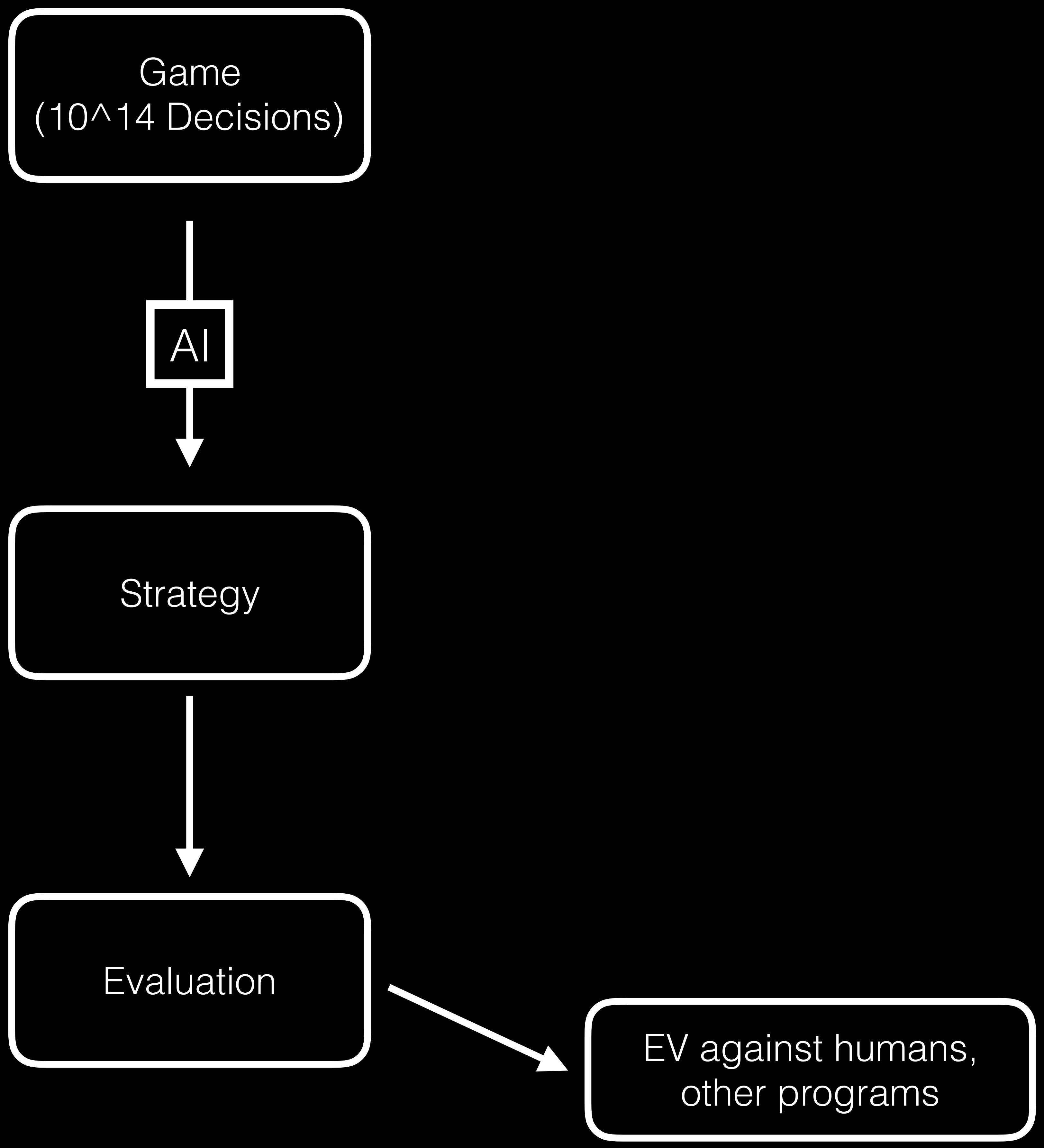
Nash Equilibrium: A strategy for every player that are all mutually best responses to the others.

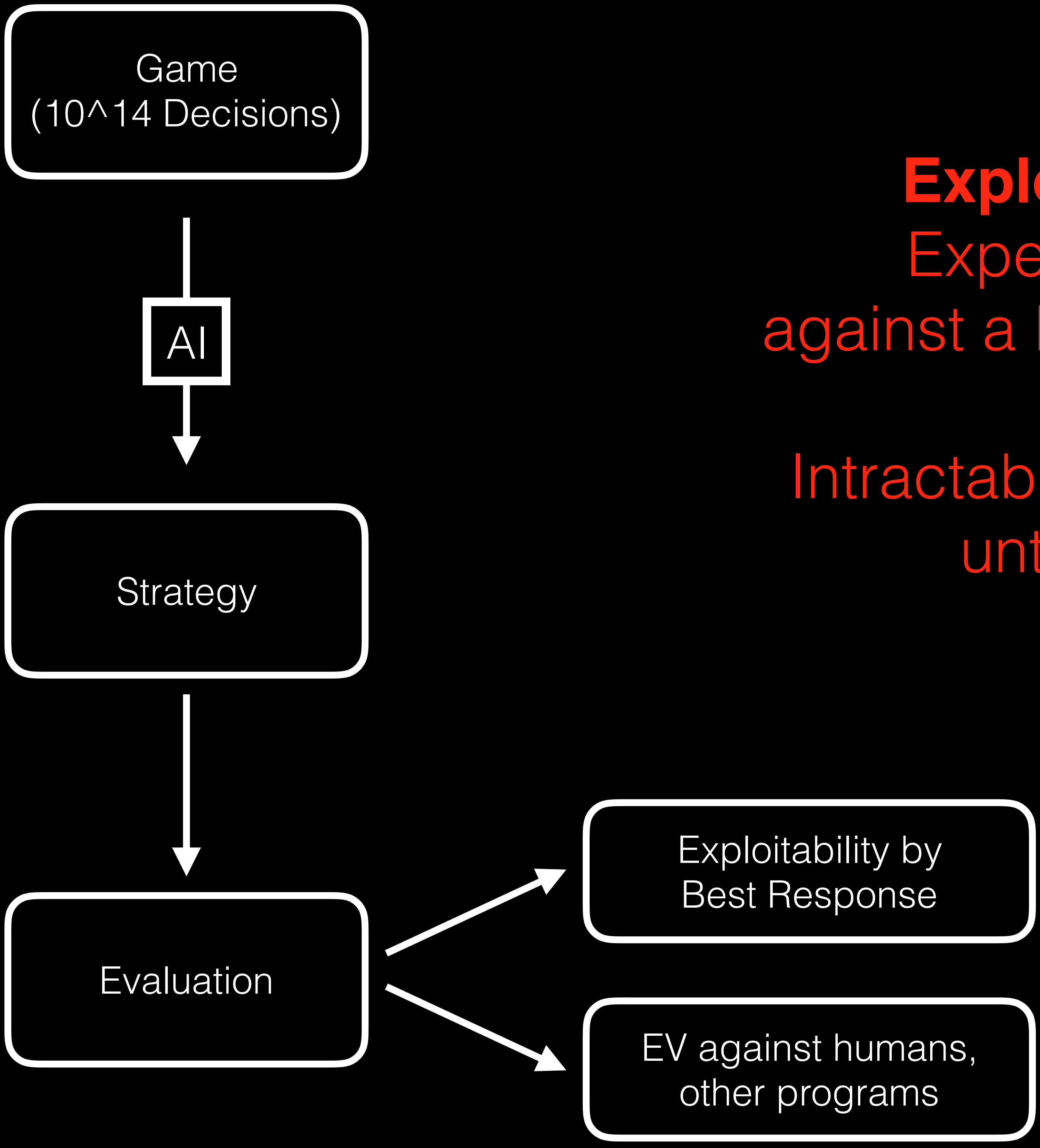
In a 2-player zero-sum game, it's guaranteed to do no worse than tie.



Solve the game by computing a Nash Equilibrium.

(Opponent Modelling comes later)


Game
(10^14 Decisions)
AI
Strategy
Evaluation
EV against humans,
other programs



Exploitability:
Expected loss
against a best response.

Intractable to compute
until 2011.

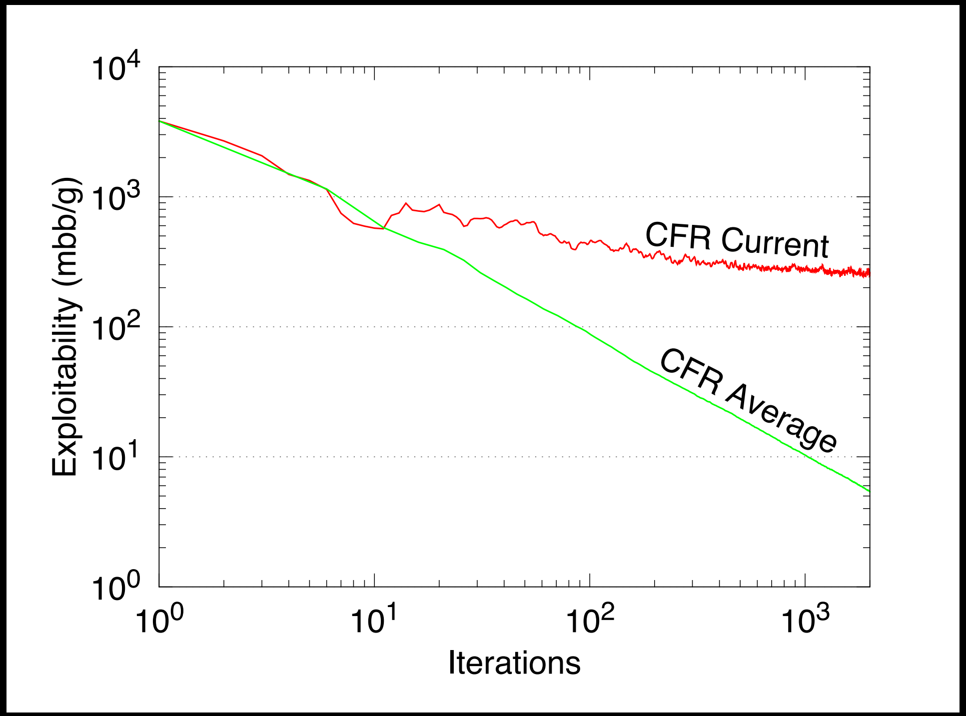
The AI Step: Counterfactual Regret Minimization (CFR)

1. Start with Uniform Random strategy.

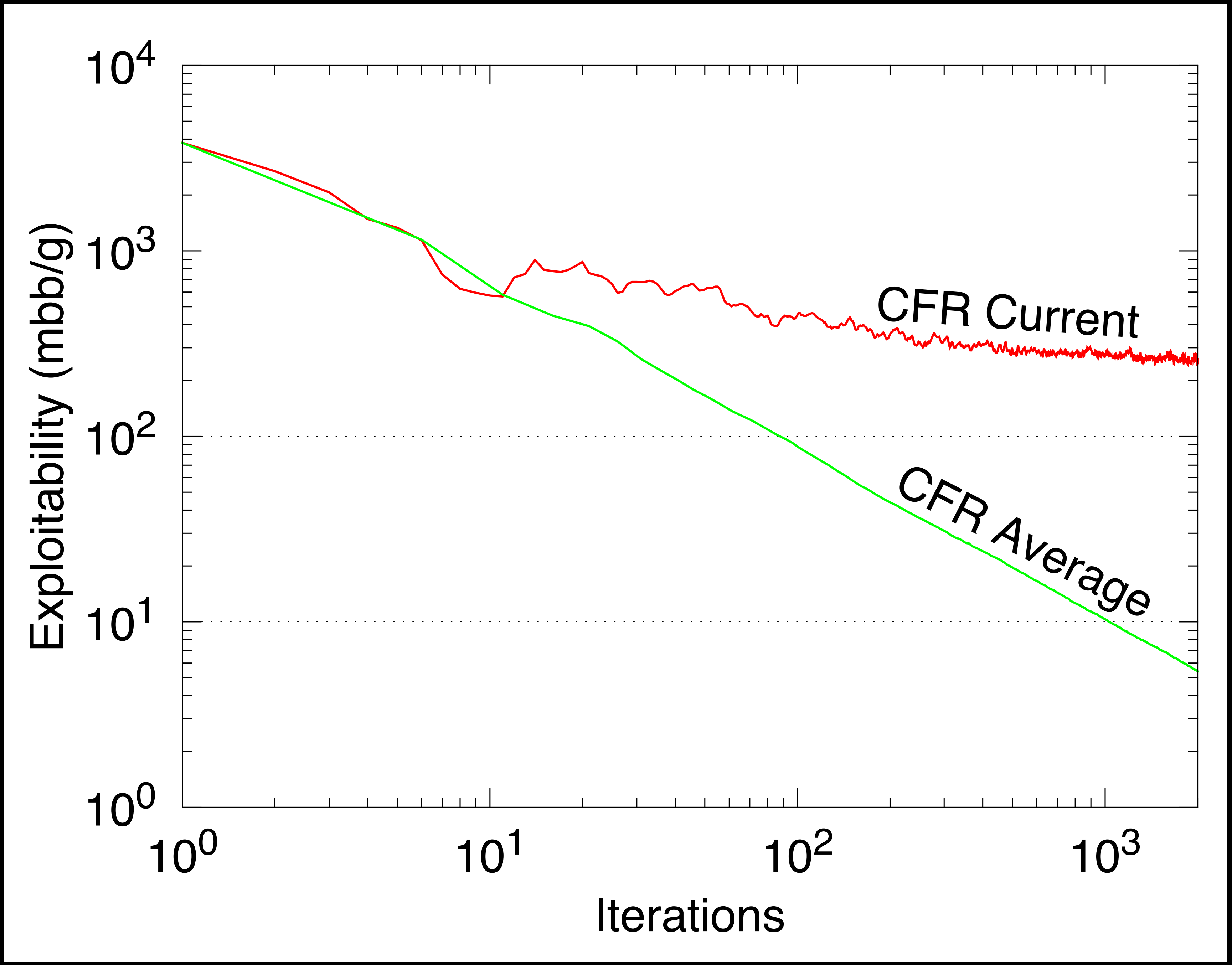
2. Repeatedly plays against itself.  vs 

2a. Update: At each decision, use the historically best actions more often. (minimizing regret)

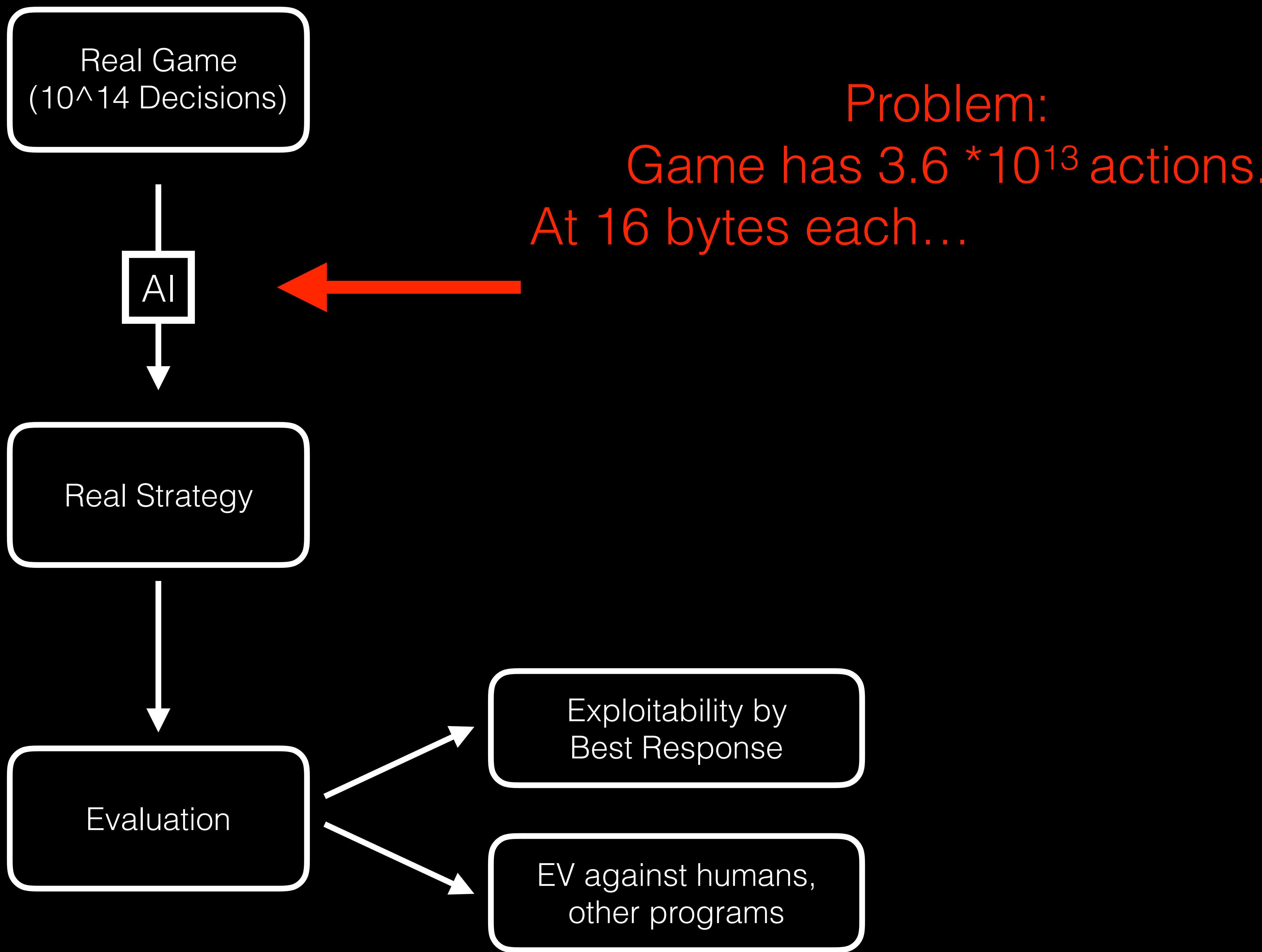
3. Average strategy converges towards a Nash equilibrium.

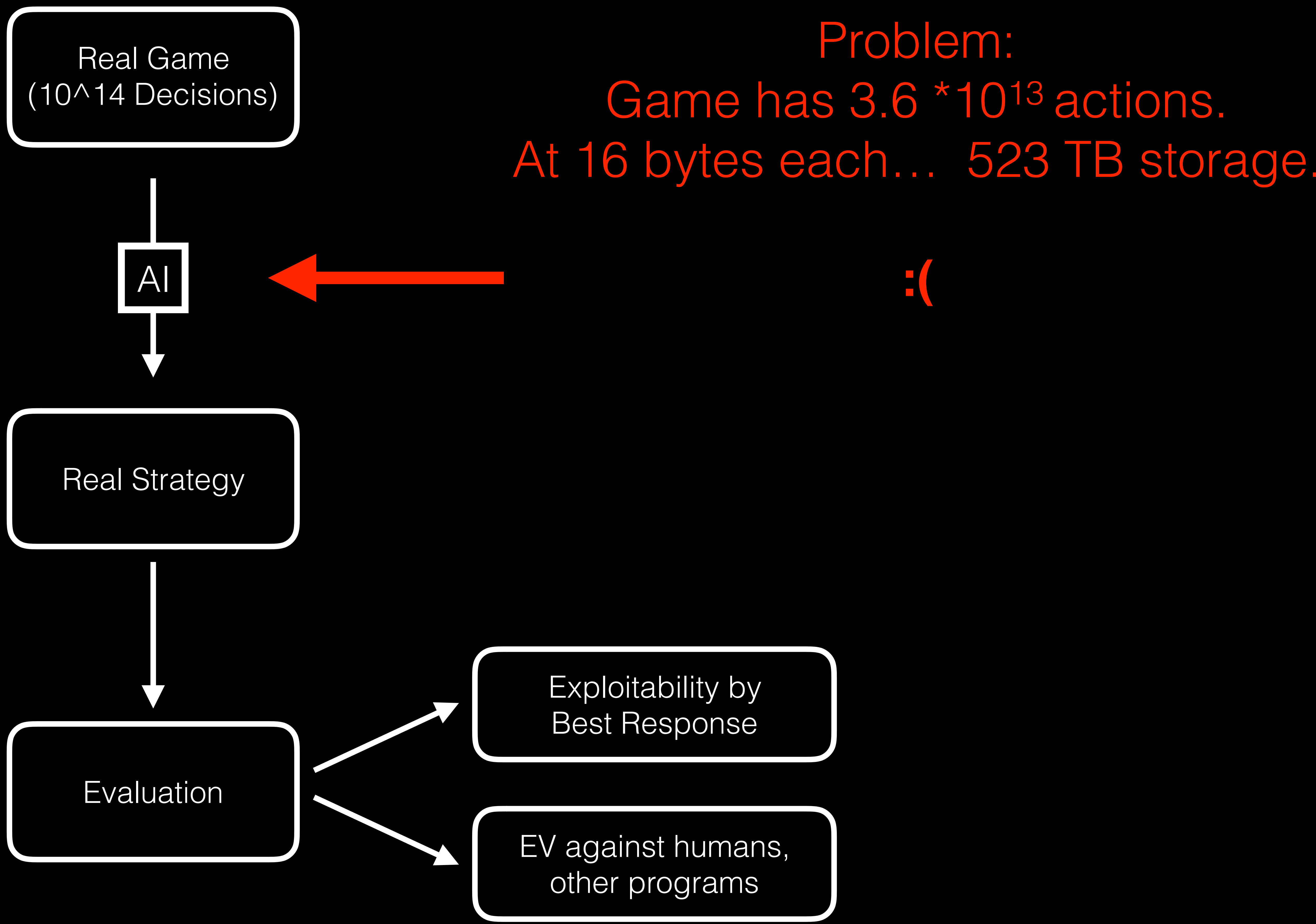


The AI Step: Counterfactual Regret Minimization (CFR)

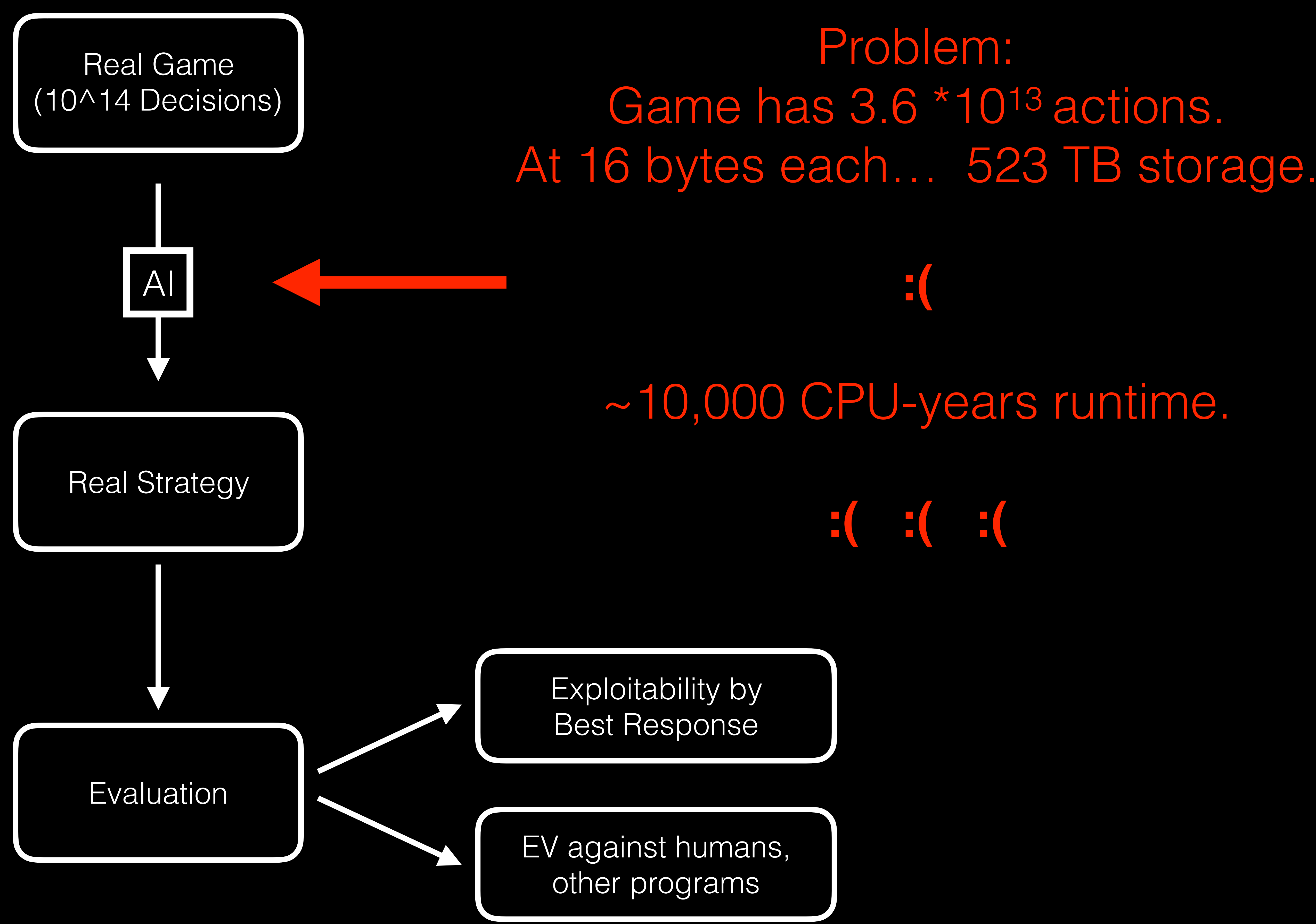


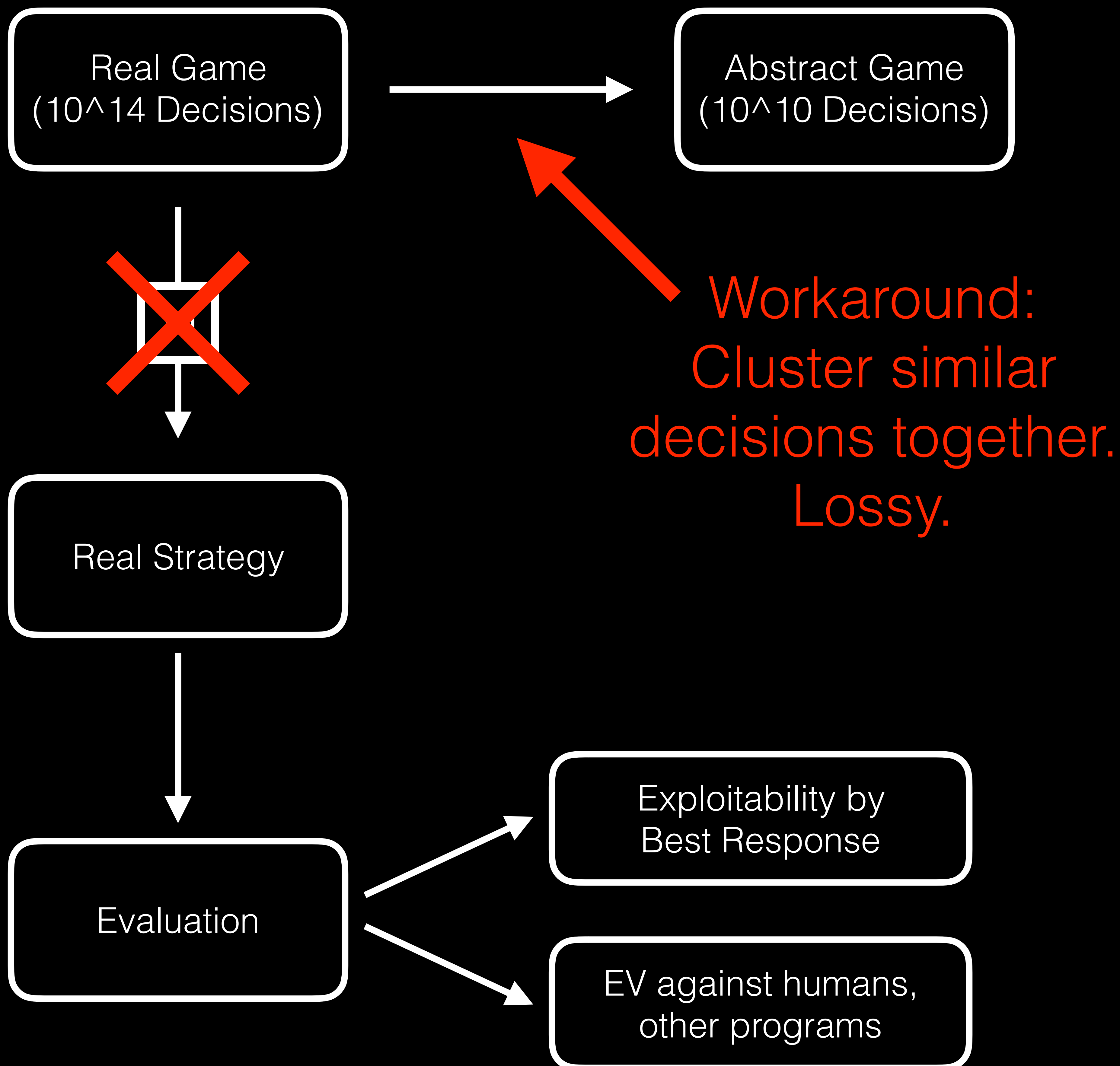
Memory Cost: **2 doubles** per **Action-at-Decision-Point**
(16 bytes)

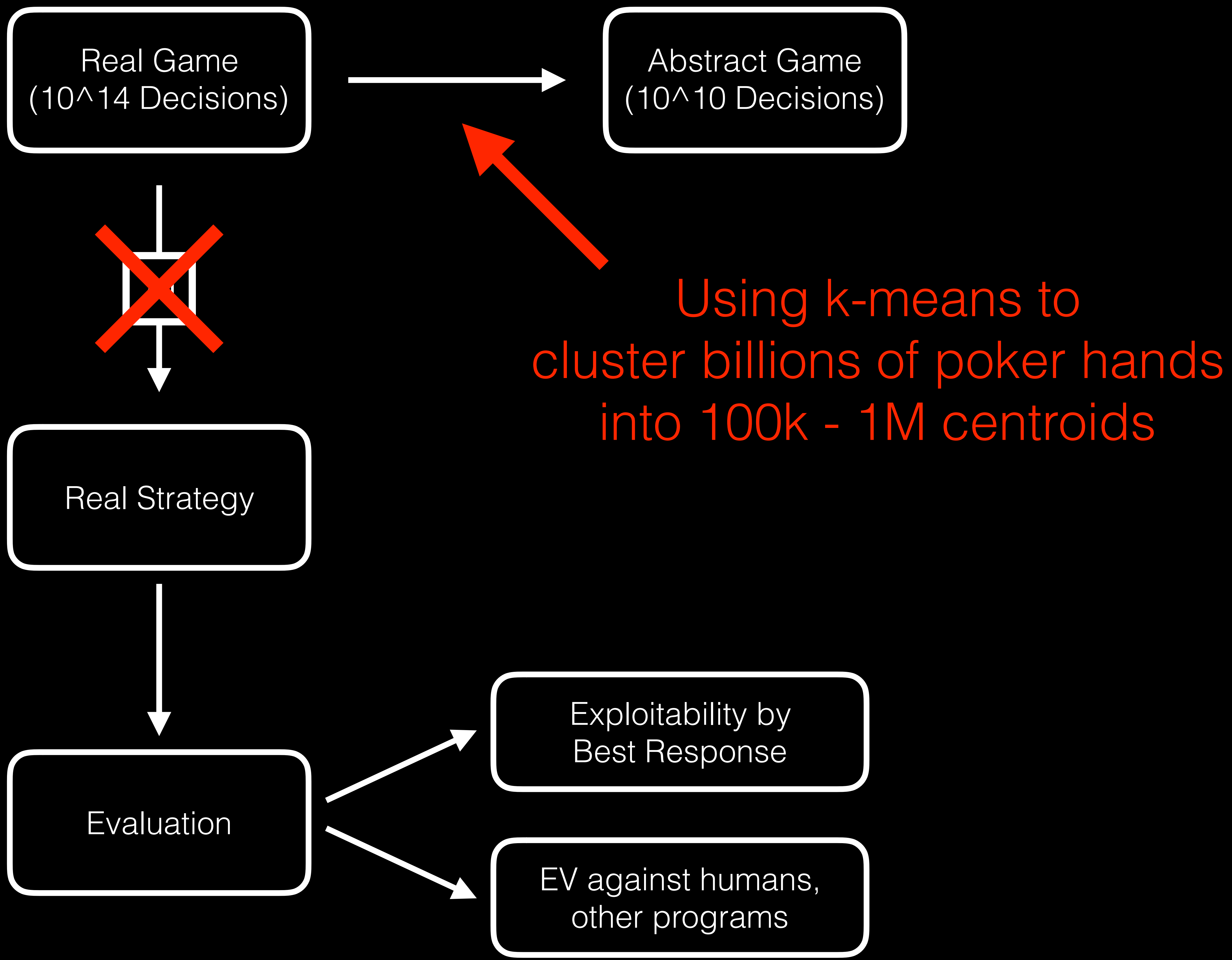

Real Game
(10^14 Decisions)
AI
Real Strategy
Evaluation
Exploitability by
Best Response
EV against humans,
other programs
Problem:
Game has $3.6 * 10^{13}$ actions.
At 16 bytes each…



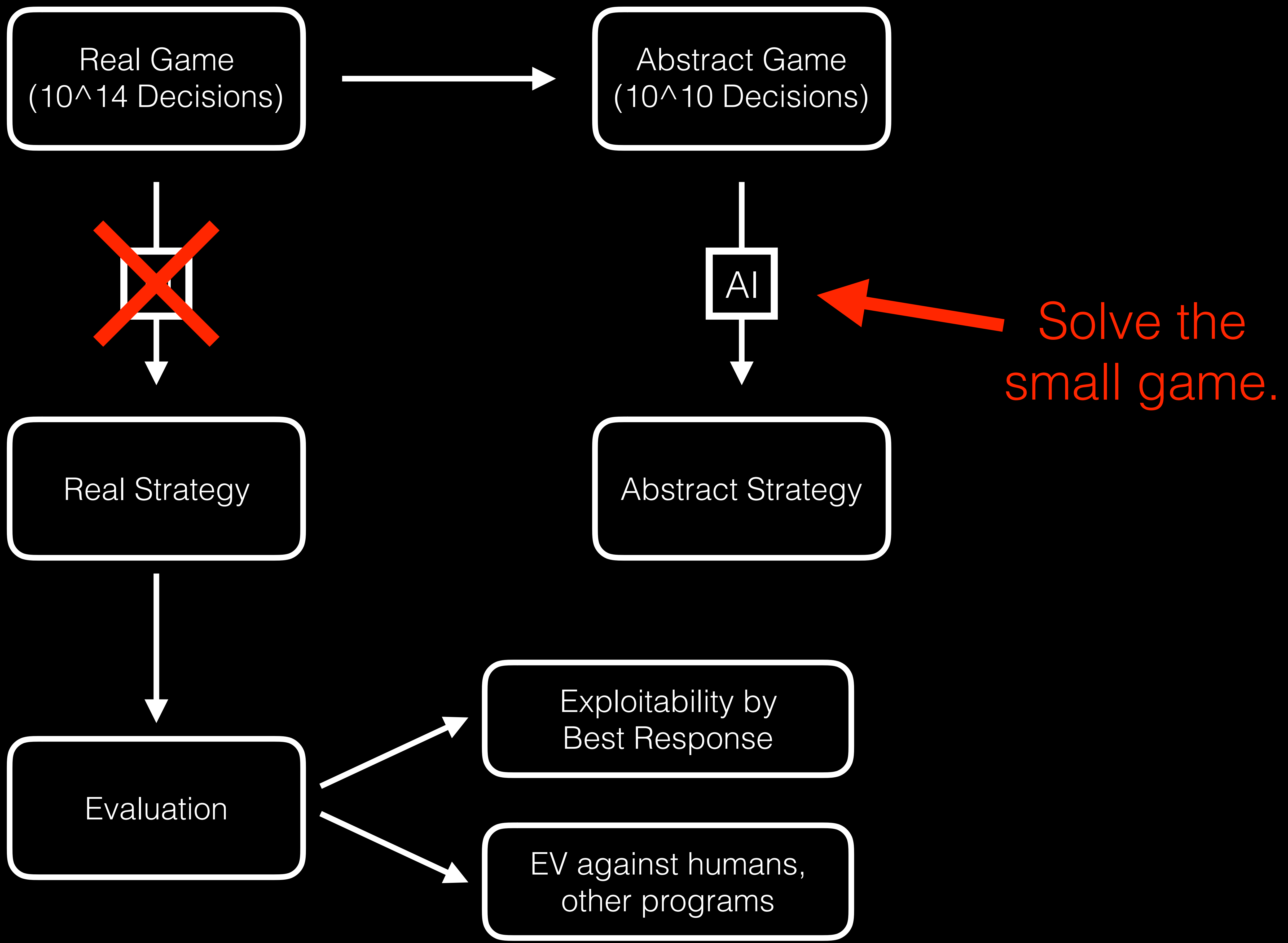
Real Game
(10^14 Decisions)
AI
Real Strategy
Evaluation
Exploitability by
Best Response
EV against humans,
other programs
Problem:
Game has 3.6 *10^13 actions.
At 16 bytes each… 523 TB storage.
:(


Real Game
(10^14 Decisions)
AI
Real Strategy
Evaluation
Exploitability by
Best Response
EV against humans,
other programs
Problem:
Game has 3.6 *10^13 actions.
At 16 bytes each… 523 TB storage.
:(
~10,000 CPU-years runtime.
:(:(:(

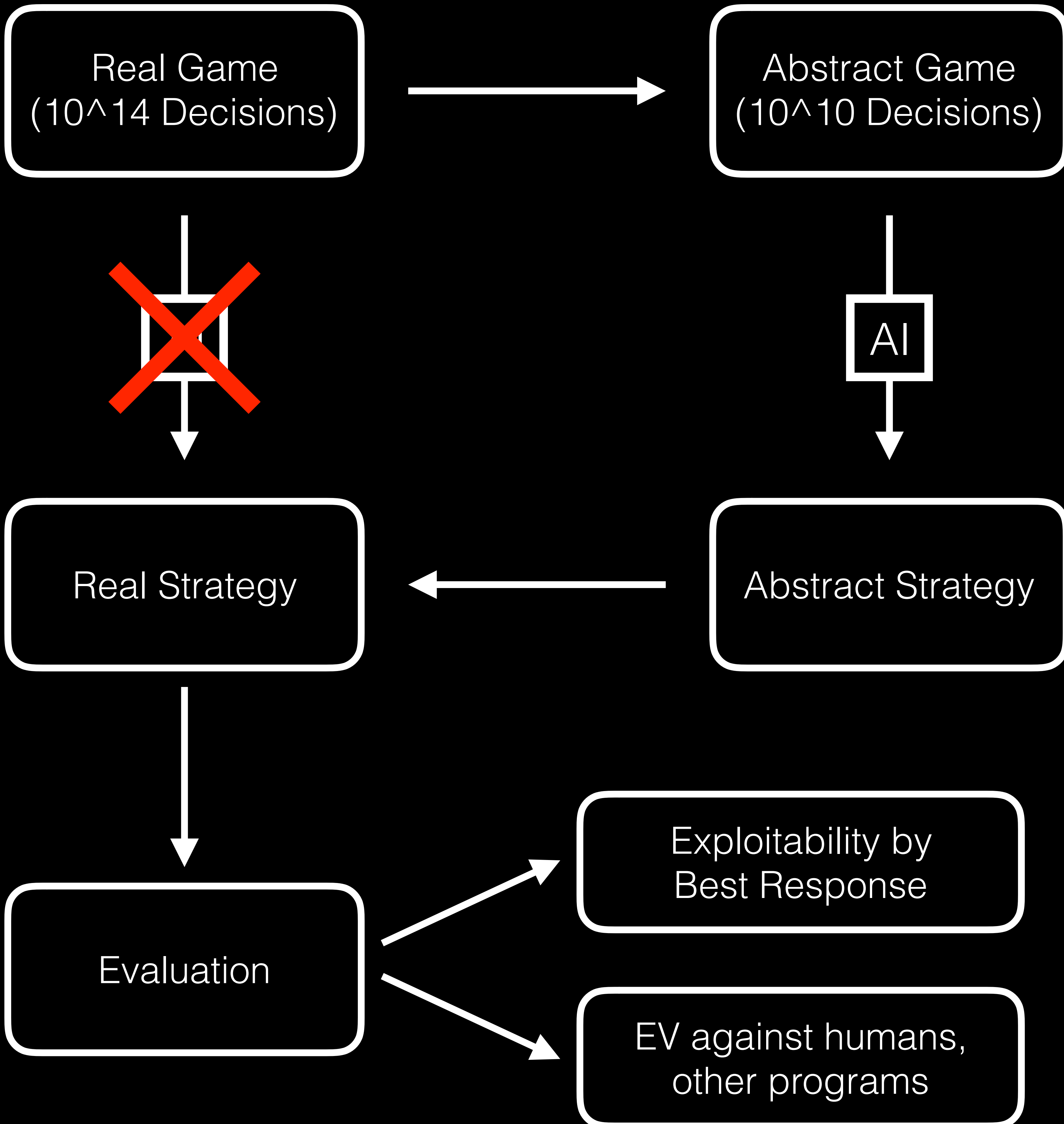

Real Game
(10^14 Decisions)
Abstract Game
(10^10 Decisions)
Workaround:
Cluster similar
decisions together.
Lossy.
Real Strategy
Evaluation
Exploitability by
Best Response
EV against humans,
other programs



Real Game
(10^14 Decisions)
Abstract Game
(10^10 Decisions)
Using k-means to
cluster billions of poker hands
into 100k - 1M centroids
Real Strategy
Evaluation
Exploitability by
Best Response
EV against humans,
other programs

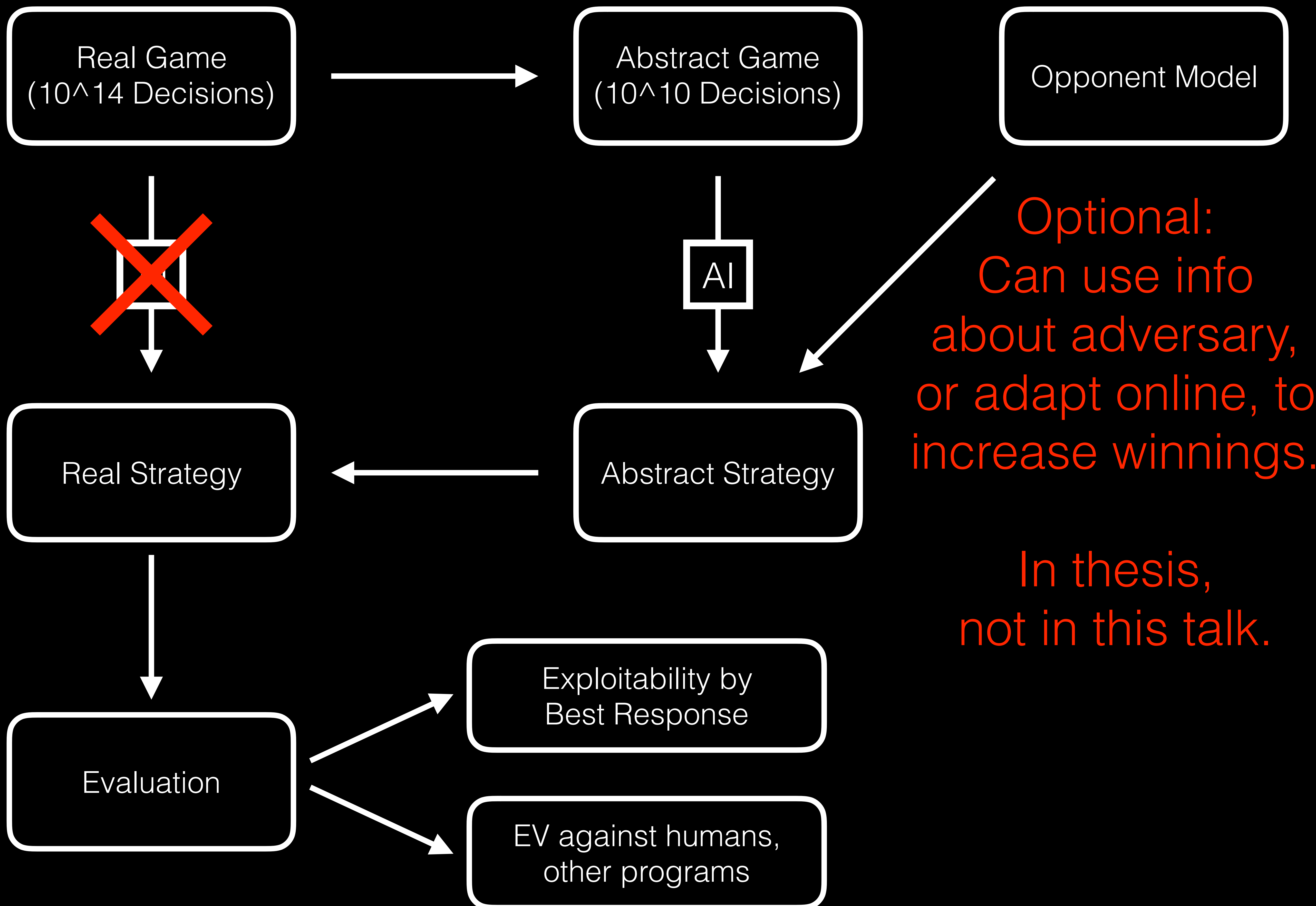


Real Game
(10^14 Decisions)
Abstract Game
(10^10 Decisions)
AI
Solve the
small game.
Real Strategy
Abstract Strategy
Evaluation
Exploitability by
Best Response
EV against humans,
other programs



Use small strategy
to act in the
real game.

NOTE:
Not optimal!
Lossy abstraction!


Real Game
(10^14 Decisions)
Abstract Game
(10^10 Decisions)
Opponent Model
AI
Optional:
Can use info
about adversary,
or adapt online, to
increase winnings.
In thesis,
not in this talk.
Real Strategy
Abstract Strategy
Evaluation
Exploitability by
Best Response
EV against humans,
other programs

Intuition:

Using abstraction limits the strategy’s strength.

Merging decisions together loses information.

Intuition:

Using abstraction limits the strategy's strength.

Merging decisions together loses information.

Bigger (finer-grained)
and
Better (feature-preserving)
abstractions

$\Rightarrow$

Better Strategies:
wins more,
less exploitable

Intuition:

Using abstraction limits the strategy's strength.

Merging decisions together loses information.

Bigger (finer-grained)
and
Better (feature-preserving)
abstractions

→

Better Strategies:
wins more,
less exploitable

Better Computers,
Better Algorithms

→

Can solve
bigger abstractions

Abstraction-Solving-Translation was enough to beat top human pros.

In retrospect, it was easy:
~8 GB RAM, a few CPU-days.
Fairly small abstractions, too!

2007: Narrow loss. 4 GB strategy.
2008: Narrow win. 8 GB strategy.

In 2011, we discovered that these strategies were VERY exploitable.

The Man-vs-Machine strategies were beatable, but small.

At the time, we thought: to be optimal, maybe we just have to solve a big enough abstraction!

If we can reduce exploitability to “1 milli-big-blind”, then it’s *essentially* solved.

Close enough - justification later in this talk.

Solving Attempt #1 (2008-2011):

The Man-vs-Machine strategies were
beatable, but small.

At the time, we thought: to be optimal,
maybe we just have to solve a
big enough abstraction!

If we can reduce exploitability to “1 milli-big-blind”,
then it’s *essentially* solved.

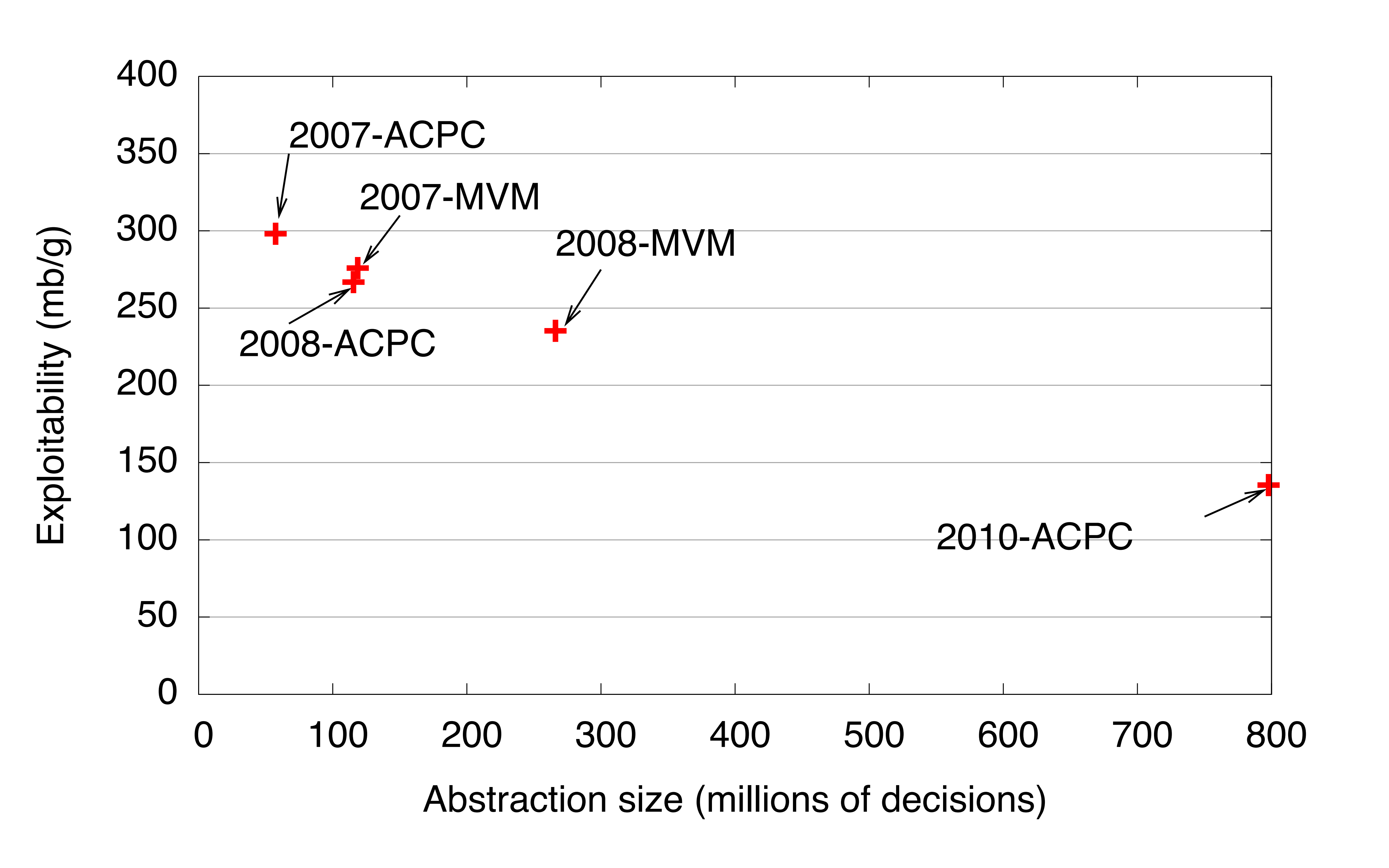
Close enough - justification later in this talk.

In 2011, we wrote a fast algorithm for
finding perfect real-game counter-strategies.
(IJCAI 2011)

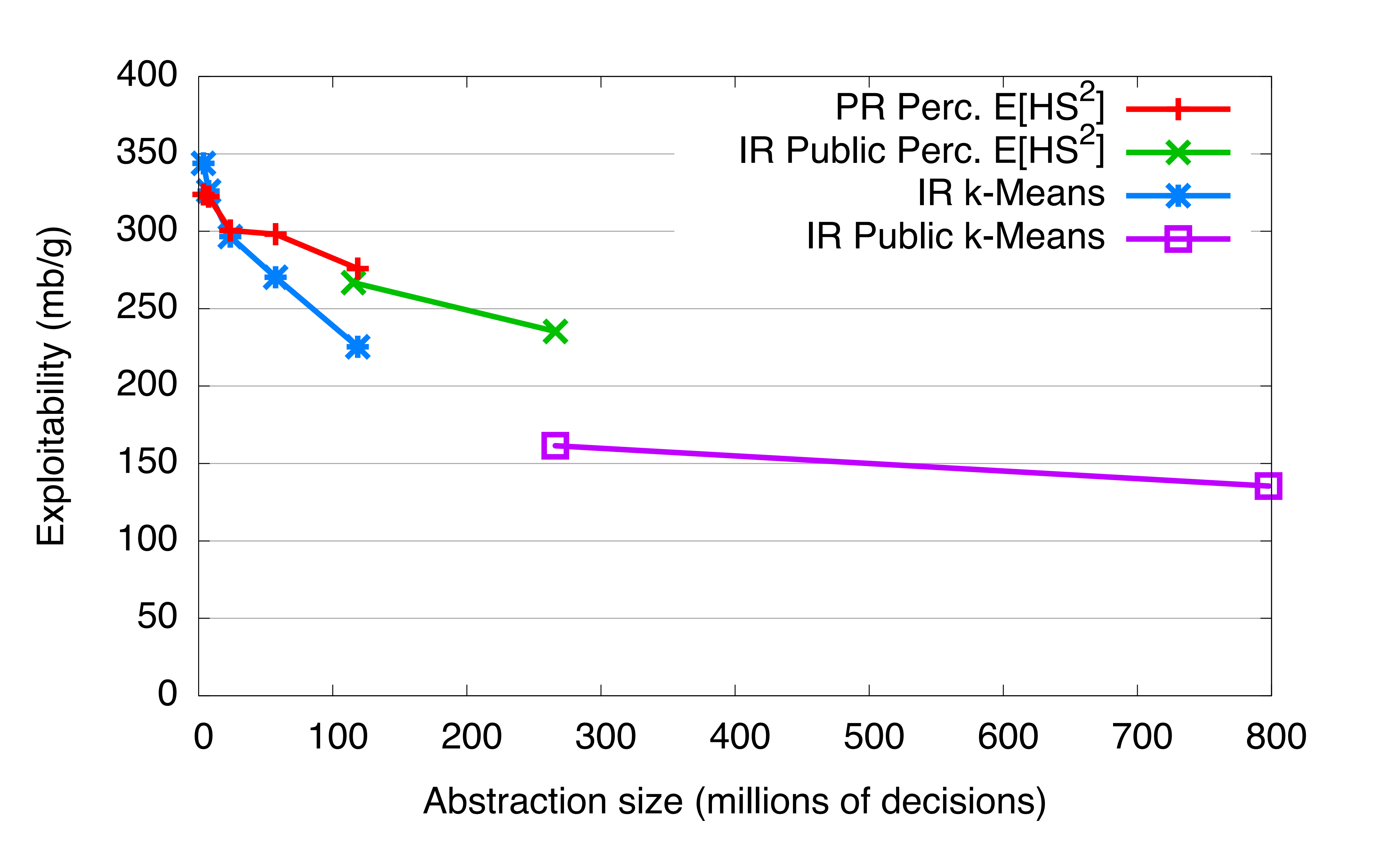
For the first time, we could measure
exploitability!

We turned a 10 CPU-year computation into
a 76 CPU-day computation. 1 day on the cluster.

Looking back at 5 years of progress!

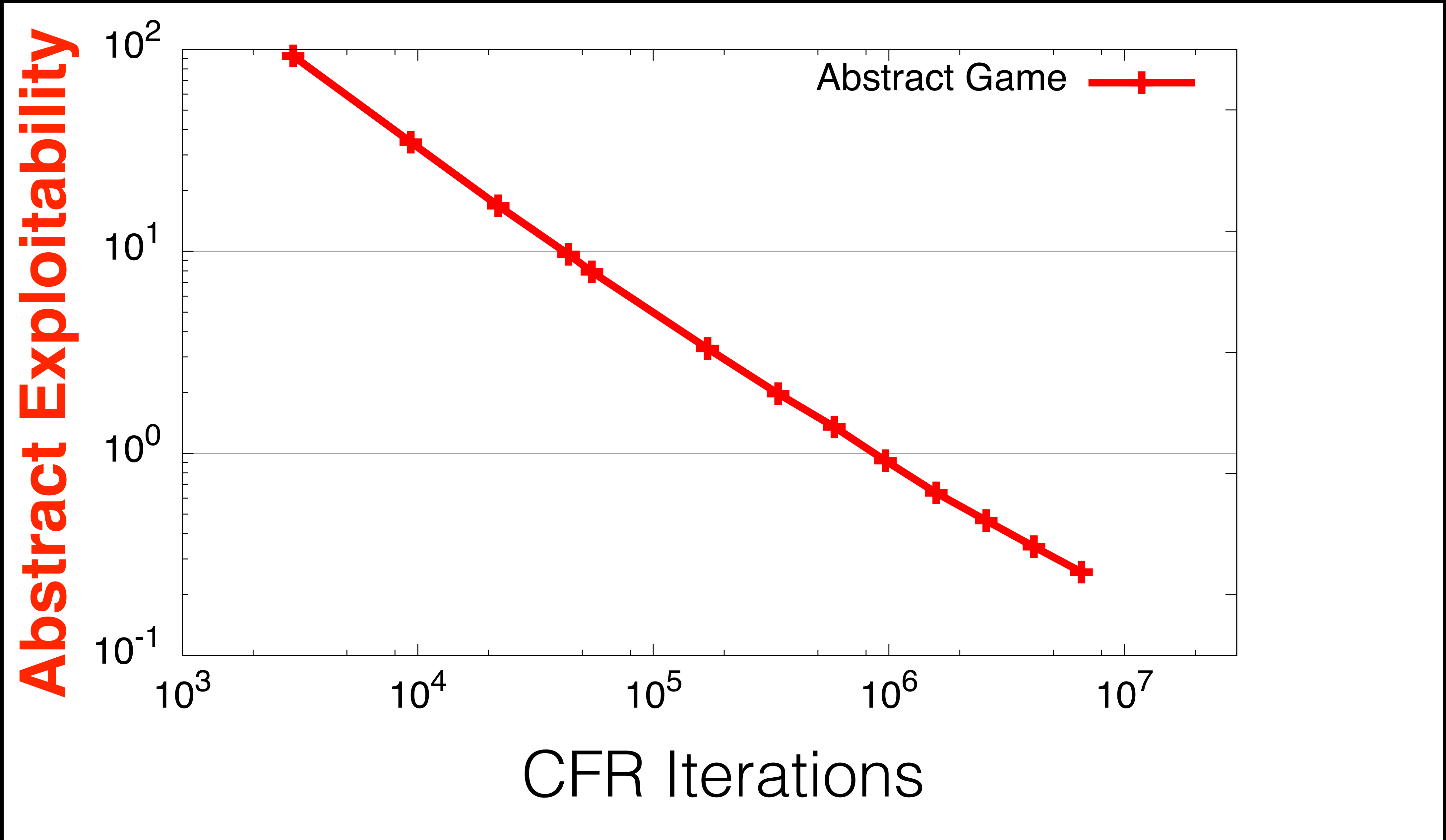


We'll just solve a big enough abstraction!

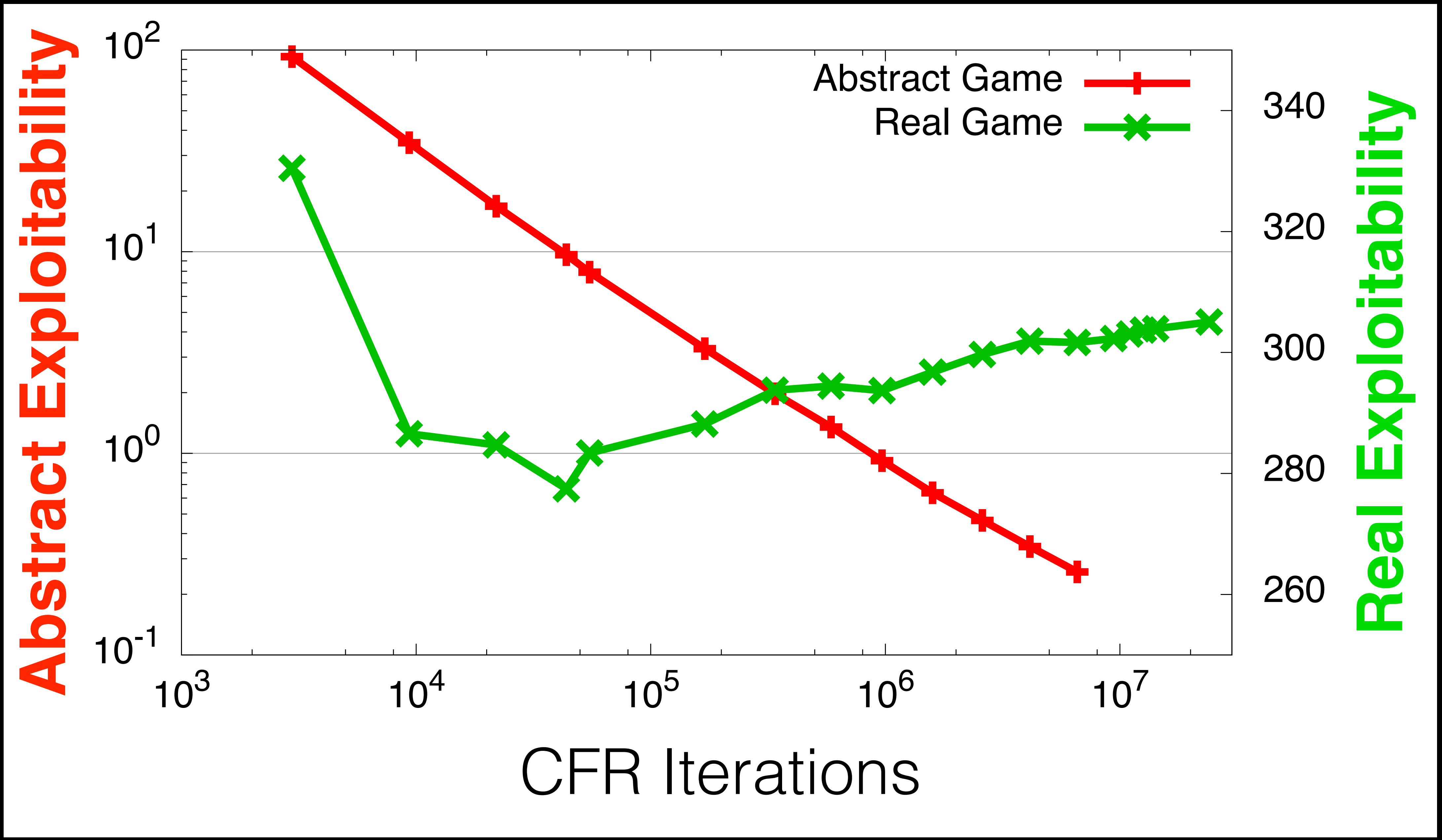


This was worrying… Flattening out already?

…But here's the overfitting effect:



…But here's the overfitting effect:



So: we’re far from solved,
and have a serious problem!

But we’re stuck with abstraction.

Can a different algorithm avoid overfitting?

Solving Attempt #2 (2012):

We'll solve a really big abstraction,
but *properly*,
so we don't overfit.

We’re solving a 2-player game.

If both players use abstraction, we overfit.

We’re solving a 2-player game.

If both players use abstraction, we overfit.

What if one player uses abstraction,
and their opponent doesn’t?

By definition, abstracted player
minimizes exploitability!

CFR-BR (AAAI 2012)

Normally, even one unobstructed player
would cost 262 TB of memory.

But we *can* do it without that much…
The 76-day best response computation does that!

Maybe if we run that in a loop…

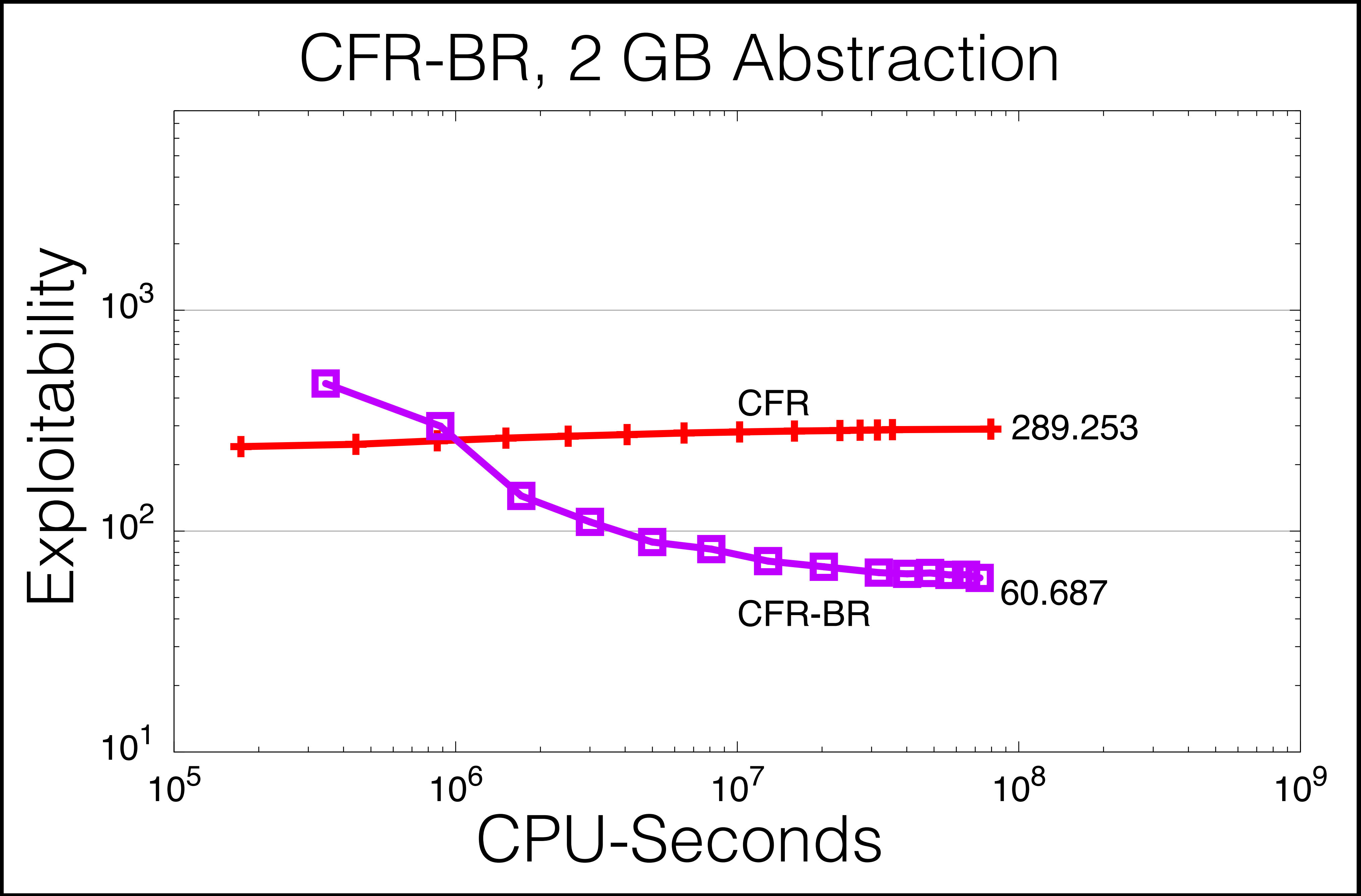
CFR-BR (AAAI 2012)

Normally, even one unobstructed player
would cost 262 TB of memory.

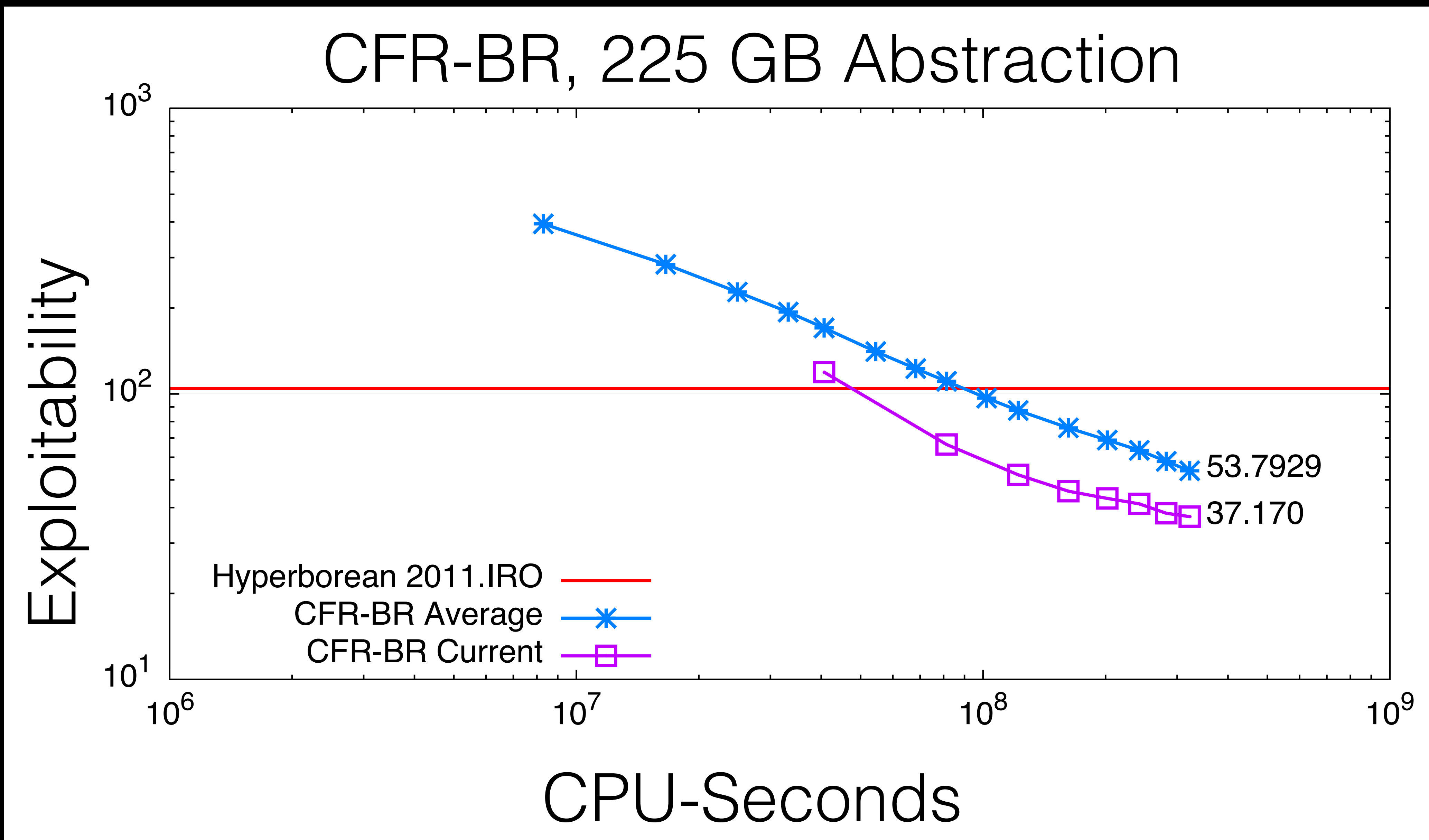
But we *can* do it without that much…
The 76-day best response computation does that!

Maybe if we run that in a loop… and use sampling
tricks to avoid the time cost… it's feasible!

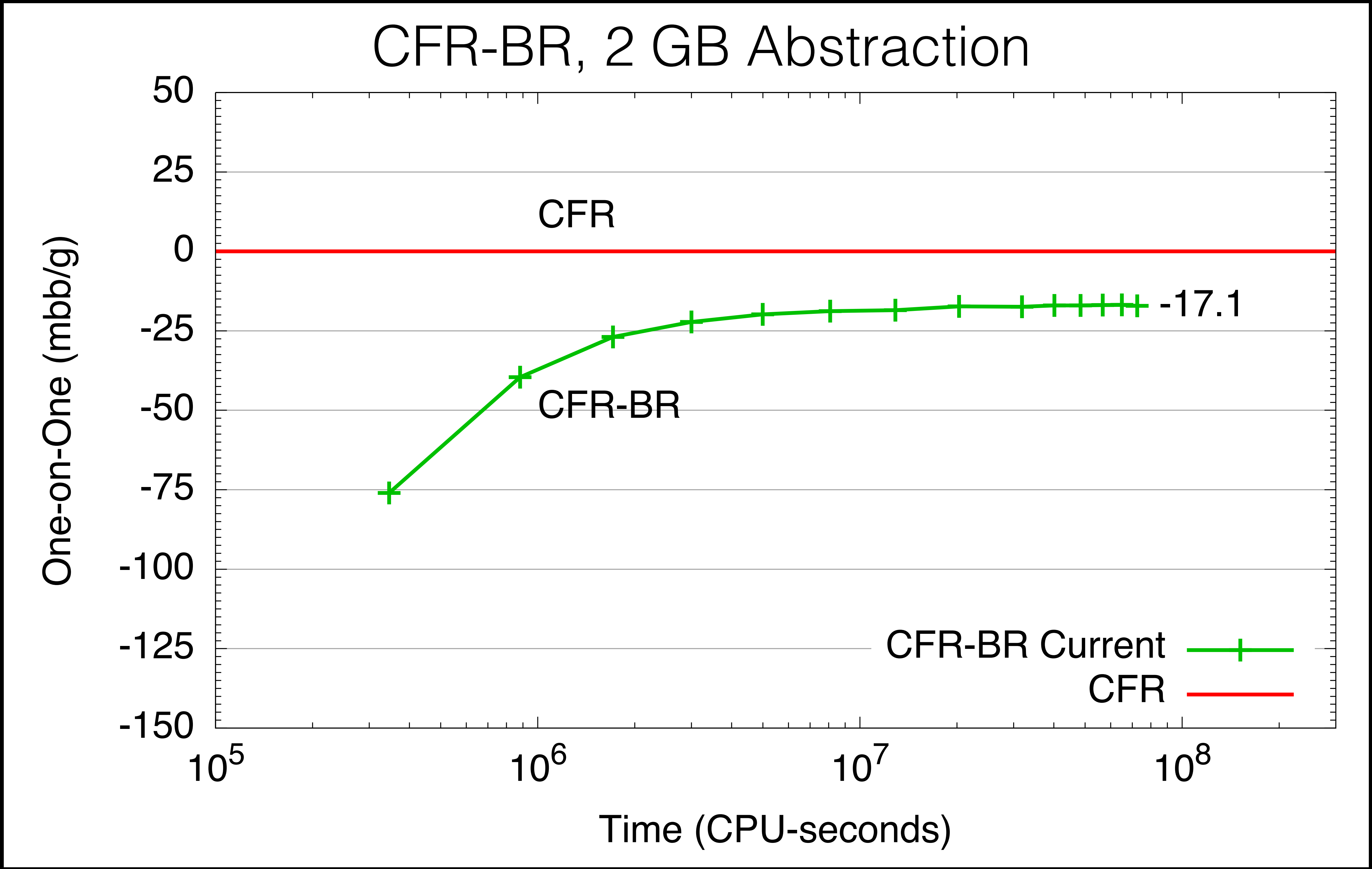
Promising results! CFR-BR has no overfitting, and is far less exploitable!
Small abstraction, but beat all previous strategies!



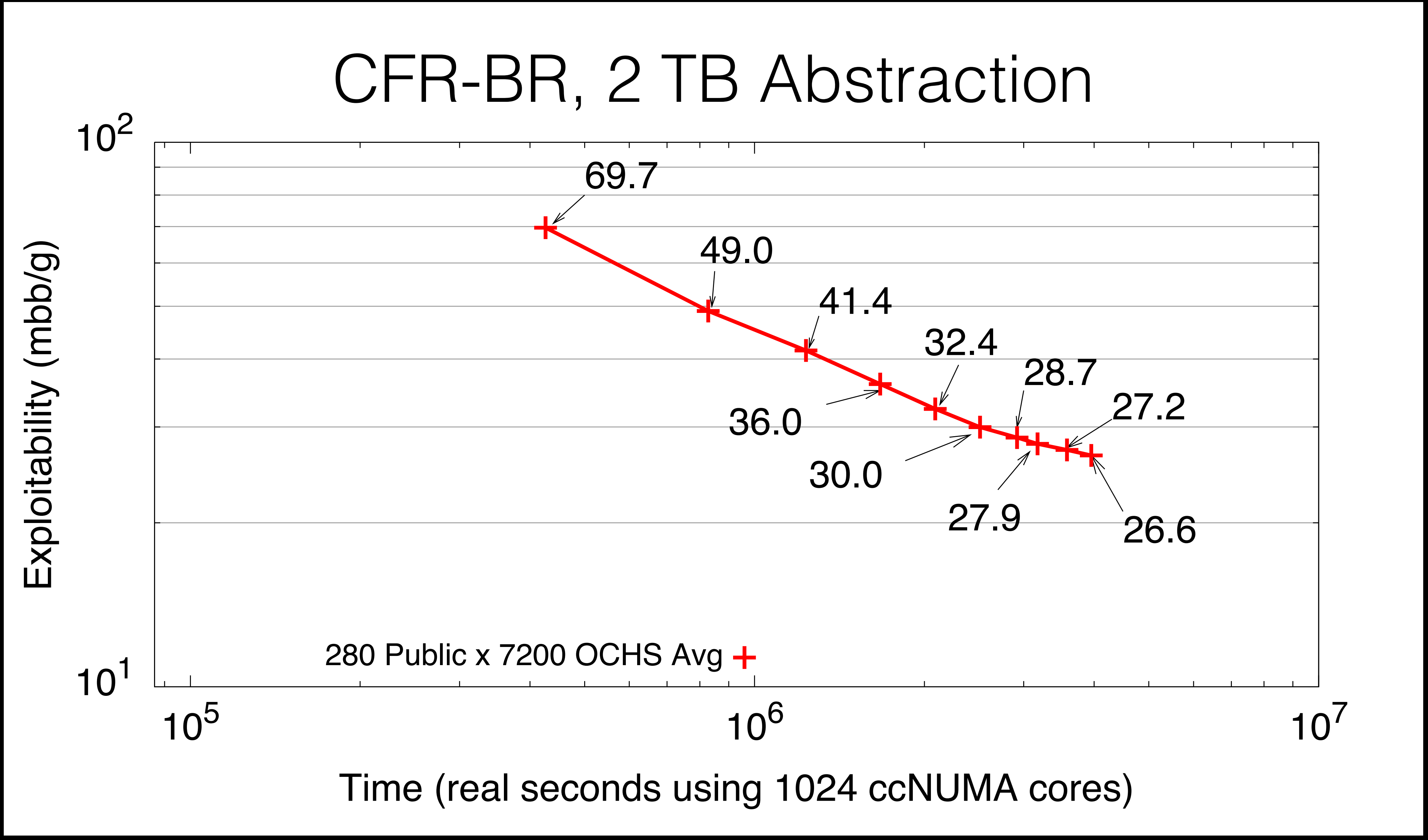
In a big strategy (225 GB to solve), we got closer to optimal than ever before.



However, CFR-BR *lost* in actual games.
Assuming opponent is stronger —> too pessimistic!

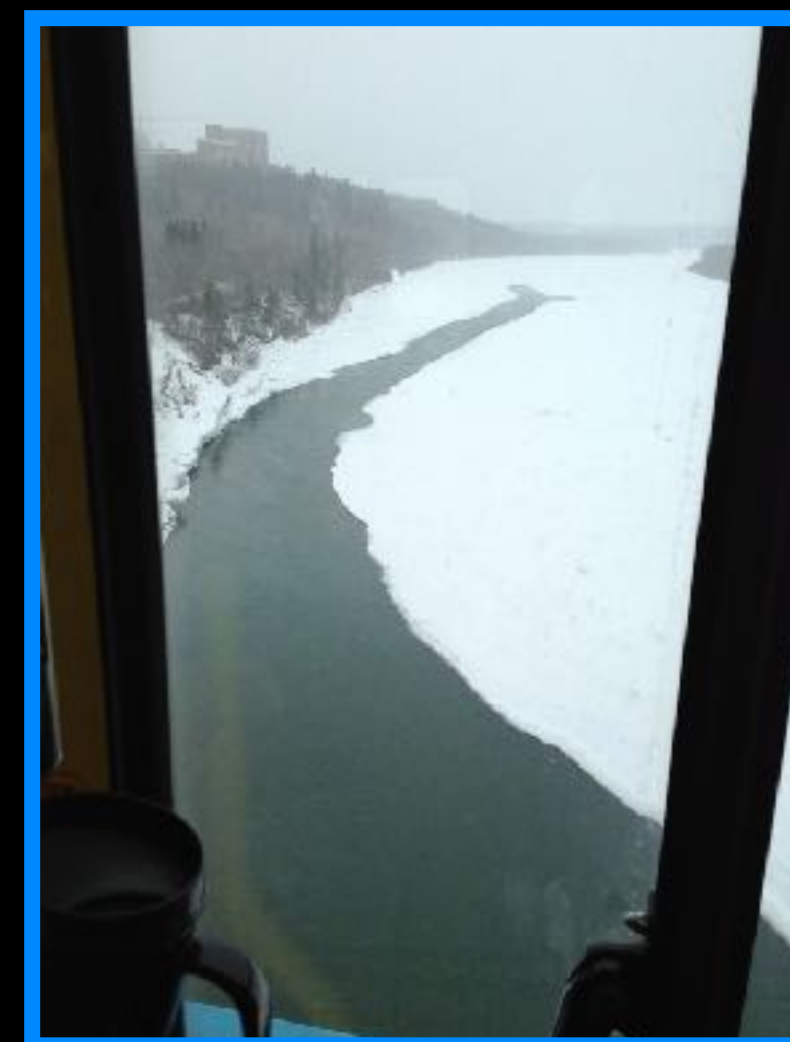


And still wasn't getting low enough:



That last strategy was computed on "Hungabee", an SGI UV 1000 in GSB. 16TB, 2048 cores.

North Saskatchewan River, -10C day



That last strategy was computed on "Hungabee", an SGI UV 1000 in GSB. 16TB, 2048 cores.

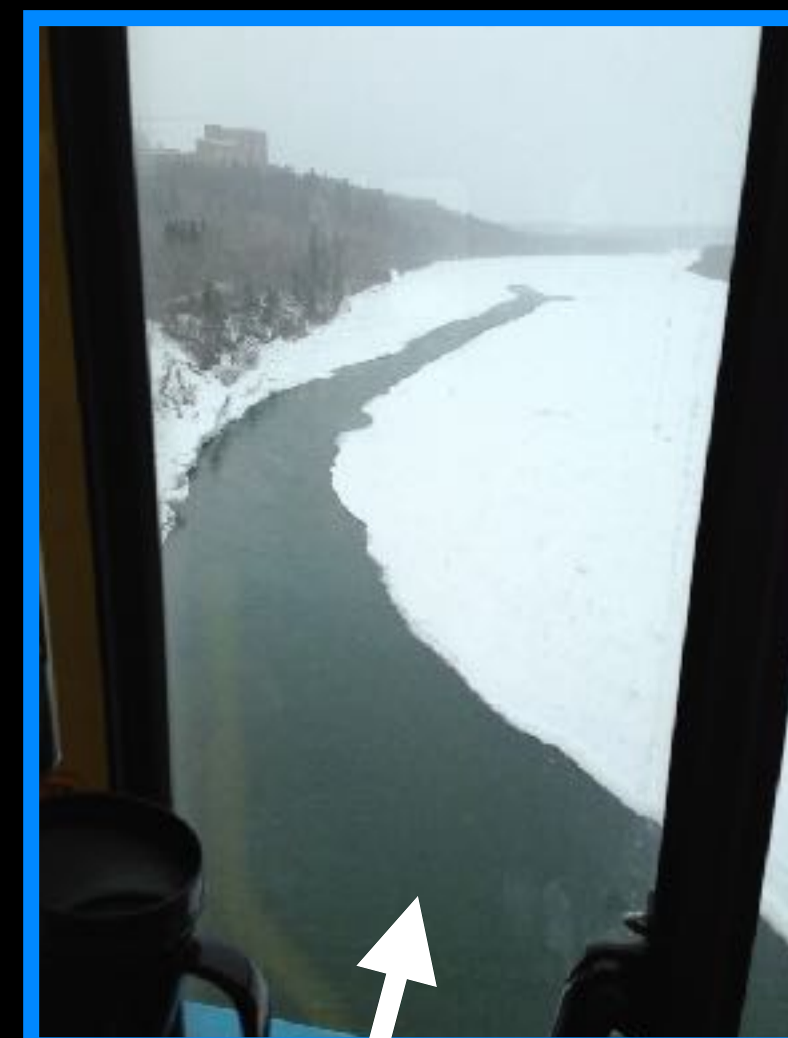
Water cooling, heat dumped to river.

North Saskatchewan River, -10C day

That last strategy was computed on "Hungabee", an SGI UV 1000 in GSB. 16TB, 2048 cores.

Water cooling, heat dumped to river.

North Saskatchewan River, -10C day



Program Output

Solving Attempt #3 (2013):

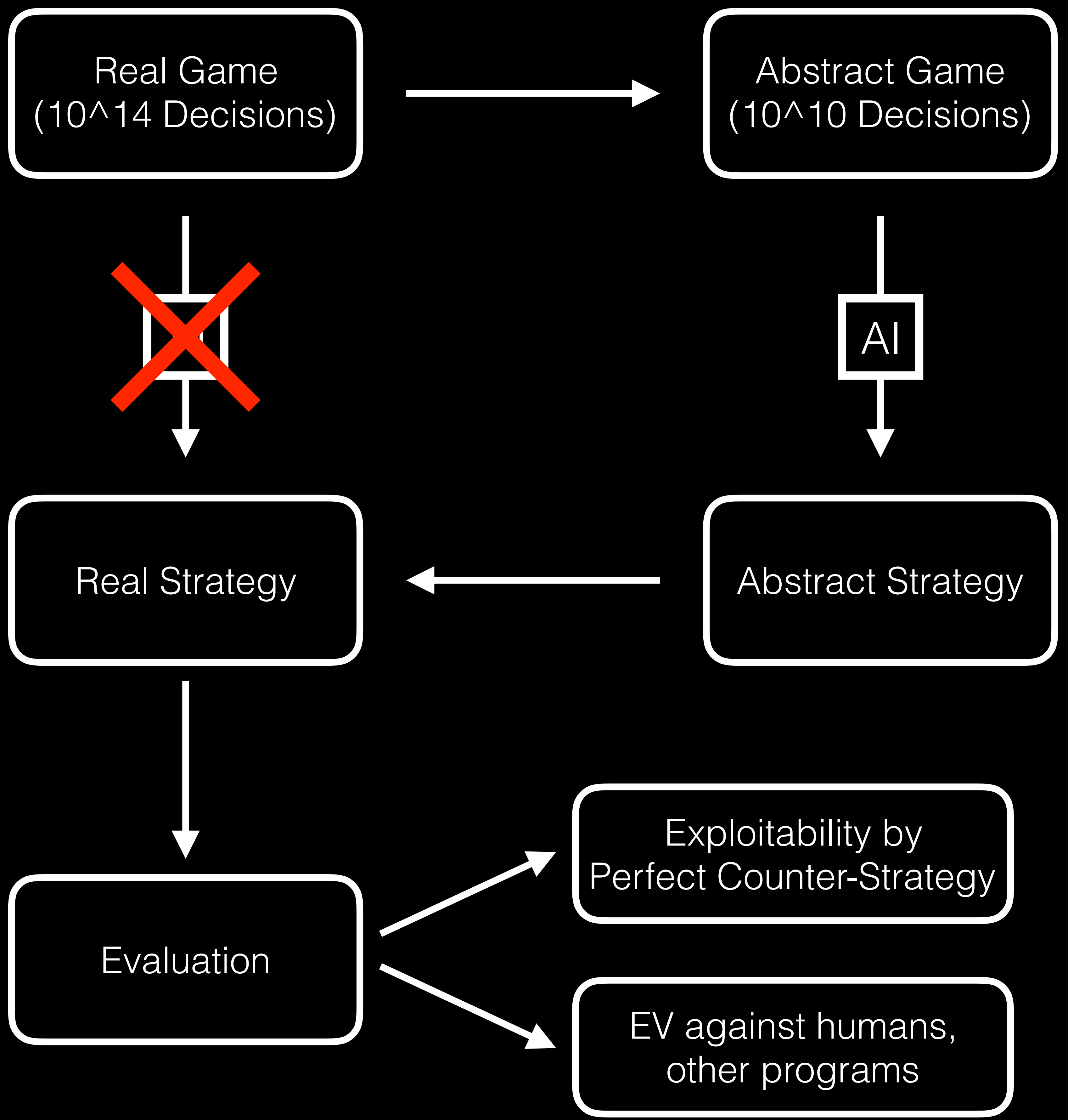
CFR-D: We'll avoid the memory cost
by solving game fragments as needed.

Watch for this in Neil Burch's upcoming thesis!

Flaw: ~16 GB instead of 523 TB of storage…
…but **massive** increase in CPU time required.

Finally:

Heads-Up Limit Texas Hold’em is Solved.
Science, 2015.



Hm. Abstraction is a dead end for perfection.

Was solving it directly *really* infeasible?

Old predictions:
Memory: 523 TB
CPU: ~10k years

In October 2013, our coauthor
Oskari Tammelin contacted us
with two ideas:

1

Poker-specific data compression.
523 TB —> 17 TB

In October 2013, our coauthor
Oskari Tammelin contacted us
with two ideas:

1. Poker-specific data compression.
523 TB —> 17 TB

2. CFR+. A new (at that time theoretically unproven) variant that converges **amazingly** quickly.
Key change: floor regret values at zero.

Third piece:
Massive resources from Compute Canada.

From our earlier attempts,
we had experience with large
distributed programs.

Third piece:
Massive resources from Compute Canada.

"Mammouth" cluster in Quebec.
We used 200 nodes,
24 cores/node. 4800 cores.

Each node had 32 GB RAM,
and 1 TB of local disk.

Each node handled a set of
subgames. Solve with
massive parallelism.

One last wrinkle: *Essentially* Solving a Game

Our algorithms converge towards optimal play in the limit.

“Solved” means unbeatable. We can only approximate it.
So how close is “close enough”?

One last wrinkle: *Essentially* Solving a Game

What if a human lifetime of play wasn't enough for someone to claim to beat our program?

One last wrinkle: *Essentially* Solving a Game

What if a human lifetime of play wasn't enough for someone to claim to beat our program?

(200 games/hour) * (12 hours/day) * (70 years)
= 60 million games.

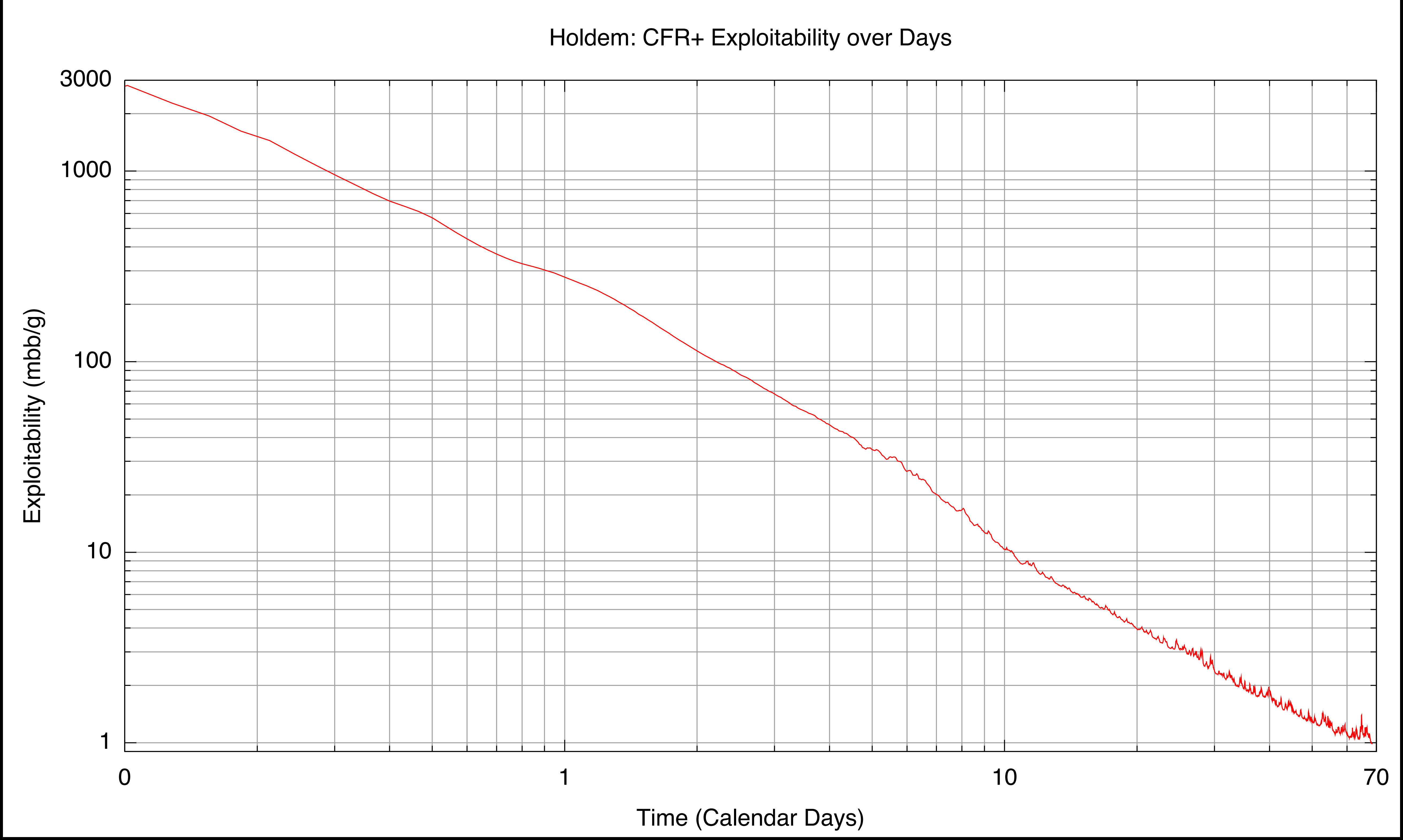
One last wrinkle: *Essentially* Solving a Game

What if a human lifetime of play wasn't enough for someone to claim to beat our program?

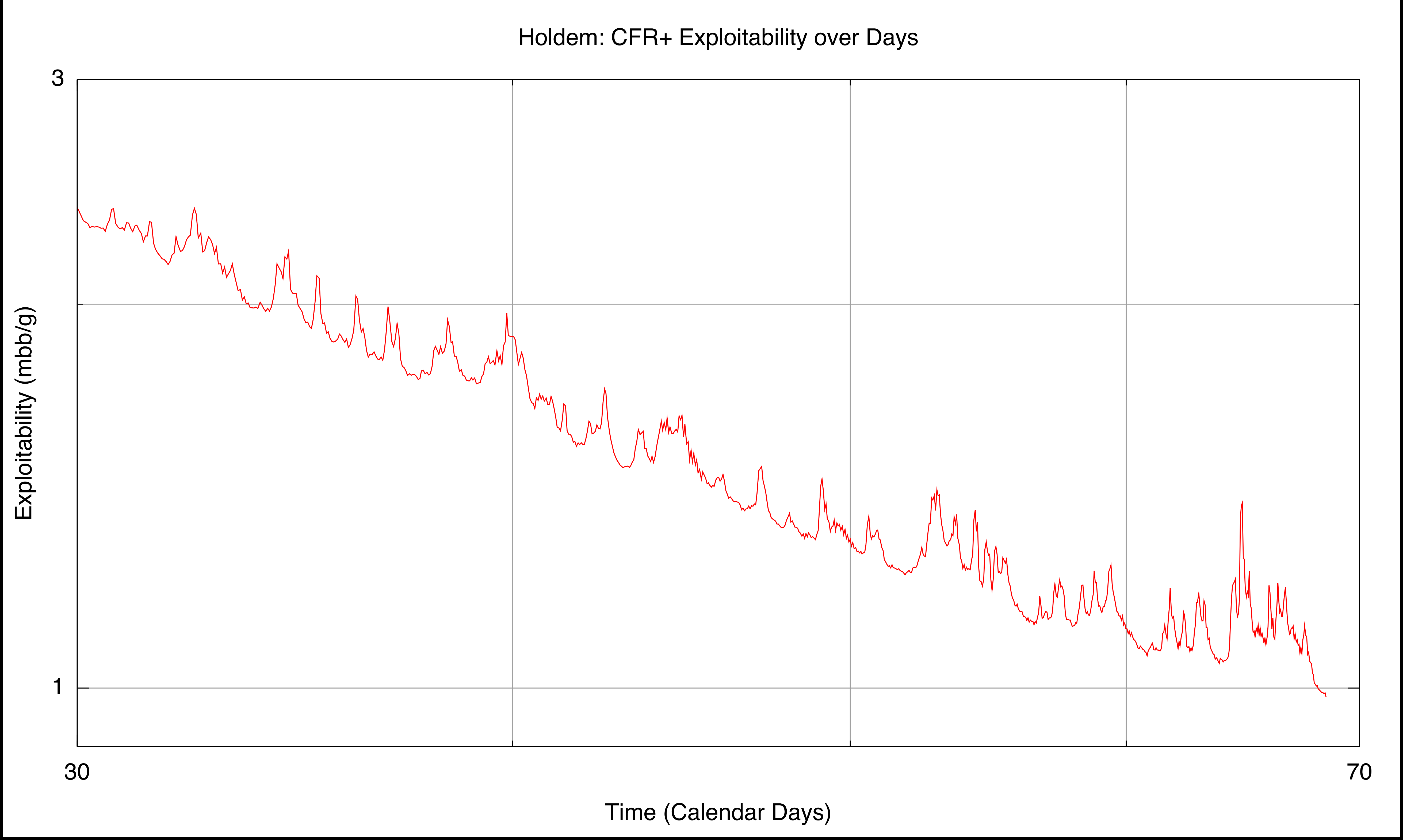
(200 games/hour) * (12 hours/day) * (70 years) = 60 million games.

That isn't enough to discern "1 milli-big-blind" of exploitability with 95% confidence.
So that's our goal.

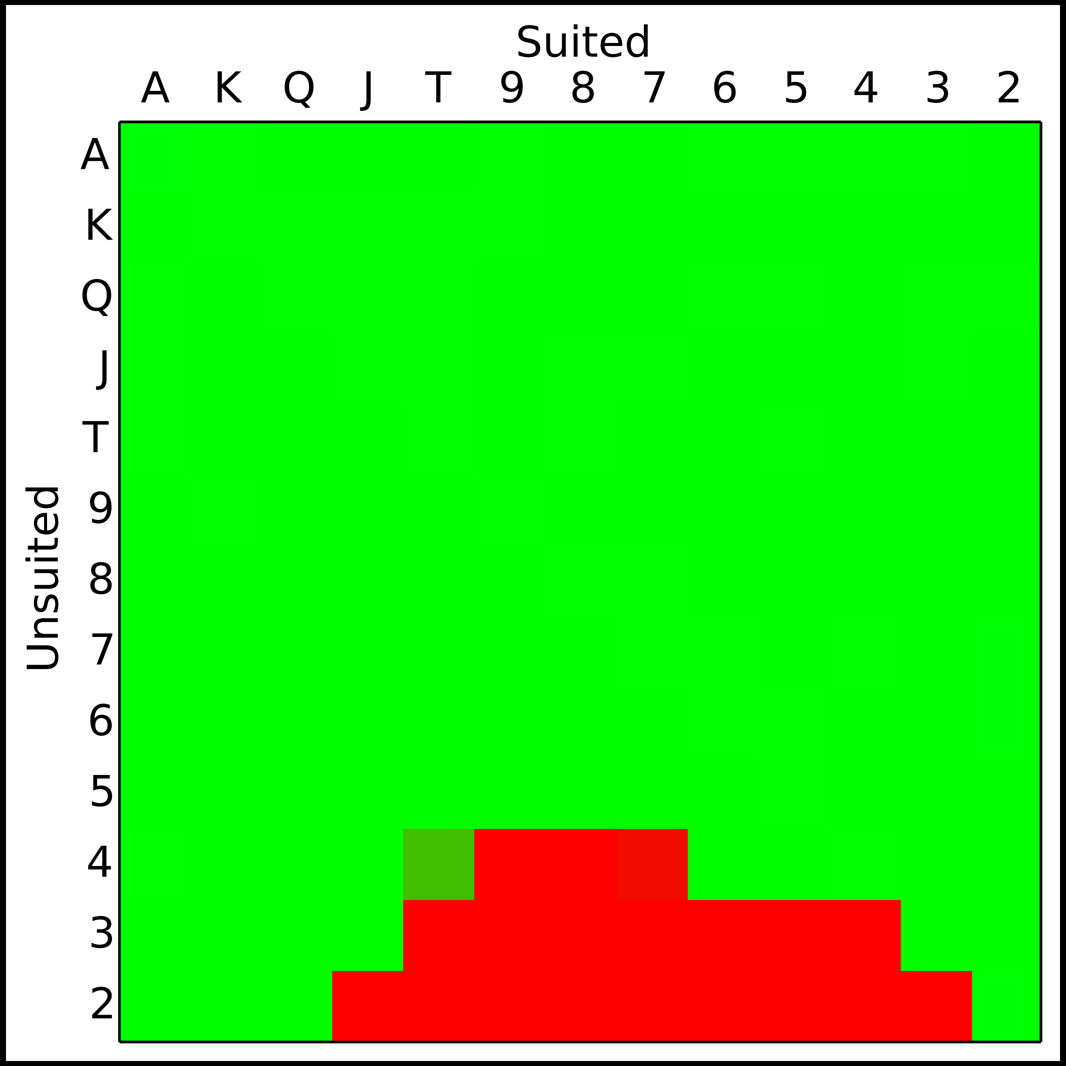
After 70 days (900 CPU-years), we reached 0.986 mbb/g. Essentially solved.



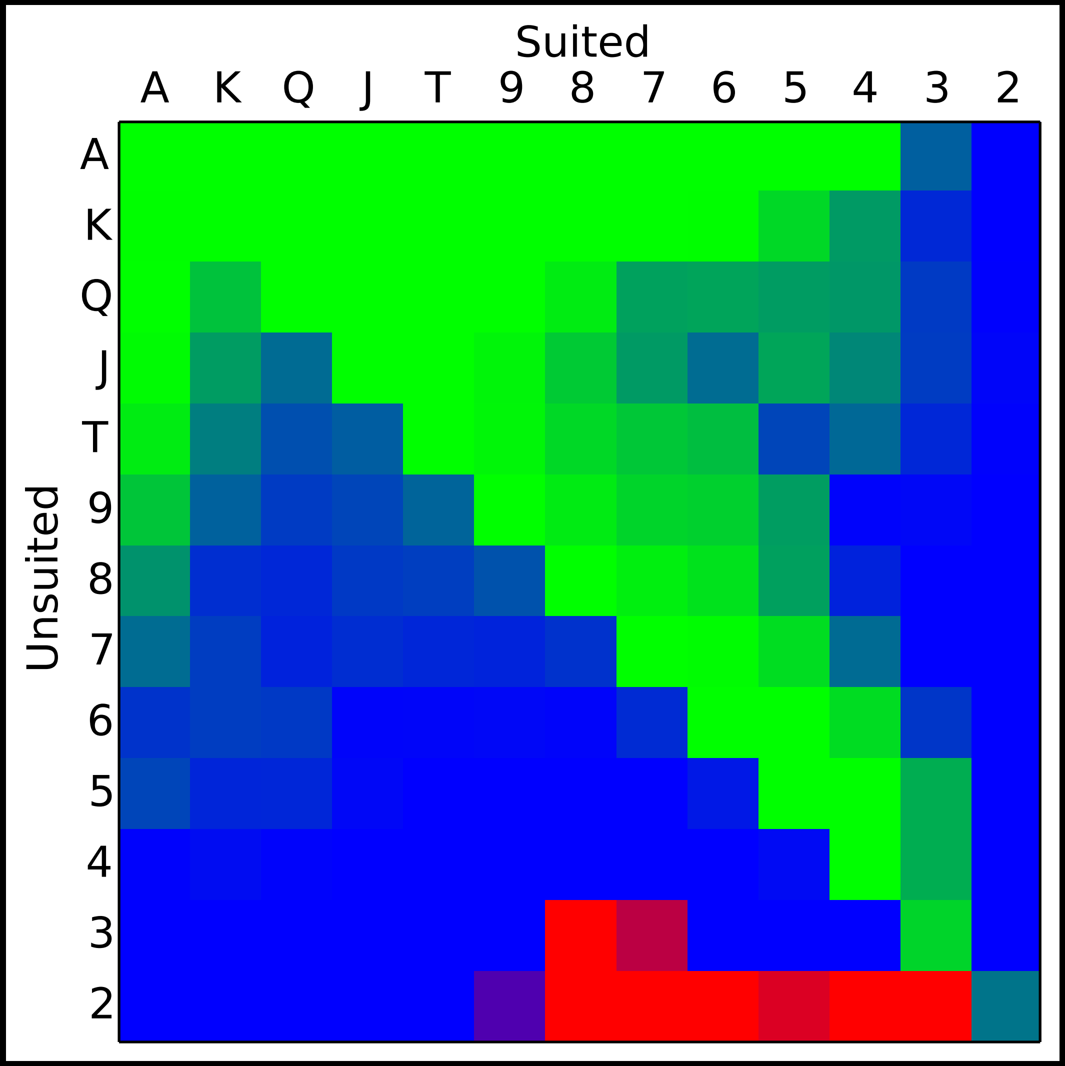
After 70 days (900 CPU-years),
we reached 0.986 mbb/g.
Essentially solved.



Game Start

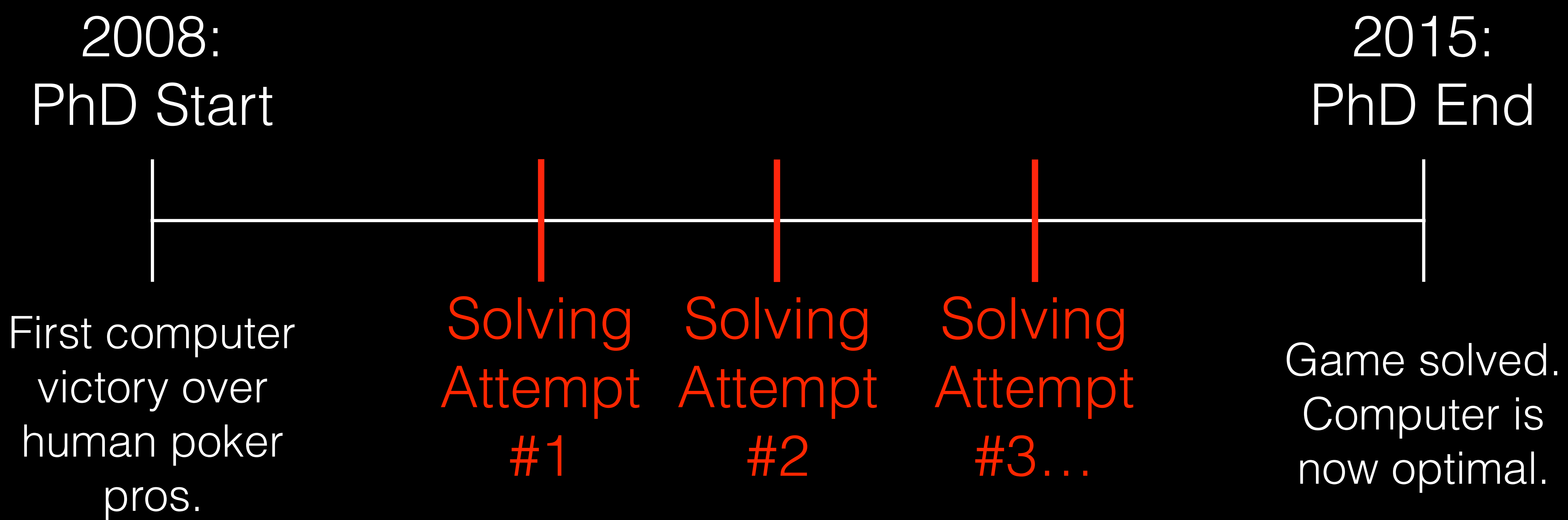


After a Raise



Play against it, inspect strategy, download the code:
http://poker.srv.ualberta.ca

Conclusion:



- My research spanned the End-to-End task of Abstraction-Solving-Translation

- Much easier to surpass humans than to be perfect!

- General set of tools: applicable to other games, and outside the games domain entirely.