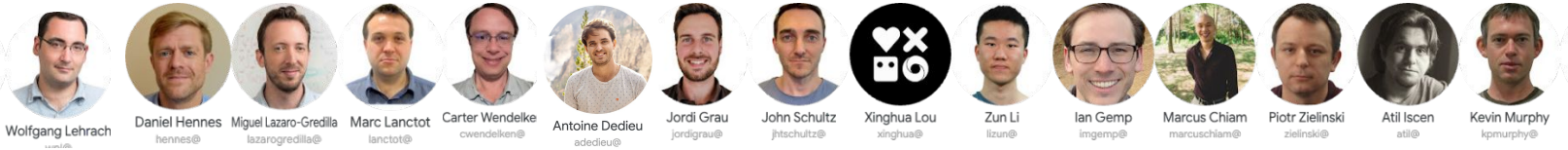
Google DeepMind

Code Synthesis for Improving Agentic Decision Making

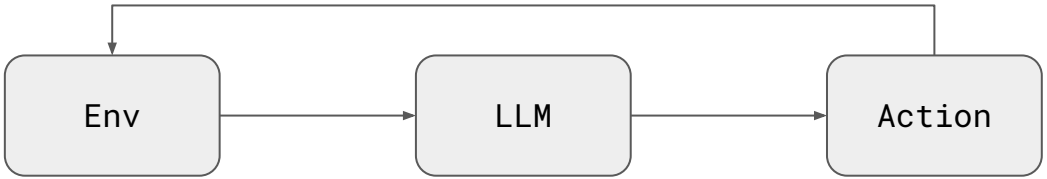
Kevin Murphy

April 2026

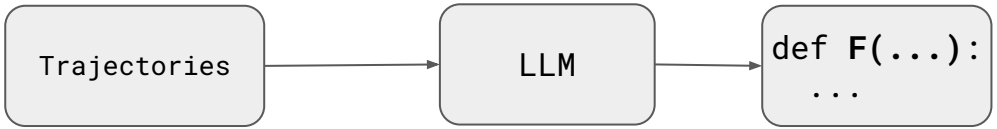


Standard approaches to LLM-based decision making

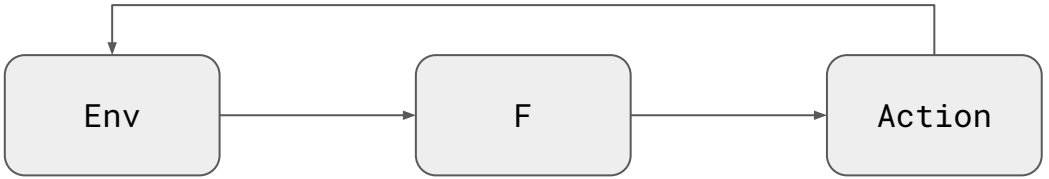
LLM-as-policy



Code-as-policy (Offline training)

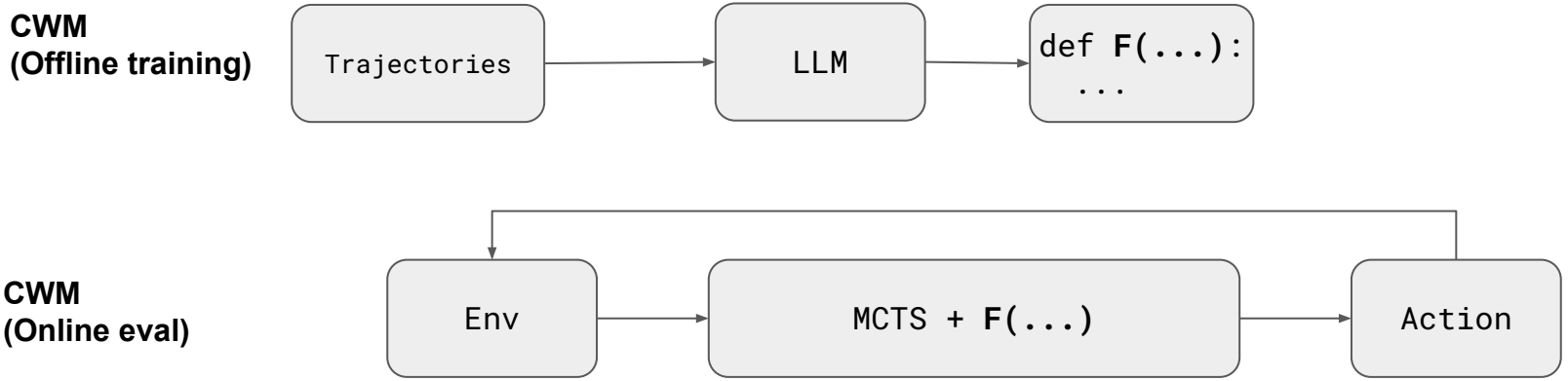


Code-as-policy (online eval)



Our approach 1: Code as world model*

WM = F: P(s(t+1) | s(t), a(t)) [and F': P(o(t) | s(t) for POMDP]



Code World Models for General Game Playing

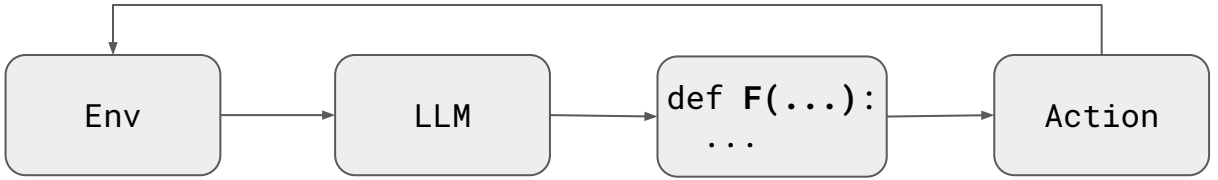
Wolfgang Lehrach, Daniel Hennes, Miguel Lazaro-Gredilla, Xinghua Lou, Carter Wendelken, Zun Li, Antoine Dedieu, Marc Lanctot, Atil Iscen, John Schultz, Marcus Chiam, Ian Gemp, Piotr Zielinski, Satinder Singh, Kevin Patrick Murphy

Published: 26 Jan 2026, Last Modified: 10 Apr 2026 ICLR 2026 Poster Everyone Revisions BibTeX

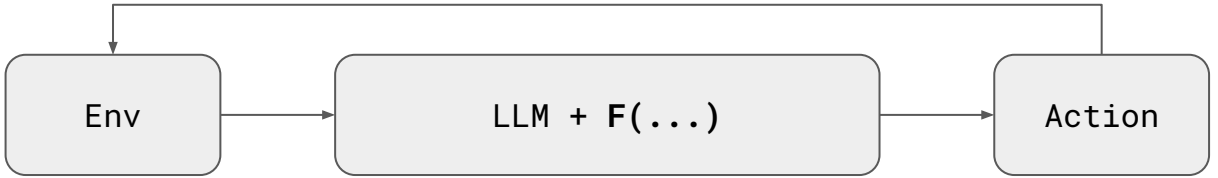
*In this talk, we ignore the issue of temporal abstraction, so focus on one-step models

Our approach 2: Code-as-harness

Code-as-harness (Online training)



Code-as-harness (Eval)



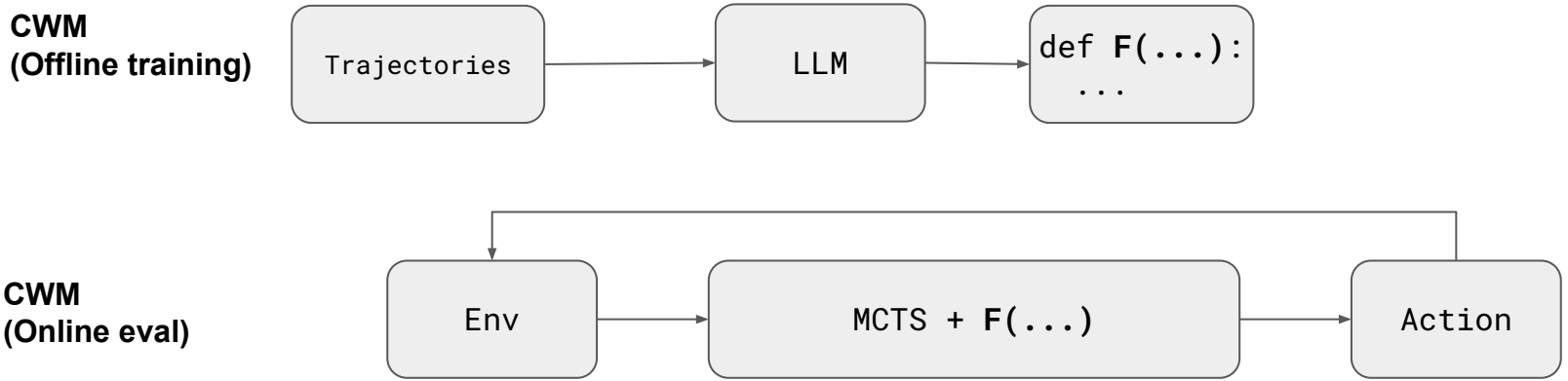
AUTOHARNESS: IMPROVING LLM AGENTS BY AUTOMATICALLY SYNTHESIZING A CODE HARNESS

Xinghua Lou, Miguel Lazaro-Gredilla, Antoine Dedieu, Carter Wendelken, Wolfgang Lehrach, Kevin Murphy

Published: 05 Mar 2026, Last Modified: 05 Mar 2026 ICLR 2026 Workshop RSI Poster Everyone Revisions

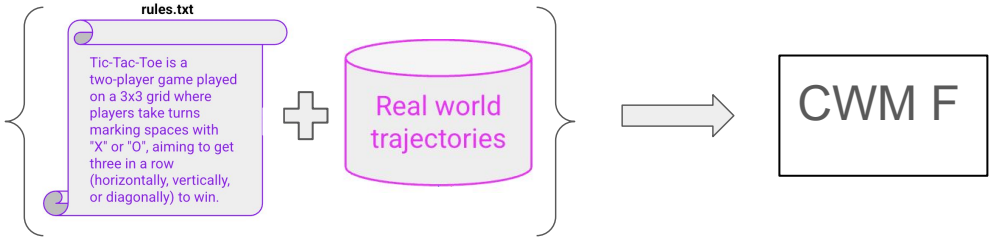
Code-as-world model: Why?

- Why not use LLM-as-policy?
  - Low quality actions, especially in adversarial multi-player games and/or novel (OOD) games
  - Slow at test time
  - Training LLM using RL is very slow and sample inefficient
- Why not use LLM to learn code-as-policy?
  - This can work well for simple environments
  - But it can be hard to learn a parametric function that does well in adversarial multi-player games (due to non-stationarity and need for strategic reasoning)



Code-as-world model: How?

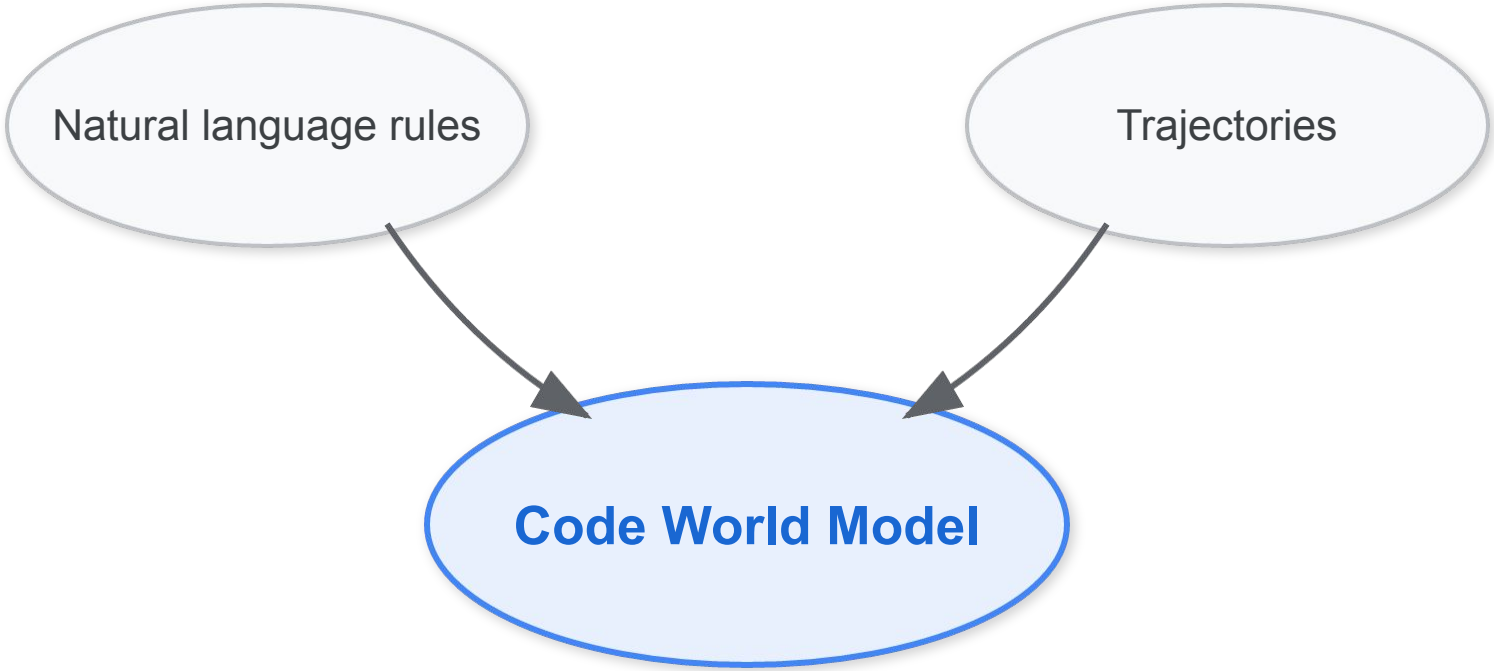
Offline learning



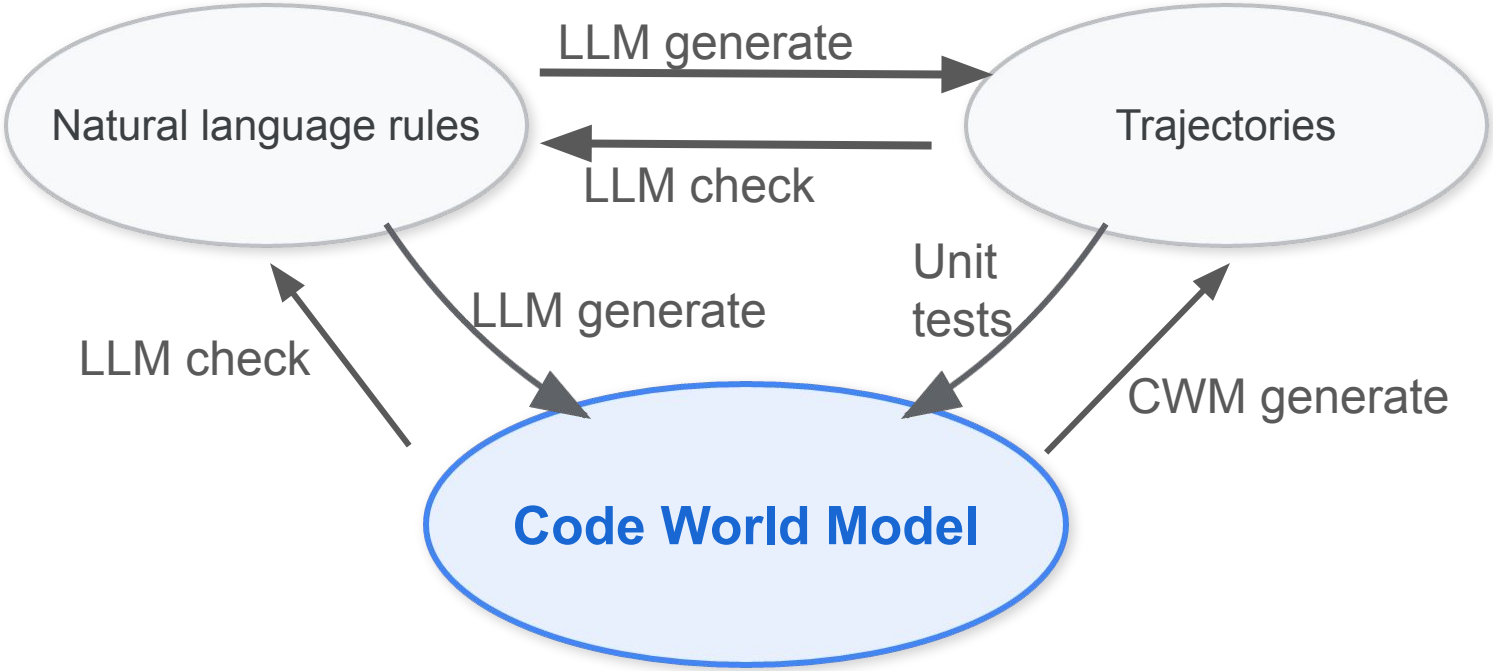
Online decision-making using MCTS

For single player games, we also tried **model-based RL** (cf Dyna, Dreamer), that learns an amortized policy using F as a simulator, which avoids MCTS at run time.

Why use both rules and trajectories?



Consistency between representations can be a powerful cue



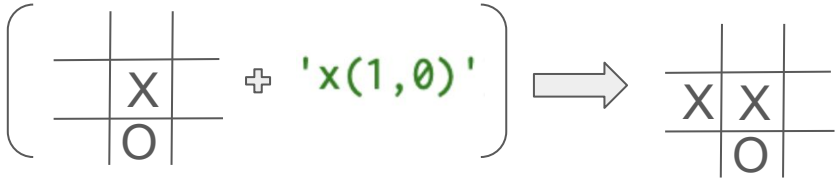
How to learn F?

-Search over python functions to find one that fits the transition data

$$\hat{F} = \arg\max_F \text{score}(F)$$
$$\text{score}(F) = \sum_i \sum_{(s_t, \mathbf{a}_t, s_{t+1}) \in \text{traj}_i} \mathbf{I}[F(s_t, \mathbf{a}_t) == s_{t+1}]$$

- Blackbox opt with scalar feedback is too inefficient. Add text feedback!
- Need an optimizer that does “Text gradient descent”

Score(F) = #passing unit tests (tic tac toe)



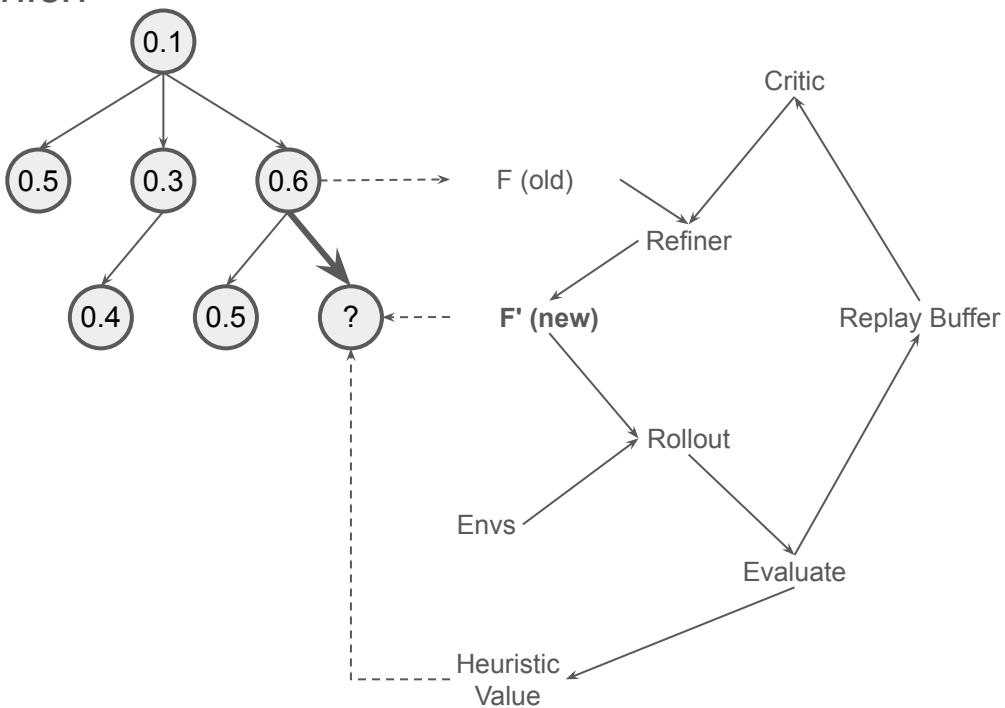
Algorithm 1: Test (fully observed)

Input: $(s_{1:T}, \mathbf{a}_{1:T}^{0:N}; F)$
for $t = 1 : T$ **do**
    assert$(s_{t+1} == F(s_t, \mathbf{a}_t))$

Textual feedback to LLM optimizer!

Tree search (Rex)

-Nodes contain previously generated functions
-Use Thompson sampling to decide which node to expand next at each step
- LLM improves node using Scalar and text feedback



*"REx: Code Repair with LLMs gives an Exploration-Exploitation Tradeoff". Tang et al, 2024.

Population Based search (AlphaEvolve)

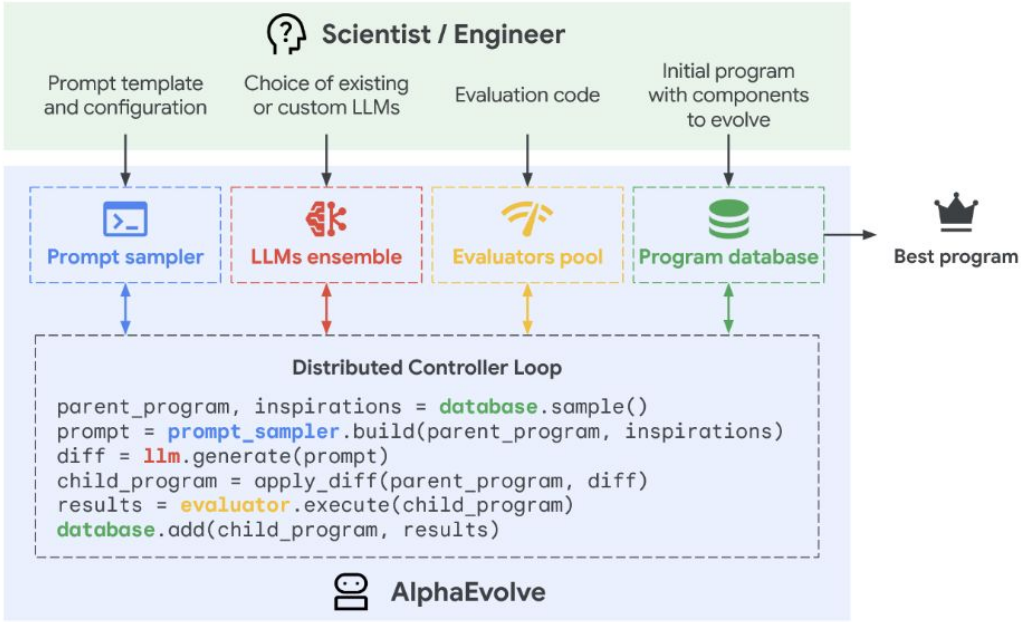


Figure 1. AlphaEvolve discovery process. Adapted from [3]

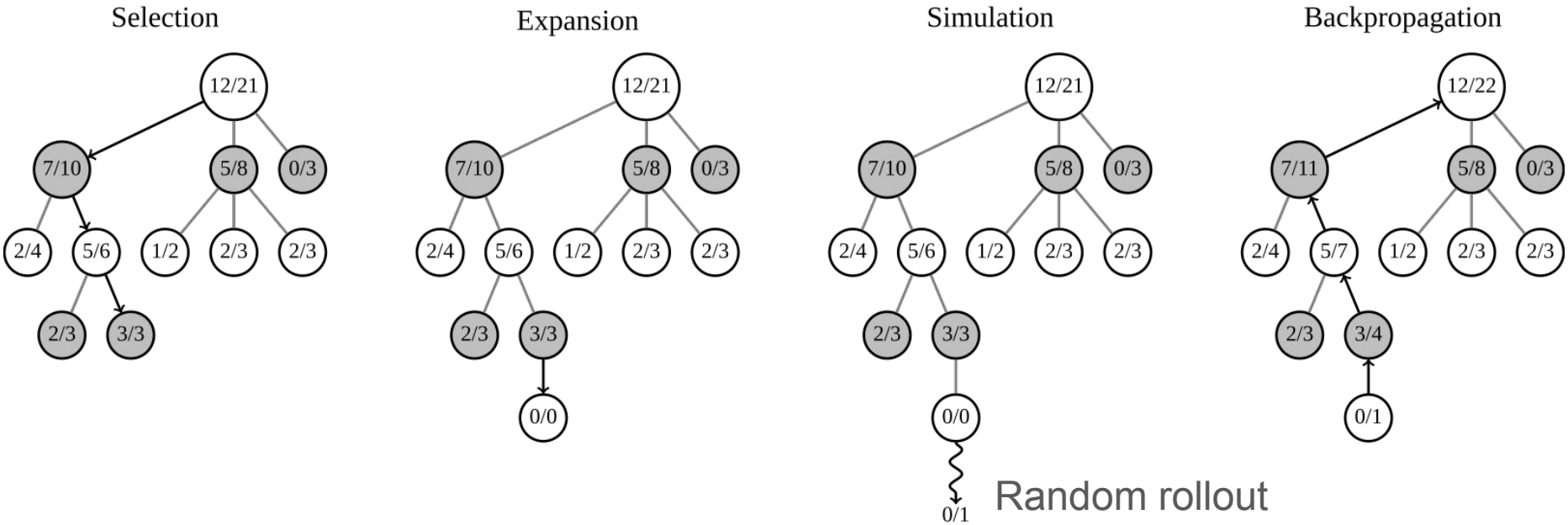
*Novikov, A., et al. (2025). “AlphaEvolve: A coding agent for scientific and algorithmic discovery” (arXiv:2506.13131).

Gemini 2.5 Pro results on CWM learning

Table 4: Perfect information games, refinement via tree search.

Game	OOD	transition accuracy			# LLM calls
		train	test	online	
Backgammon	✗	1.00000	0.99932	1.00000	16.8
Connect four	✗	1.00000	1.00000	1.00000	2.0
Tic-tac-toe	✗	1.00000	1.00000	1.00000	2.0
Gen. tic-tac-toe	✔	1.00000	1.00000	1.00000	2.4
Gen. chess	✔	1.00000	1.00000	1.00000	5.2

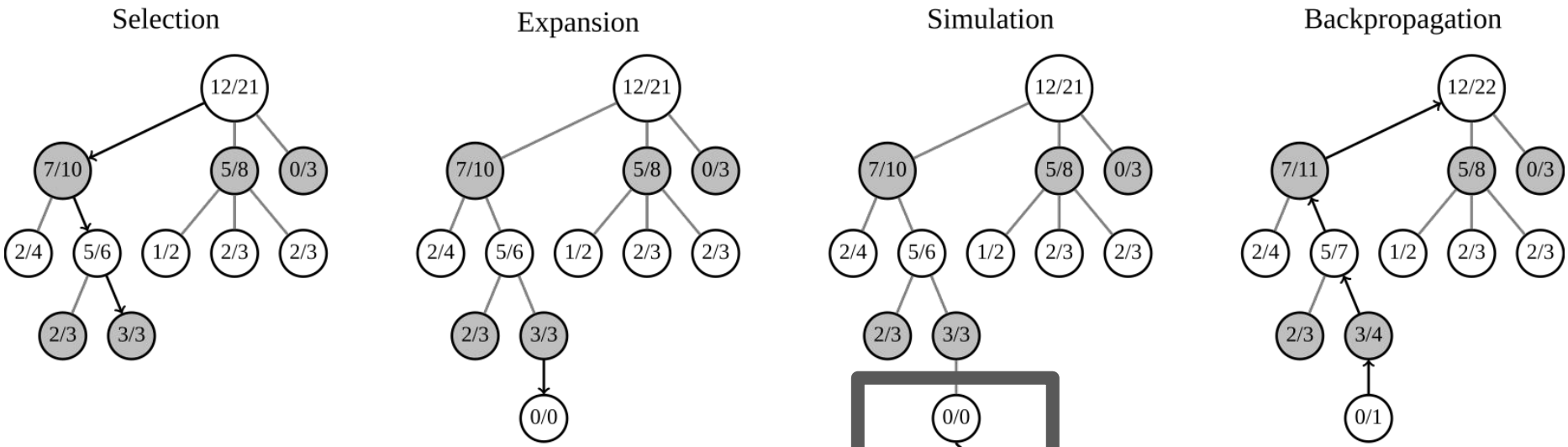
Planning with CWM: Monte Carlo Tree Search (MCTS)



- CWMs allow computationally efficient expansion of search space
- For rollouts, we use self-play, ie we assume the opponent uses the same policy as us (with roles flipped). So s' = F(s, (a1=pi(s), a2=pi'(s)).

Source: wikipedia

Speedup: learn code value fn for leaf nodes



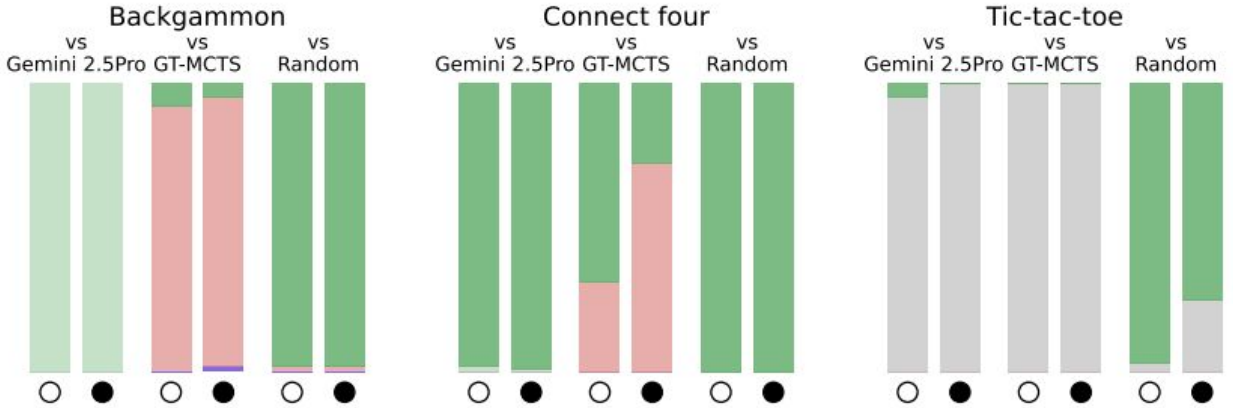
We prompt LLM to propose V.
Multiple functions are generated and the best one is selected through a tournament (since no GT to regress on)

Replace MC rollouts with value function V

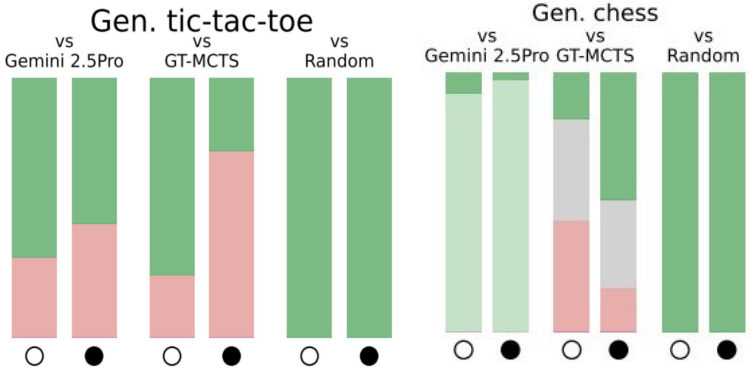
Source: wikipedia

Perfect info game results

In distribution games



Out of distribution games

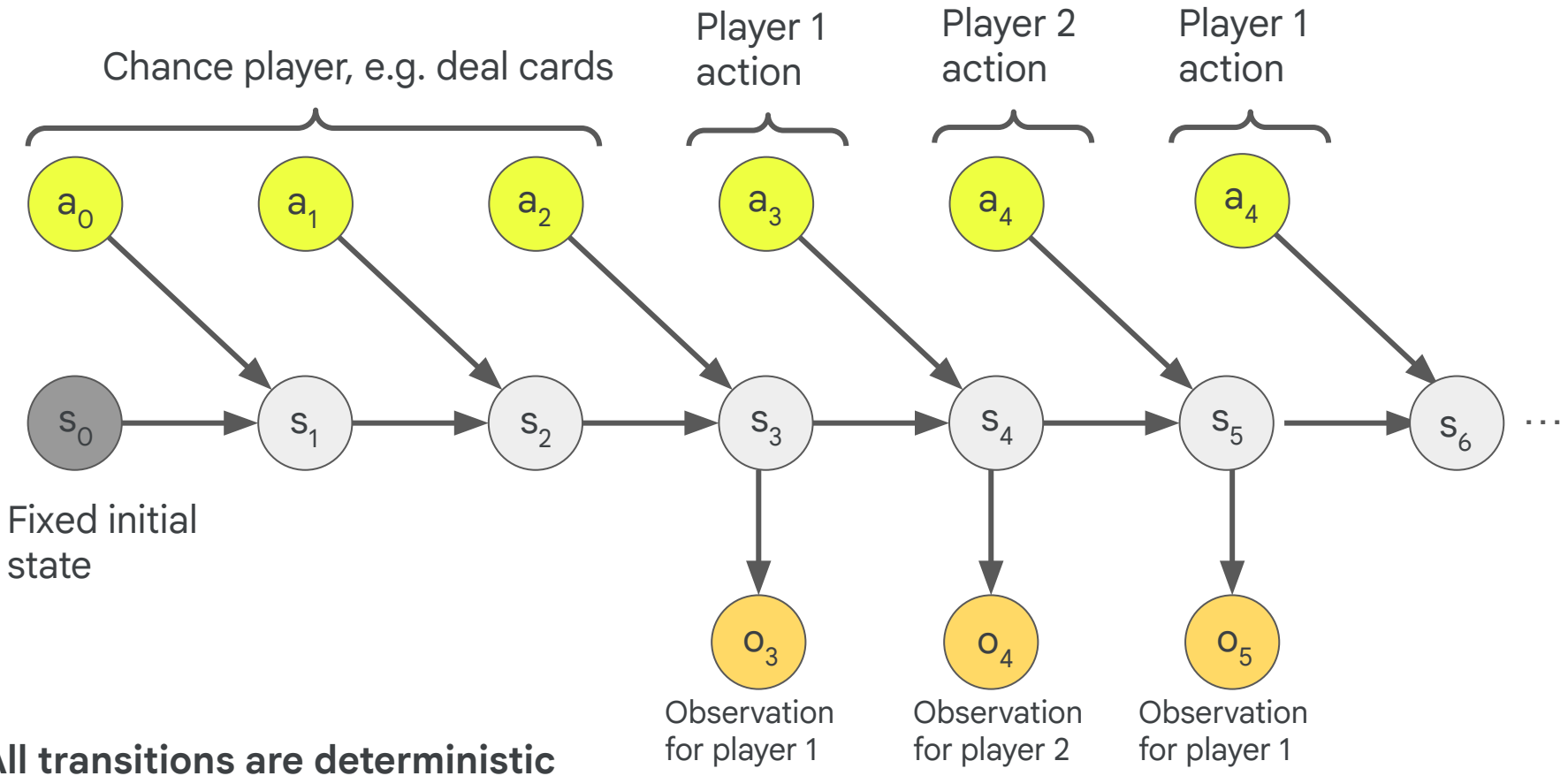


Opponents:

- Gemini (not cheating)
- GT: running MCTS on ground truth model (cheating)
- Random - picks valid actions (cheating)

Win | Win by Forfeit | Draw | Loss | Loss by Forfeit | ○ CWM-MCTS as Player 0 | ● CWM-MCTS as Player 1

Partially observed multi-player games



All transitions are deterministic

Partially observed multi-player games (eg Poker)

Scenario	Training	Testing
Fully observed (centralized)	$\tau = (s_{1:T}, \mathbf{a}_{1:T}^{1:N})$	$h_t = (s_{1:t}, \mathbf{a}_{1:t}^{1:N})$
Partially observed, open deck learning	$\tau^i = (o_{1:T}^i, a_{1:T}^i, s_{1:T})$	$h_t^i = (o_{1:t}^i, a_{1:t}^i)$
Partially observed, closed deck learning	$\tau^i = (o_{1:T}^i, a_{1:T}^i)$	$h_t^i = (o_{1:t}^i, a_{1:t}^i)$

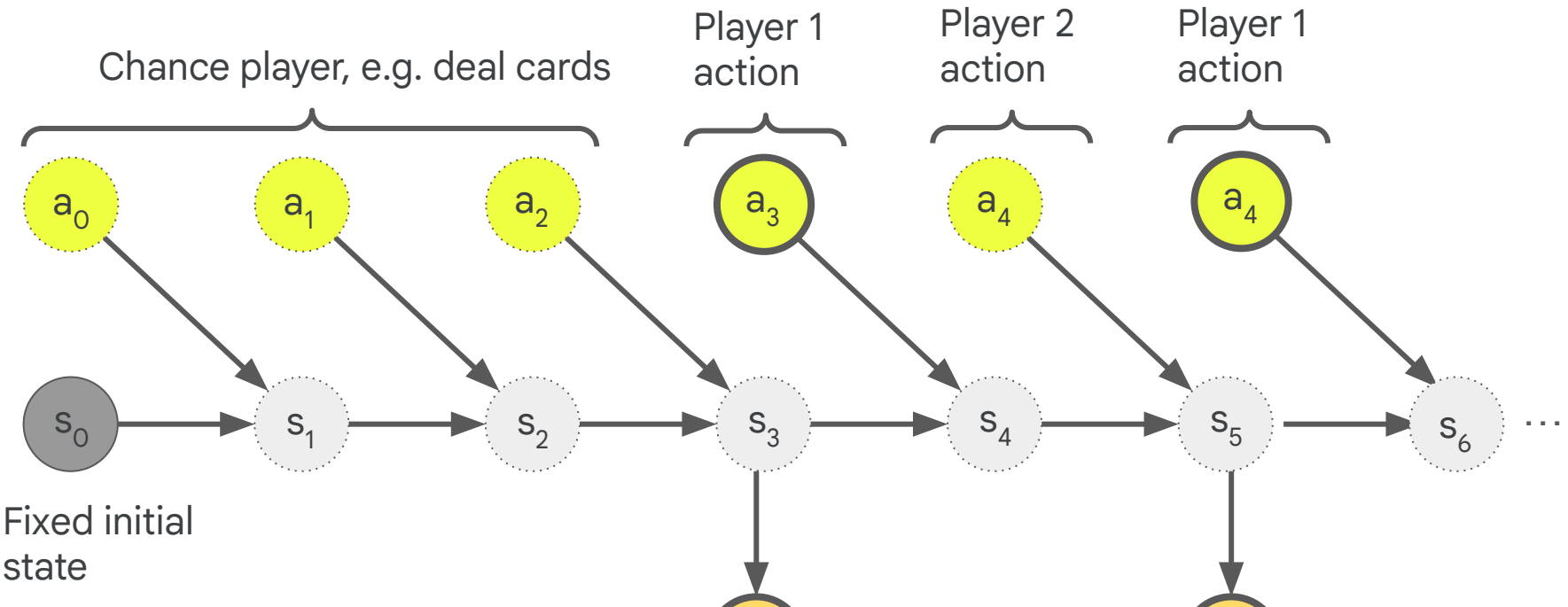
Need to learn world model and obs model - for open deck, same algo as before, but need different unit tests for obs model

$$
\begin{aligned}
s_{t+1} &= F(s_t, \mathbf{a}_t^{1:N}) \\
o_t^i &= M_i(s_t)
\end{aligned}
$$

Also need to learn inference function, to impute all hidden actions (including for chance player), from which all hidden states can be computed (causal world model)

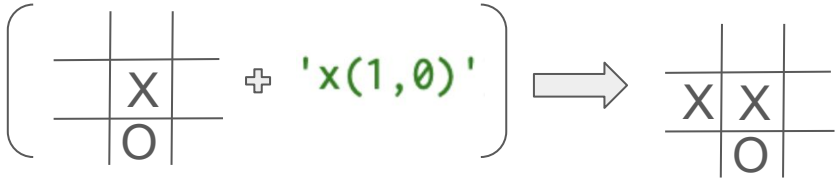
$$
\mathbf{a}_{1:t}^{1:N} = I_i(o_{1:t}^i, a_t^i)
$$

Given only player observations/actions, infer *all* actions



All transitions are deterministic

Reminder - Fully observed setting: unit tests for F



Algorithm 1: Test (fully observed)

Input: $(s_{1:T}, \mathbf{a}_{1:T}^{0:N}; F)$
for $t = 1 : T$ **do**
  assert$(s_{t+1} == F(s_t, \mathbf{a}_t))$

Textual feedback to LLM optimizer!

Unit tests (open deck: states are observed)

Replay resampled actions, check for consistency with observations per agent

Algorithm 2: Test open deck

Input: $(o^i_{1:T}, a^i_{1:T}, s_{1:T}; F, I, M)$
for $t = 1 : T$ **do**
  // Test world model using latent states
  assert$(s_{t+1} == F(s_t, a_t))$
  assert$(o^i_t == M_i(s_t))$
  // Test imputation model
  $\hat{\mathbf{a}}^{0:N}_{1:t} = I_i(o^i_{1:t}, a^i_{1:t})$
  $\hat{s}_0 = s_0$ // initial state is always observed
  for $k = 1 : t$ **do**
    $\hat{s}_k = F(\hat{s}_{k-1}, \hat{\mathbf{a}}^{0:N}_k)$ // unroll hypothetical hidden state sequence
  assert$(o^i_t == M_i(\hat{s}_t))$ // does hypothetical state match player's observed data?

Cf. "LLM-guided probabilistic program induction for POMDP model Estimation." Curtis et al, 2026

Synthesizing `CWM` and `I`

Gin rummy is just too hard

Game	OOD	transition accuracy			inference accuracy			# LLM calls
		train	test	online	train	test	online	
Bargaining	✗	1.0000	0.9827	1.0000	1.0000	1.0000	1.0000	23.0
Leduc poker	✗	1.0000	0.9977	0.9942	1.0000	1.0000	1.0000	4.4
Gin rummy	✗	0.7816	0.7455	0.9044	0.5857	0.5376	0.9678	500.0
Quadranto	✔	1.0000	1.0000	1.0000	1.0000	0.9864	0.9916	6.0
Hand of war	✔	1.0000	0.9814	0.9868	1.0000	0.9357	1.0000	144.0

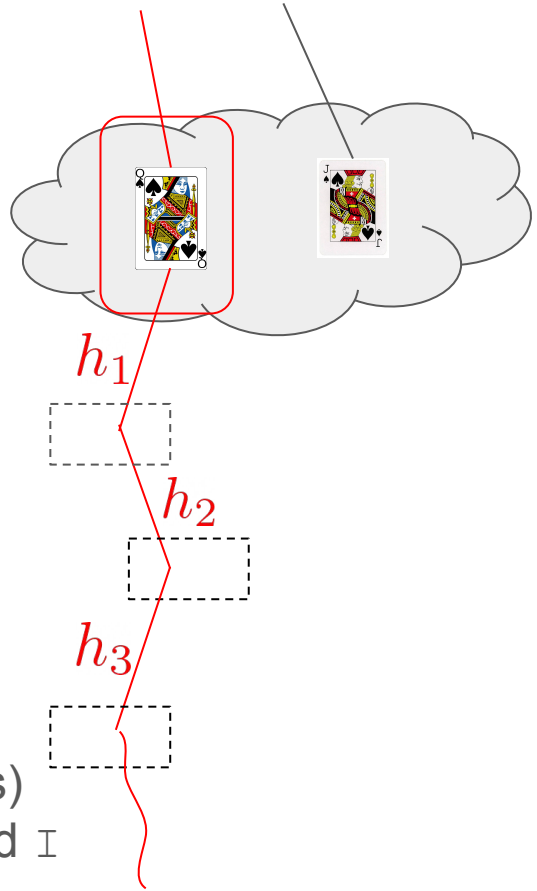
Overfitting starts to be an issue.

Planning with Information State MCTS (IS-MCTS)

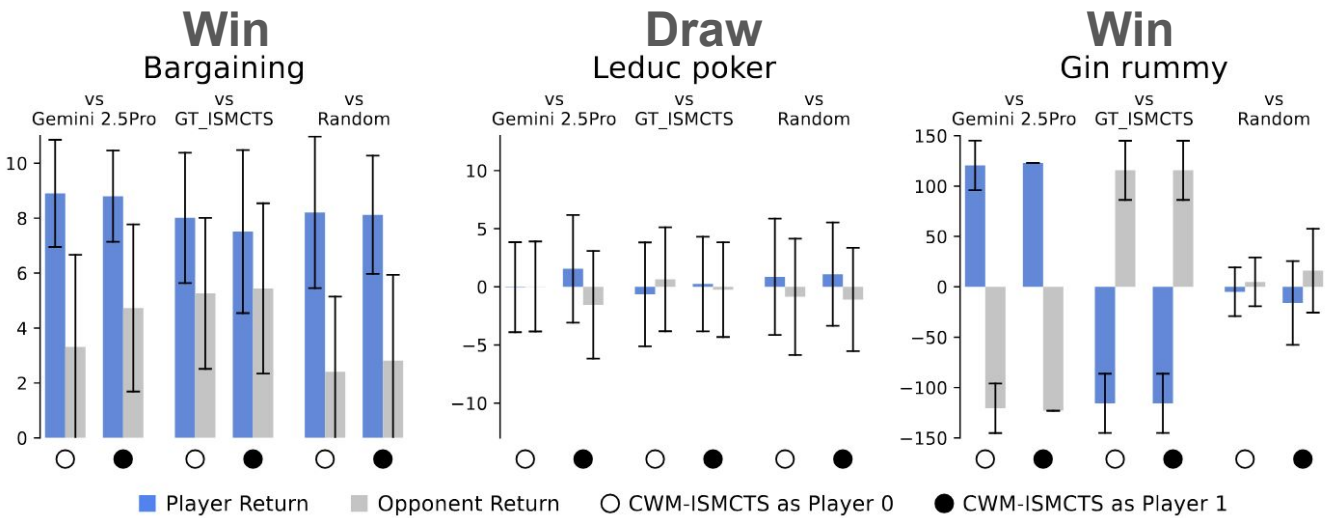
(Cowling et al. 2012)

D(s) is a distribution over ground states h in s

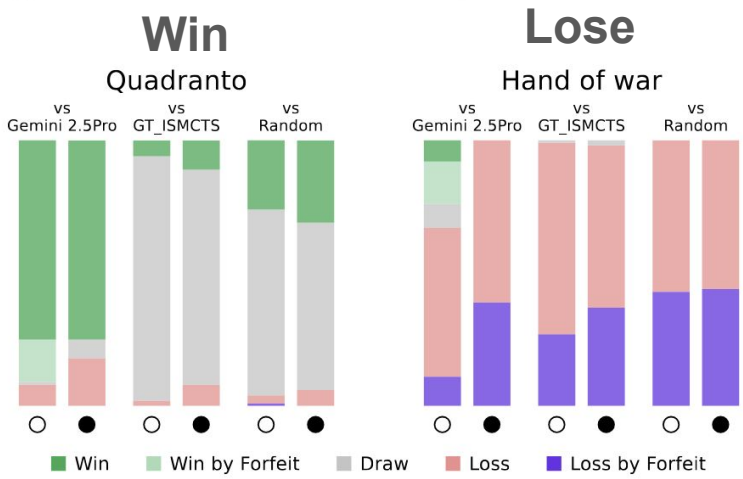
1. Sample a ~ `I`(obs_history), generate h0 = M(a)
2. TreePolicy:
   a. Simulate $h_t \rightarrow h_{t+1}$ until h_k is not in the tree (i.e. using UCB(s))
   b. Add h_k to the tree
3. Use random rollout or `V` value function



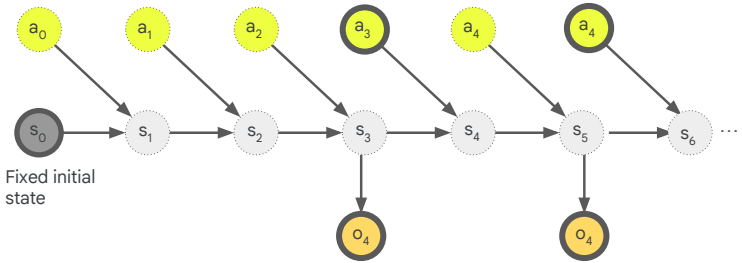
- Some seeds fail at inference (e.g., unshuffled cards)
  - => Do tournament between seeds to pick good `I`



Out of distribution



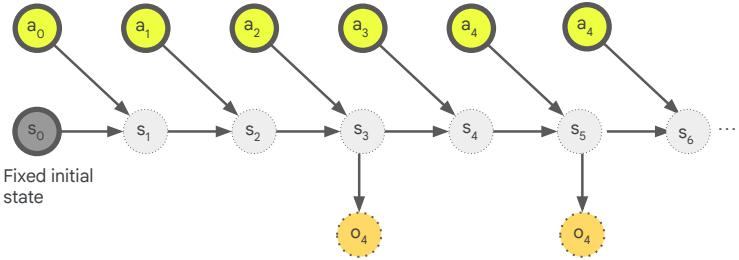
Learning imperfect information games, closed deck



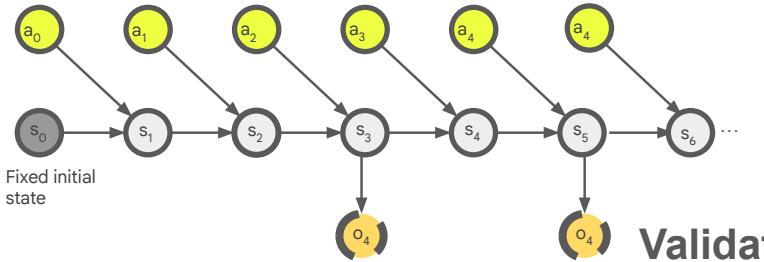
$$\tau = \{(\mathbf{o}_t^{1:N}, \mathbf{a}_t^{1:N}, \mathbf{o}_{t+1}^{1:N})\}$$

Cf autoencoder

Encode with I: Resample history of actions



Decode with M: Deterministically replay states given actions



Validate: Do we generate correct observations?

Unit tests (closed deck: states are hidden)

Replay resampled actions, check for consistency with observations per agent

Algorithm 3: Test closed deck

Input: $(o^i_{1:T}, a^i_{1:T}, \cancel{s_{1:T}}, F, I, M)$
for $t = 1 : T$ **do**
  // ~~Test world model using latent states~~
  assert($\cancel{s_{t+1} == F(s_t, a_t)}$)
  assert($\cancel{o^i_t == M_i(s_t)}$)
  // Test imputation model
  $\hat{\mathbf{a}}^{0:N}_{1:t} = I_i(o^i_{1:t}, a^i_{1:t})$
  $\hat{s}_0 = s_0$ // initial state is always observed
  for $k = 1 : t$ **do**
    $\hat{s}_k = F(\hat{s}_{k-1}, \hat{\mathbf{a}}^{0:N}_k)$ // unroll hypothetical hidden state sequence
  assert($o^i_t == M_i(\hat{s}_t)$) // does hypothetical state match player's observed data?

Deriving a log-likelihood lower bound on the model

- Encoder/Decoder are **under-specified**
- Consider a simple dice game with 4 sided dice.

CMW1: True model
Has dice with 4 sides

Actual game dynamics:

- Support size = 4
- Probability mass is concentrated
- Observations perfectly matched

CWM2: mis-specified model
Uses dice with 6 sides

Model assumptions:

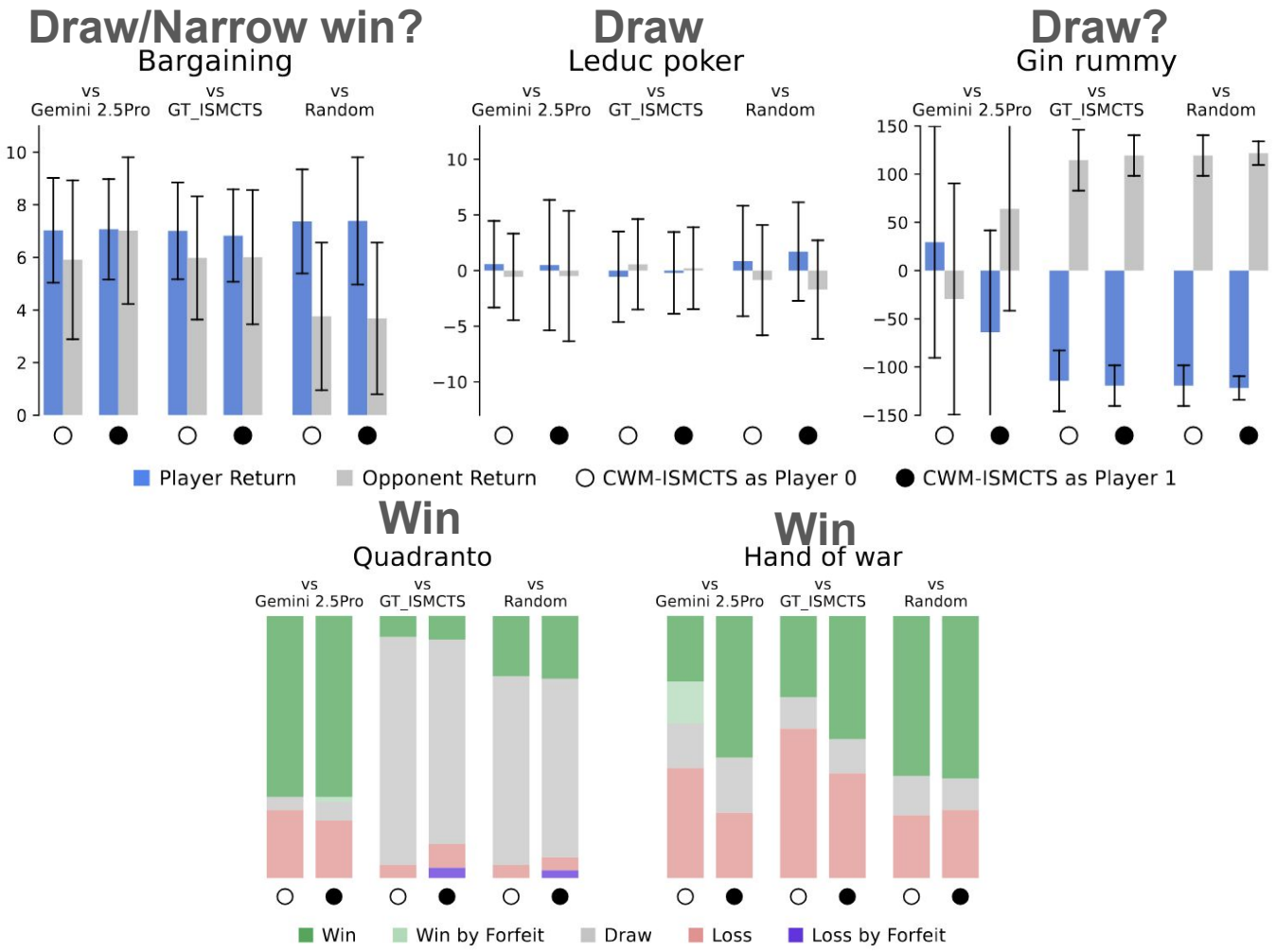
- **Support size = 6** (too many options)
- Probability mass "leaks" to impossible outcomes
- Lower bound is loose due to poor modelling of chance events

$$p_M(o^i_{1:t}) = \sum_{h_t} p_M(o^i_{1:t}|h_t)p_M(h_t) \leq p_M(o^i_{1:t}|\tilde{h}_t)p_M(\tilde{h}_t) = p_M(\tilde{h}_t)$$

Closed deck learning is a lot more challenging

Game	OOD	inference accuracy			# LLM calls
		train	test	online	
Bargaining	✗	1.00000	0.67359	0.76000	88.2
Leduc poker	✗	1.00000	0.97080	0.96585	9.0
Gin rummy	✗	0.05538	0.09523	0.53953	500.0
Quadranto	✔	1.00000	0.95183	0.96085	99.0
Hand of war	✔	0.86250	0.82130	0.94835	338.2

- Needs 100s of attempts to get code right
- Promising verifiable target for RLEF?


Draw/Narrow win?
Bargaining
Draw
Leduc poker
Draw?
Gin rummy
vs Gemini 2.5Pro
vs GT_ISMCTS
vs Random
Player Return
Opponent Return
CWM-ISMCTS as Player 0
CWM-ISMCTS as Player 1
Win
Quadranto
Win
Hand of war
Win
Win by Forfeit
Draw
Loss
Loss by Forfeit

Hard to learn world models for TextArena etc…

[Golf-v0]
[GAME] You are playing Golf (Card Game) - Player 0.
Goal: Get the lowest total score in this single round. If columns share the same value, they are summed as 0.
Card Values: A=1, 2-10=face value, J/Q=10, K=0

Actions (use exact brackets):
- '[draw]' - Draw from deck
- '[take]' - Take from discard pile
- '[swap X Y]' - Swap drawn card with position X (row) Y (column)
- '[discard]' - Discard the drawn card

[GAME] Your hand:
  Col: 1 2 3
Row 1: ? ? A♦
Row 2: ? ? 7♦
Discard pile: A❤
~~Your options: [draw], [take]~~

[KuhnPoker-v0]
[GAME] You are Player 1 in a 3 round game of Kuhn Poker.
Game Rules:
- Kuhn Poker uses a 3-card deck with J, Q, K (J lowest, K highest)
- Each player antes 1 chip and receives 1 card each round (note that the cards are dealt without replacement, so you cannot have the same card as your opponent).
- Game continues for 3 rounds
- The player with the most chips after all rounds wins

Action Rules:
- '[check]': Pass without betting (only if no bet is on the table)
- '[bet]': Add 1 chip to the pot (only if no bet is on the table)
- '[call]': Match an opponent's bet by adding 1 chip to the pot
- '[fold]': Surrender your hand and let your opponent win the pot

[GAME] ### Starting round 1 out of 3 rounds. Your card is: 'K'
~~[GAME] Your available actions are: '[check]', '[bet]'~~

[Chess-v0]
[GAME] You are playing White in a game of Chess.
Make your moves in UCI format enclosed in square brackets (e.g., [e2e4]).
[GAME] Current board:

```
  +-----------------+
8 | r n b q k b n r |
7 | p p p p p p p p |
6 | . . . . . . . . |
5 | . . . . . . . . |
4 | . . . . . . . . |
3 | . . . . . . . . |
2 | P P P P P P P P |
1 | R N B Q K B N R |
  +-----------------+
    a b c d e f g h
```

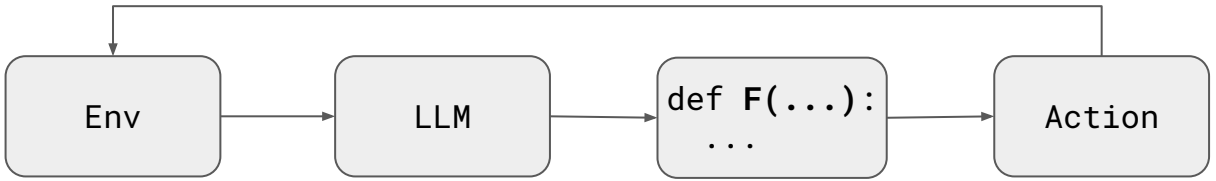
[LinesOfAction-v0]
[GAME] You are Player 0 in game of LinesOfAction.
Your pieces are 'O', opponent pieces are 'X'.
Move format: `[b1b3]` (from-coord to-coord). A legal move travels horizontally, vertically, or diagonally a number of squares equal to the total pieces (any colour) in that line. You may jump over your own pieces, but not opponent pieces; landing on an opponent captures it. Win when all your pieces are 8-neighbour connected.
[GAME] Board:

```
    a   b   c   d   e   f   g   h
  +---+---+---+---+---+---+---+---+
8 |   | O | O | O | O | O | O |   | 8
  +---+---+---+---+---+---+---+---+
7 | X |   |   |   |   |   |   | X | 7
  +---+---+---+---+---+---+---+---+
6 | X |   |   |   |   |   |   | X | 6
  +---+---+---+---+---+---+---+---+
5 | X |   |   |   |   |   |   | X | 5
  +---+---+---+---+---+---+---+---+
4 | X |   |   |   |   |   |   | X | 4
  +---+---+---+---+---+---+---+---+
3 | X |   |   |   |   |   |   | X | 3
  +---+---+---+---+---+---+---+---+
2 | X |   |   |   |   |   |   | X | 2
  +---+---+---+---+---+---+---+---+
1 |   | O | O | O | O | O | O |   | 1
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
```

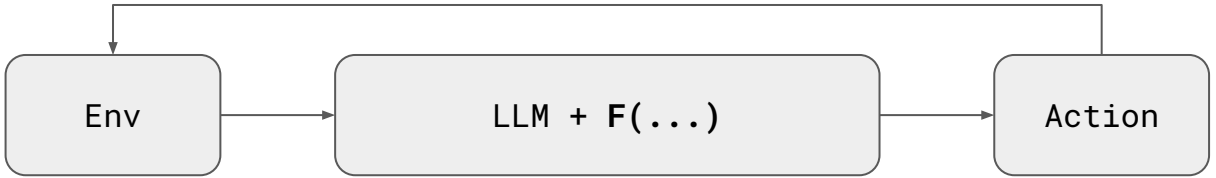
Solution: Code-as-harness

- Why not use LLM to learn code-as-world-model?
  - It is hard to learn reliable world models for complex environments such as text games
  - MCTS on a world model can be slow
- Why not use LLM-as-policy?
  - Low quality actions - sometimes selects illegal actions! (Immediately lose in a multi-player game setting.)
- Why not use LLM to learn code-as-policy?
  - Hard to do strategic reasoning in multplayer games using fixed code policy (need test-time inference/search)
- Solution: code as harness
  - Use LLM as base policy
  - Learn partial world model to filter out bad action proposals

Code-as-harness (Online training)



Code-as-harness (Eval)



Why need LLM harness?

- LLMs play illegal moves even on chess!
- GameArena Chess competition: **78% of 2.5-Flash's losses were due to illegal move**
- 2.5-Pro still makes illegal moves even with dedicated Chess pre-training data in the mixture

LLM	Total	Win	Loss	Illegal_Move	Win_Rate	Illegal_Move_Rate	Loss_by_Illegal_Move_Rate
Grok 4	360	290	35	0.0	0.805556	0.000000	0.000000
o3	360	329	5	0.0	0.913889	0.000000	0.000000
o4-mini	360	229	104	4.0	0.636111	0.011111	0.038462
Gemini 2.5 Pro	360	267	58	8.0	0.741667	0.022222	0.137931
Claude Sonnet 4	360	131	208	15.0	0.363889	0.041667	0.072115
Claude Opus 4	360	115	215	65.0	0.319444	0.180556	0.302326
DeepSeek R1	360	112	214	78.0	0.311111	0.216667	0.364486
GPT 4.1	360	141	197	122.0	0.391667	0.338889	0.619289
Gemini 2.5 Flash	360	37	316	245.0	0.102778	0.680556	0.775316
Kimi K2	360	30	329	298.0	0.083333	0.827778	0.905775

Ref: https://www.kaggle.com/datasets/kaggle/chess-text-gameplay

Update 24 August 2025: Chess vs Gemini 2.5 Pro Thinking

Out of pure interest, I decided to attempt another game against Gemini 2.5, with the full game available here (Google Internal Conversation). Playing as Black, Gemini predictably chose the most popular Ruy Lopez in response to 1. e4 e5, driving the position into my favourite opening, the Berlin Wall. I did not make the most optimal moves, deviating from established theory on move 13 with Rd8?! instead of the most prudent Rh6. Gemini correctly capitalised on my mistake, weakening my pawn structure after 16. Nxe6 fxe6, in line with established concepts in the Berlin Wall. However, after 17… h4 we once again reached the end of theoretical knowledge, leading to an almost immediate piece blunder with 20. Nd6+, followed by an attempted illegal move with 24. dxe5.

This time, I asked for Gemini's reflection on the game, with particular interest in whether it found it to be more difficult to keep track of the position's state, or more difficult to reason about the best move in a position. I also provided a full PGN of the game, as well as the FEN for the final position. Reflecting on the game, Gemini said 'This game perfectly illustrates the challenges of playing in a text-only format. For me, keeping **track of the position state** was by far the most challenging aspect.'

Ref: Chess as a Benchmark for AGI

Code-as-harness: What?

Algorithm 6: Code as verifier (harness)

```
def policy(h_t, F, LLM):
k = 0 // Trial number
a_t^0 = ∅ // Initial action
v_t^0 = False // Is trial valid?
while not v_t^k and k < maxTries do
    a_t^{k+1} = LLM(prompt="propose action", h_t, a_t^{0:k}, v_t^{0:k})
    v_t^{k+1} = F(h_t, a_t^{k+1})
    k = k + 1
if v_t^k then
    return a_t^k
else
    return fail
```

Feedback on previous failed attempts

Check if proposed move is valid before trying

F is a "**partial world model**":
predict validity bit, not entire next state

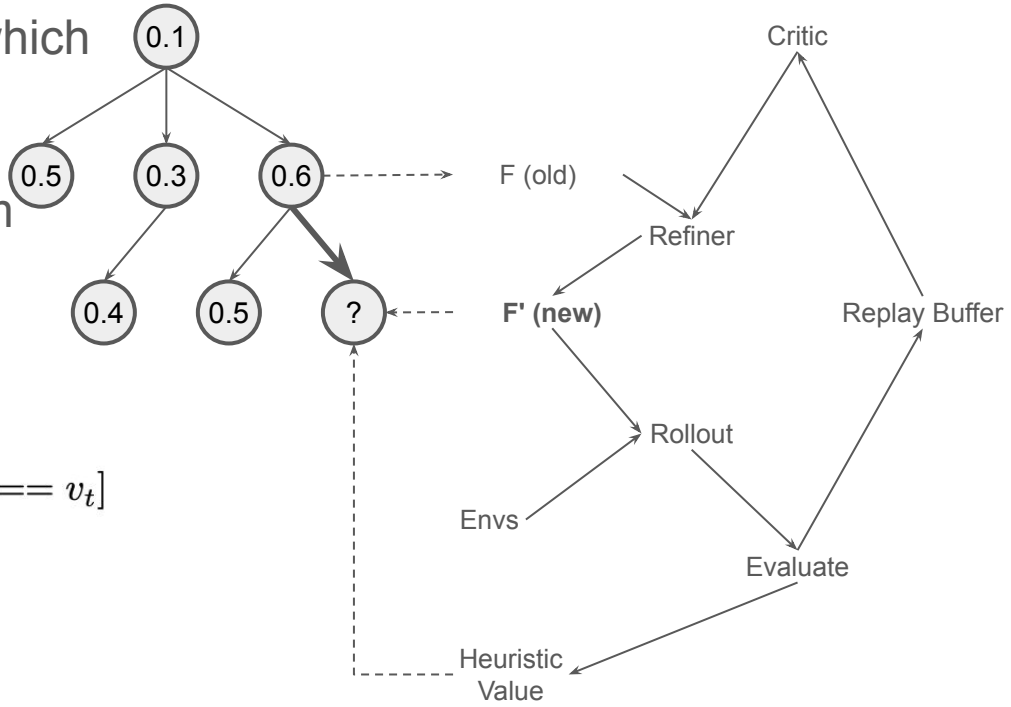
Code-as-harness: how? CodeProp once again

-Tree search over python functions, using LLM as proposal given **online** trajectories

-Use Thompson sampling to decide which node to expand at each step

- cf "REx"

-Use immediate validity feedback from env. to learn F (no delayed reward)



$$\hat{F} = \arg\max_F \text{score}(F)$$

$$\text{score}(F) = \sum_i \sum_{(s_t, \mathbf{a}_t, v_t, s_{t+1}) \in \text{traj}_i} \mathbf{I}[F(s_t, \mathbf{a}_t) == v_t]$$

Benchmark: TextArena++

Why TextArena?

- Large amount of games: 100+ games in total
- High diversity of games
  - Single/multi-players
  - Poker, board, number, probability, etc.
- Complex and heterogeneous action formats
- Extremely long context
- Output LLM-ready text format

Game selection and modification

- Only consider 1-player and 2-player games
- Skip games with free text negotiation or LLM as judge
- **++: Remove any hints of available moves**

[KuhnPoker-v0]

```
[GAME] You are Player 1 in a 3 round game of Kuhn Poker.
Game Rules:
- Kuhn Poker uses a 3-card deck with J, Q, K (J lowest, K highest)
- Each player antes 1 chip and receives 1 card each round (note
that the cards are dealt without replacement, so you cannot have
the same card as your opponent).
- Game continues for 3 rounds
- The player with the most chips after all rounds wins

Action Rules:
- '[check]': Pass without betting (only if no bet is on the table)
- '[bet]': Add 1 chip to the pot (only if no bet is on the table)
- '[call]': Match an opponent's bet by adding 1 chip to the pot
- '[fold]': Surrender your hand and let your opponent win the pot

[GAME] ### Starting round 1 out of 3 rounds. Your card is: 'K'
[GAME] Your available actions are: '[check]', '[bet]'
```

[Golf-v0]

```
[GAME] You are playing Golf (Card Game) - Player 0.
Goal: Get the lowest total score in this single round. If
columns share the same value, they are summed as 0.
Card Values: A=1, 2-10=face value, J/Q=10, K=0

Actions (use exact brackets):
- '[draw]' - Draw from deck
- '[take]' - Take from discard pile
- '[swap X Y]' - Swap drawn card with position X (row) Y
(column)
- '[discard]' - Discard the drawn card

[GAME] Your hand:
  Col:   1    2    3
Row 1:   ?    ?    A♦
Row 2:   ?    ?    7♦
Discard pile: A♥
Your options: [draw], [take]
```

[Chess-v0]

```
[GAME] You are playing White in a
game of Chess.
 Make your moves in UCI format
enclosed in square brackets (e.g.,
[e2e4]).
[GAME] Current board:
   +-----------------+
 8 | r n b q k b n r |
 7 | p p p p p p p p |
 6 | . . . . . . . . |
 5 | . . . . . . . . |
 4 | . . . . . . . . |
 3 | . . . . . . . . |
 2 | P P P P P P P P |
 1 | R N B Q K B N R |
   +-----------------+
     a b c d e f g h
```

[LinesOfAction-v0]

```
[GAME] You are Player 0 in game of LinesOfAction.
Your pieces are 'O', opponent pieces are 'X'.
Move format: `[b1b3]` (from-coord to-coord). A legal move travels horizontally,
vertically, or diagonally a number of squares equal to the total pieces (any colour) in
that line. You may jump over your own pieces, but not opponent pieces; landing on an
opponent captures it.  Win when all your pieces are 8-neighbour connected.
[GAME] Board:

    a   b   c   d   e   f   g   h
  +---+---+---+---+---+---+---+---+
8 |   | O | O | O | O | O | O |   | 8
  +---+---+---+---+---+---+---+---+
7 | X |   |   |   |   |   |   | X | 7
  +---+---+---+---+---+---+---+---+
6 | X |   |   |   |   |   |   | X | 6
  +---+---+---+---+---+---+---+---+
5 | X |   |   |   |   |   |   | X | 5
  +---+---+---+---+---+---+---+---+
4 | X |   |   |   |   |   |   | X | 4
  +---+---+---+---+---+---+---+---+
3 | X |   |   |   |   |   |   | X | 3
  +---+---+---+---+---+---+---+---+
2 | X |   |   |   |   |   |   | X | 2
  +---+---+---+---+---+---+---+---+
1 |   | O | O | O | O | O | O |   | 1
  +---+---+---+---+---+---+---+---+
    a   b   c   d   e   f   g   h
```

Main Results:
Making legal moves lead to improved **TextArena 2P/1P game plays**

Agents in evaluation

LLM-as-policy:

- Gemini-2.5-Flash
- Gemini-2.5-Pro
- Gemini-2.5-Flash + harness (Ours)

Results

- Learned harness achieved **100% legal move accuracy on all 145 2P/1P games**.
  - Evaluated on 10,000 steps for 10 envs (random seeds) per game.

env_id	Gemini-2.5-Flash	Ours
Stratego-v0	76.12%	100.00%
LinesOfAction-v0	76.16%	100.00%
SantoriniBaseFixed-v0	80.52%	100.00%
Checkers-v0	83.40%	100.00%
Chess-v0	83.98%	100.00%
Alquerque-v0	84.08%	100.00%
Othello-v0	87.68%	100.00%
Breakthrough-v0	89.52%	100.00%
LiarsDice-v0	90.56%	100.00%
Tak-v0	92.04%	100.00%
NewRecruit-v0	92.44%	100.00%
Crusade-v0	94.48%	100.00%
Chopsticks-v0	94.48%	100.00%
SpiteAndMalice-v0	94.92%	100.00%
Golf-v0	95.56%	100.00%
SimpleBlindAuction-v0	96.60%	100.00%
QuantumTicTacToe-v0	96.64%	100.00%
UltimateTicTacToe-v0	96.64%	100.00%
GermanWhist-v0	97.28%	100.00%
Poker-v0	97.48%	100.00%
Battleship-v0	97.96%	100.00%
KuhnPoker-v0	98.52%	100.00%

TextArena 2P Games with <99% legal action accuracy (Gemini-2.5-Flash)

[TextArena 2P Games]
Won 9/16 games Vs Gemini-2.5-Pro
Won 12/16 games Vs Gemini-2.5-Flash

Arena settings:

- 40 matches per game (split even as 1st and 2nd player)
- Selected 16 relatively long games (e.g. no Tic-Tac-Toe)
- Reward: 1.0 if win, -1.0 if lose, 0.0 if draw

gemini-2.5-flash+harness Vs gemini-2.5-pro	
Opponent Reward	-0.181
Agent Reward	0.181
Draws	0.055
Opponent Wins	0.382
Agent Wins	0.563
Opponent Valid Action Accuracy	0.867
Agent Valid Action Accuracy	0.941
Valid Action Accuracy	0.892

gemini-2.5-flash+harness Vs gemini-2.5-flash	
Opponent Reward	-0.342
Agent Reward	0.342
Draws	0.045
Opponent Wins	0.306
Agent Wins	0.648
Opponent Valid Action Accuracy	0.793
Agent Valid Action Accuracy	0.950
Valid Action Accuracy	0.858

gemini-2.5-flash+harness Vs. gemini-2.5-pro	Losses	Draws	Wins
Alquerque-v0	0.400	0.000	**0.600**
Breakthrough-v0	0.050	0.000	**0.950**
Checkers-v0	0.000	0.000	**1.000**
Chess-v0	0.200	0.000	**0.800**
Chopsticks-v0	**0.500**	0.225	0.275
Crusade-v0	0.325	0.100	**0.575**
GermanWhist-v0	**0.600**	0.125	0.275
Golf-v0	**0.525**	0.000	0.475
LiarsDice-v0	0.450	0.000	**0.550**
LinesOfAction-v0	0.225	0.000	**0.775**
NewRecruit-v0	**0.500**	0.150	0.350
Othello-v0	**0.800**	0.100	0.100
SpiteAndMalice-v0	**0.561**	0.000	0.439
Stratego-v0	0.025	0.000	**0.975**
Tak-v0	0.300	0.000	**0.700**
UltimateTicTacToe-v0	**0.650**	0.175	0.175

XIDs: 221050838, 220973835, 221065003, 221016458

[TextArena 1P Games]
Harnessed Gemini-2.5-Flash Outperformed Gemini-2.5-Pro and Gemini-2.5-Flash

Agents in arena:

- Gemini-2.5-Flash
- Gemini-2.5-Pro
- Gemini-2.5-Flash+Harness

Arena settings:

- 20 runs per game
- Reward: [-1.0, 1.0]

Player0 Reward (all)

	gemini-2.5-flash	gemini-2.5-pro	gemini-2.5-flash+harness (ours)
TextArena (1P)	0.673	0.707	**0.745**

Legal Action Accuracy (all)

	gemini-2.5-flash	gemini-2.5-pro	gemini-2.5-flash+harness (ours)
TextArena (1P)	94.05%	96.15%	**99.37%**

Player0 Reward (per game)

	gemini-2.5-flash	gemini-2.5-pro	gemini-2.5-flash+harness (ours)
2048-v0	0.215	**0.378**	0.313
Bandit-v0	**0.398**	0.201	0.208
Blackjack-v0	0.410	0.330	**0.480**
Cryptarithm-v0	**1.000**	0.950	**1.000**
FifteenPuzzle-v0	0.107	0.103	**0.162**
FrozenLake-v0	**1.000**	**1.000**	**1.000**
GuessTheNumber-v0	**1.000**	**1.000**	**1.000**
LightsOut-v0	0.730	0.802	**0.840**
Mastermind-v0	**1.000**	**1.000**	**1.000**
Minesweeper-v0	0.637	0.586	**0.686**
PegJump-v0	0.325	0.682	**0.782**
RushHour-v0	0.688	0.887	**1.000**
Secretary-v0	0.550	**0.700**	0.650
Sokoban-v0	0.700	0.700	**0.800**
Sudoku-v0	**1.000**	**1.000**	**1.000**
TowerOfHanoi-v0	**1.000**	**1.000**	**1.000**

Special highlight: Outperform GPT-5.2-High in TextArena 1P Games

Player0 Reward (all)

	gemini-2.5-flash	gemini-2.5-pro	gemini-2.5-flash+harness (ours)	gpt-5.2	gpt-5.2-high	gemini-3-flash	harness-as-policy (ours)
TextArena (1P)	0.673	0.707	0.745	0.635	0.844	0.833	**0.870**

Legal Action Accuracy (all)

	gemini-2.5-flash	gemini-2.5-pro	gemini-2.5-flash+harness (ours)	gpt-5.2	gpt-5.2-high	gemini-3-flash	harness-as-policy (ours)
TextArena (1P)	94.05%	96.15%	99.37%	93.87%	98.61%	99.84%	**100.00%**

Agents in evaluation

LLM-as-policy:

- Gemini-2.5-Flash
- Gemini-2.5-Pro
- GPT-5.2 (no thinking)
- GPT-5.2-High (high thinking)

Ours:

- Gemini-2.5-Flash distilled harness

Results

- **Beat GPT-5.2-High in average reward: 0.870 Vs 0.844**
- **Beat Gemini-3-Flash in average reward: 0.870 Vs 0.833**
- Test time cost:
  - **~$0.0 (ours) Vs $640** for GPT-5.2 + GPT-5.2-High

Player0 Reward (per game)

	gemini-2.5-flash	gemini-2.5-pro	gpt-5.2	gpt-5.2-high	gemini-2.5-flash+harness (ours)
2048-v0	0.215	0.378	0.212	0.745	**0.912**
Bandit-v0	0.398	0.201	0.350	**1.000**	0.459
Blackjack-v0	0.410	0.330	0.460	**0.480**	0.410
Cryptarithm-v0	**1.000**	0.950	0.600	**1.000**	**1.000**
FifteenPuzzle-v0	0.107	0.103	0.035	0.183	**0.597**
FrozenLake-v0	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
GuessTheNumber-v0	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
LightsOut-v0	0.730	0.802	0.691	**1.000**	**1.000**
Mastermind-v0	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Minesweeper-v0	0.637	0.586	0.593	**1.000**	0.940
PegJump-v0	0.325	0.682	0.221	0.429	**1.000**
RushHour-v0	0.688	0.887	**1.000**	**1.000**	**1.000**
Secretary-v0	0.550	0.700	0.600	**0.800**	0.750
Sokoban-v0	0.700	0.700	0.600	**0.867**	0.850
Sudoku-v0	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
TowerOfHanoi-v0	**1.000**	**1.000**	0.800	**1.000**	**1.000**

XIDs: 213329356, 213314820, 214313487, 214542593, 214546192, 214499807

Chess demo

Game Board

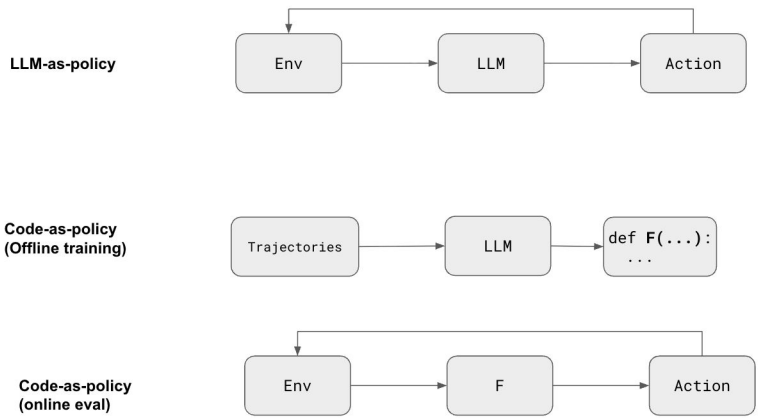
Harness - 83.8% Success

[AutoHarness]

Environment: Chess-v0
Illegal action: No
Current status: Monitoring actions

Summary

Standard approach



Our approach