LeAct: Learning to Reason from Expert Actions

Ziran Yang* **Chengshuai Shi** **Raj Ghugare**
Benjamin Eysenbach **Karthik Narasimhan** **Chi Jin**
Princeton University

Abstract

Modern reasoning models depend on reasoning data, today sourced from human annotations or distilled from stronger LLMs. However, a rich and largely untapped source of supervision lies in expert systems (e.g., game engines, classical planners, theorem provers), which routinely produce near-optimal actions across diverse domains. But these experts are silent: they commit to an action without writing down the chain of thought (CoT) behind it. Recovering that CoT as natural-language reasoning would distill expert knowledge into a student that generalizes beyond the demonstrated actions. We treat it as a latent variable and study how to recover it from the action alone. Our approach, LeAct (**Le**arning to reason from **Act**ions), optimizes this latent variable: the student samples candidate CoTs for each expert action, and we retain those that measurably improve its own probability of recovering the action. Across imperfect-information games at multiple scales and a simulated robotics benchmark, LeAct reaches the solver's numerical floor on small enumerable games. At larger scale, it is $5\times$ closer to the solver than the strongest expert-iteration baseline. At Flop Hold'em ($\sim 10^9$ infosets), LeAct wins head-to-head by $+60$ mbb/g, and on the in-domain robotics probe it is the only evaluated recipe that improves best-of-64 cube-placement progress over the joint coldstart. We present a principled framework and the result: expert systems become a categorically new source of reasoning teachers for foundation models.

1 Introduction

Building chain-of-thought (CoT) data is now a crucial bottleneck for reasoning language models [32, 10]. There is no shortage of work trying to scale beyond human annotation by automatically labeling CoT, either through distillation from stronger LLMs [13, 29, 10] or by bootstrapping a model's own outputs [56, 9, 36]. But these methods rarely close the loop directly on the CoT itself. Filtering on synthetic CoT typically targets answer correctness or surface plausibility. But whether the CoT actually contributes to the student's prediction remains uncertain.

We draw from a different source: silent action experts (game solvers [60, 44, 5], classical planners, robotic planners [45], theorem provers [25]). Each commits to a near-optimal action at every state, but neither the action nor the reasoning behind it is expressed in natural language. The naive approach, asking a student LLM to write a CoT for each expert action, runs into rationalization [49, 21]: the CoT can be plausible while bearing no causal link to the action it accompanies. We close the loop instead. For every state-action pair, the student samples multiple candidate CoTs, and we retain only those that raise the student's own probability of recovering the expert action. The filter is the load-bearing step: replacing it with random ranking erases the advantage in our ablations.

LeAct (**Le**arning to reason from **Act**ions) realizes this idea as an iterative training pipeline (Fig. 1). The conventional *reason-then-act* paradigm [54] takes reasoning as input and produces actions at inference time; LeAct runs that loop in reverse at training time, conditioning on the expert action,

*Correspondence: zirany@princeton.edu

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

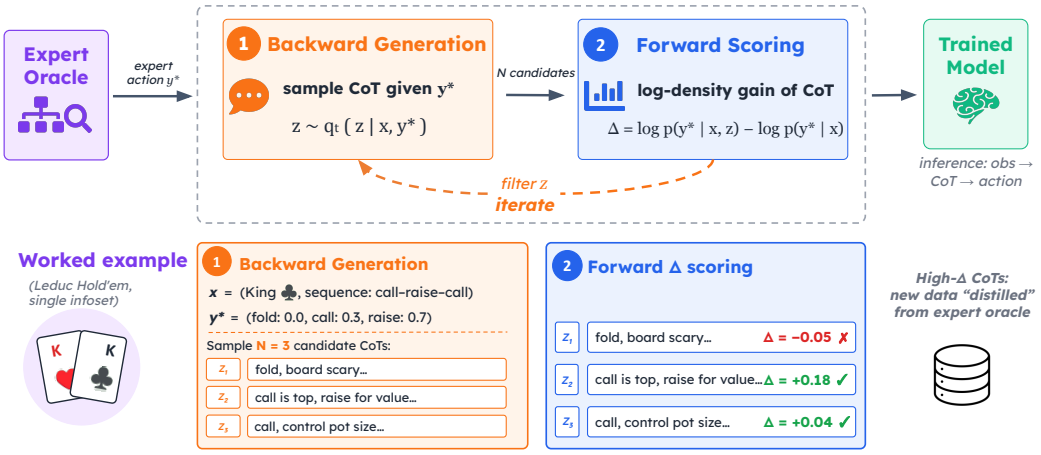


Figure 1: `LeAct` **turns silent action experts into reasoning teachers.** For each state x, the student samples candidate explanations conditioned on the oracle's action $\pi^*(\cdot \mid x)$. We keep those that raise the student's predicted probability of the action and use them as CoT to fine-tune the student.

sampling reasoning that recovers it, and training on the reasoning that survived the filter. The recipe accepts any action-only oracle: a CFR solver, a deep RL policy, or a frontier LLM committed to a single demonstration trajectory. We test all three families: `CFR` solvers on enumerable poker settings like Leduc Hold'em [43]; a `DeepCFR` [5] policy on Flop Hold'em ($\sim 10^9$ infosets); and frontier-LLM successful trajectories on a simulated robotics benchmark [8].

Contributions. We make three: a new non-LLM source of CoT supervision, a latent-variable derivation of the algorithm, and empirical results on various games and a robotics benchmark.

- **A non-LLM data source for CoT supervision.** Rather than mining intermediate text [36, 56], `LeAct` converts a structured action policy $\pi^*(\cdot \mid x)$ from any non-LLM oracle into natural-language reasoning data, a categorically distinct supervision modality from prior CoT pipelines.
- **A principled algorithm** (§3). We treat the CoT as a latent variable connecting state X to expert action $Y^* \sim \pi^*$, and derive a learning procedure from this view. The practical procedure ranks candidate CoTs by a proxy for their contribution to the student's decision-making and uses π^* as the SFT target, motivated by the ideal action-space objective. `LeAct` sits in the recent line of latent-EM self-improvement [36], but anchors the latent to an external expert action rather than to text the model itself produced.
- **Empirical results** (§5–6). We test `LeAct` on imperfect-information games [20] at multiple scales and on a simulated robotics cube-stacking benchmark [8]. A student trained with `LeAct` matches the expert solver on small games, beats both behavior-cloning and expert-iteration baselines at the large games we test, and is the only evaluated recipe that improves best-of-64 progress over the joint coldstart on the in-domain robotics tasks. The advantage holds on unseen states, and we trace it to the explanation-selection step.

2 Related Work

The closed-loop feedback gap in CoT supervision. There is no shortage of CoT-supervision work: humans label rationales by hand [53, 16], stronger LLMs distill them into smaller students [13, 29, 10], and self-improvement loops mine latent thoughts from text the model itself produces [56, 14, 57, 9, 41, 36]. What none of these provide is a closed-loop feedback signal that decides *which* CoTs are worth training on by checking them against an external authority. Existing pipelines either skip the filter entirely (annotation, distillation), filter by binary correctness against a single verifiable answer [56, 9, 41], or filter by the model's own likelihood [36, 6]. In every case the supervision signal is either absent or LLM-internal. `LeAct` fills this gap: each candidate CoT survives only if it

measurably increases the student's own probability of recovering an external oracle's action, closing the loop with supervision that originates outside the LLM family.

External verifiable feedback for training reasoning. A second line of work uses an external oracle or verifier as the training signal, instead of (or alongside) text the LLM itself produced. Pairing a generator with an external checker is a long-running training recipe across game outcomes, formal verifiers, and theorem provers [40, 28, 39, 47, 15, 33, 24, 25, 59, 22, 30]. In the LLM-reasoning literature this signal takes three forms. At the outcome level, reinforcement learning with verifiable rewards [38, 10, 18, 32, 12, 27, 17, 51] runs RL against a binary reward when the final answer matches ground truth. Expert Iteration [2] and rejection sampling fine-tuning [7, 55, 46] use the same outcome signal in an SFT loop: sample candidates, keep those the oracle endorses, fine-tune on the survivors. At the step level, process reward models [50, 23, 52, 58, 37] train a separate verifier and apply it to each reasoning step. `LeAct` uses a graded version of the Expert Iteration signal: each candidate CoT is scored by how much it raises the student's probability of the oracle's full action distribution, a continuous signal that works where the optimal action is mixed and no single token is the correct answer. Conditioning the backward CoT on the expert action is a hindsight-relabelling step [1, 26], re-pairing each state with an outcome the model knows is correct. A complementary line treats reasoning itself as inference-time search [11, 42]; `LeAct` acts at training time, selecting which CoT samples become supervision rather than how to deploy them.

3 Method

The goal is to convert action-only oracles into thinking traces for training LLMs. Two obstacles: the reasoning trace z is never observed, inherently a latent variable; and the observed actions are in structured game action space rather than text, modality misalignment. We design a closed-loop that adapts Ruan et al. [36]'s proposal-and-score framework to our scope of action supervision instead of text backfilling.

3.1 Problem Formulation

Setup. We consider a decision-making setting where states $x \in \mathcal{X}$ are drawn from a task of interest, in which an action oracle (a `CFR` solver, a classical planner, a verifiable controller, or a trained policy) supplies a per-state expert action distribution $\pi^*(\cdot \mid x)$. The student LLM follows a reason-then-act paradigm, generating an intermediate chain-of-thought z before committing to an action y; we model this explicitly by treating z as a latent reasoning variable and factoring the joint distribution as $p_\theta(z,\, y \mid x) \;=\; p_\theta(z \mid x)\, p_\theta(y \mid x,\, z)$. Training data is $\{(x_i, \pi^*(\cdot \mid x_i))\}_i$: we observe states and expert policies but *no* reasoning traces.

Objective. Maximise the marginal log-likelihood of expert actions under the student LLM,

$$\mathcal{J}(\theta) \;=\; \mathbb{E}_x\, \mathbb{E}_{y\sim\pi^*(\cdot|x)}\big[\log p_\theta(y \mid x)\big], \tag{1}$$

intractable because $p_\theta(y \mid x) = \sum_z p_\theta(z \mid x)\, p_\theta(y \mid x, z)$ marginalises a latent natural-language trace. For a fixed trace, the ideal action-level expectation can be evaluated against a full oracle distribution, or a Dirac mass for a single-action oracle. The practical textual-SFT surrogate is described in §3.3.

3.2 A View from Variational-EM

Backward proposal and IWAE bound. Introduce a backward proposal $q_{\mathrm{bwd}}(z \mid x, y)$, instantiated by prompting the LLM to generate a CoT given (x, y), and draw N candidates $z_1, \ldots, z_N \sim q_{\mathrm{bwd}}(\cdot \mid x, y)$ per state. The N-sample importance-weighted bound [6], which we denote $\mathcal{L}_N(\theta)$, is

$$\mathcal{J}(\theta) \;\geq\; \mathcal{L}_N(\theta) \;:=\; \mathbb{E}_x\, \mathbb{E}_{y\sim\pi^*}\, \mathbb{E}_{z_{1..N}\sim q_{\mathrm{bwd}}(\cdot|x,y)} \left[\log \frac{1}{N} \sum_{i=1}^{N} \frac{p_\theta(z_i, y \mid x)}{q_{\mathrm{bwd}}(z_i \mid x, y)} \right], \tag{2}$$

no looser than the $N{=}1$ ELBO under the assumptions of Burda et al. [6]. The proposal may be held fixed. Burda et al. [6, Eq. 8] show that the gradient of $\mathcal{L}_N(\theta)$ is a per-sample weighted SFT update,

$$\nabla_\theta\, \mathcal{L}_N \;=\; \mathbb{E}_x\, \mathbb{E}_{y\sim\pi^*}\, \mathbb{E}_{z_{1..N}\sim q_{\mathrm{bwd}}} \left[\sum_{i=1}^{N} \widetilde{w}_i\, \nabla_\theta \log p_\theta(z_i, y \mid x) \right], \tag{3}$$

with normalised soft weights $\widetilde{w}_i = w_i / \sum_j w_j$ and $w_i = p_\theta(z_i, y \mid x)/q_{\text{bwd}}(z_i \mid x, y)$, and q_{bwd} held fixed at the previous-round parameters within each round, evolving across rounds through the shared M-step updates.

Factorising the IWAE weight w_i into a task term and a proposal term. Bayes' rule factorises each per-sample weight w_i from Eq. (3) as

$$w_i \;=\; \underbrace{\frac{p_\theta(y \mid x, z_i)}{p_\theta(y \mid x)}}_{=\exp \Delta(z_i;\, x, y)} \cdot \underbrace{\frac{p_\theta(z_i \mid x)}{q_{\text{bwd}}(z_i \mid x, y)}}_{\rho(z_i;\, x, y)} \cdot p_\theta(y \mid x), \tag{4}$$

with $\Delta(z;\, x, y) := \log p_\theta(y \mid x, z) - \log p_\theta(y \mid x)$. The marginal $p_\theta(y \mid x)$ is constant in i and cancels in $\widetilde{w}_i = w_i / \sum_j w_j$. The two remaining factors live on opposite sides of the action–CoT modality boundary. Δ is a ratio over the structured expert action y (a probability vector or discrete label whose task-level meaning is independent of its language rendering), and asks whether z_i raises the student's probability of y (implementation in §3.3). ρ is a ratio over the free-form CoT z_i, which has no such structured target; it measures whether the proposal exploited y in *generating* z_i, not whether z_i usefully *decides* y. We use $\widetilde{w}_i \propto \exp \Delta(z_i;\, x, y)$ as a task-grounded surrogate, omitting the proposal-ratio factor. This is a heuristic rather than an identity. §6 verifies the resulting Δ-only filter is load-bearing; App. A.1 gives the bound-level statement and the conditions under which the ρ-drop preserves ranking.

LeAct surrogate objective. The rho-dropped weights motivate the following weighted-SFT surrogate. For this theoretical surrogate, the candidate scores Δ_i and traces are treated as fixed:

$$\max_\theta \; \mathbb{E}_x \, \mathbb{E}_{y \sim \pi^*(\cdot \mid x)} \, \mathbb{E}_{z_{1..N} \sim q_{\text{bwd}}} \Big[\sum_{i=1}^{N} \widetilde{w}_i \, \underbrace{\log p_\theta(z_i,\, y \mid x)}_{\text{SFT loss}} \Big], \quad \widetilde{w}_i \;=\; \frac{e^{\Delta_i}}{\sum_{j=1}^{N} e^{\Delta_j}}. \tag{5}$$

The practical procedure additionally conditions on a summary of the full oracle policy, averages scores over actions, and uses SFT on a positive-score top-K subset (§3.3). These are surrogate choices, not an exact reduction of Eq. (5).

3.3 `LeAct` in Practice

We now specify the three stages that produce the SFT dataset $\mathcal{D}$ and realise the M-step: backward generation of CoT candidates, forward-delta evaluation with top-K selection, and expert-policy forcing combined with joint backward supervision.

Backward generation (E-step proposal). We instantiate the backward proposal q_{bwd} by prompting the current student. The prompt gives the state x and a qualitative summary $\tilde{\pi}^*$ of the expert distribution $\pi^*(\cdot \mid x)$ (for example, "mostly fold with occasional calls"), and asks the student to explain why this strategy is optimal. We draw N candidates $z_1, \ldots, z_N$ per state. The qualitative redaction prevents the backward generator from copying π^* verbatim from its prompt (example traces in App. G). Conditioning on the expert action anchors the explanation in a known-good answer rather than searching forward for one, applying the hindsight-relabelling intuition (§2) to CoT supervision: the silent action becomes a teaching signal once paired with the explanation it licenses.

Forward delta scoring and top-K selection. For each candidate z_n we use a parsed-policy proxy for $\Delta(z_n; x, y)$ from Eq. (4); both terms are read from the student's textual policy line rather than its action-token likelihoods (App. A.4). The conditional $\log p_\theta(y \mid x, z_n)$ uses the line decoded under the candidate trace; the baseline $\log p_\theta(y \mid x) = \log \mathbb{E}_{z \sim p_\theta(z \mid x)} \big[p_\theta(y \mid x, z) \big]$ is estimated by Monte Carlo from M self-generated forward traces $z^{(i)} \sim p_\theta(z \mid x)$ (no gold-action conditioning). We then form the per-state average $\bar{\Delta}(z; x) = \mathbb{E}_{y \sim \pi^*}[\Delta(z; x, y)]$. For oracles that commit to a single action rather than a distribution, the log-likelihood terms in $\Delta(z; x, y)$ are replaced with a heuristic action-match reward against y^*; the rest of the pipeline is unchanged.

Carrying the softmax weights $\widetilde{w}_i \propto e^{\Delta_i}$ into the M-step costs $O(N \cdot |\mathcal{X}|)$. We sparsify in two complementary steps:

$$\mathcal{Z}_x \;=\; \text{top-}K \big\{ z_j \sim q_{\text{bwd}} : \bar{\Delta}(z_j;\, x) > 0 \big\}, \qquad \mathcal{D} \;=\; \{(x, z) : x \in \mathcal{X},\; z \in \mathcal{Z}_x\}. \tag{6}$$

Algorithm 1: The `LeAct` algorithm

Input : Expert oracle π^*, initial model θ_0, states $\mathcal{X}$, backward prompt template P_{bwd}, candidates N, selection count K, rounds R
Output : Trained model θ_R
for *round* $r = 1, 2, \ldots, R$ **do**
  `// E-step: backward generation`
  for *each state* $x \in \mathcal{X}$ **do**
    Sample $z_1, \ldots, z_N \sim p_{\theta_{r-1}}\big(\cdot \mid P_{\mathrm{bwd}}(x, \tilde{\pi}^*(x))\big)$
  `// E-step: forward delta scoring (impl. App. A.4)`
  for *each candidate* (x, z_j) **do**
    $\bar{\Delta}_j \leftarrow \bar{\Delta}(z_j; x)$ `// average Δ from Eq. (4),` $y \sim \pi^*(\cdot \mid x)$
  `// Hard top-`K` on positive deltas`
  for *each state* x **do**
    $\mathcal{Z}_x \leftarrow \text{top-}K\{z_j : \bar{\Delta}_j > 0\}$
  `// M-step: forward + backward joint SFT`
  $\mathcal{D}_r^{\mathrm{fwd}} \leftarrow \{(x,\ (z, \pi^*(\cdot \mid x))) : x \in \mathcal{X},\ z \in \mathcal{Z}_x\}$
  $\mathcal{D}_r^{\mathrm{bwd}} \leftarrow \{((x, \tilde{\pi}^*),\ z) : x \in \mathcal{X},\ z \in \mathcal{Z}_x\}$
  $\theta_r \leftarrow \mathrm{SFT}\big(\theta_{r-1},\ \mathcal{D}_r^{\mathrm{fwd}} \cup \mathcal{D}_r^{\mathrm{bwd}}\big)$
return θ_R

The positive-$\bar{\Delta}$ filter admits only traces that outperform the estimated forward baseline under our scoring rule. This selection rule does not by itself guarantee improvement in the IWAE bound or the marginal-likelihood objective. The top-K indicator is the softmax truncation, cutting cost to $O(K \cdot |\mathcal{X}|)$. The gate tightens across rounds: as the student improves, traces that merely match its current forward output drop out.

Expert-policy forcing and joint backward supervision (M-step). The practical update uses a joint SFT loss $\mathcal{L} = \mathcal{L}_{\mathrm{fwd}} + \mathcal{L}_{\mathrm{bwd}}$ on the traces selected by Eq. (6). Hard selection and the auxiliary backward loss are additional design choices, not algebraic consequences of Eq. (5). $\mathcal{L}_{\mathrm{fwd}}$ trains the forward student on inputs x to emit z followed by the oracle's full action distribution $\pi^*(\cdot \mid x)$. The analogy is teacher forcing in sequence modelling: there the ground-truth token replaces the model's own sample as the next-step target; here the oracle's distribution replaces any student-decoded policy as the action target. We call this *expert-policy forcing* (EPF). $\mathcal{L}_{\mathrm{bwd}}$ retrains the backward proposal on inputs $(x, \tilde{\pi}^*)$ to emit z, the reweighted-wake-sleep [3] update on q_{bwd}. For a distributional oracle, EPF uses the oracle policy as the textual action target; for a single-action oracle, it uses that action. The ideal action-space cross-entropy identity and its relation to textual SFT are distinguished in App. A.2. Backward SFT trains the proposal to reproduce the selected traces. Because selection uses a proposal-ratio-free score, hard truncation, and shared parameters, this update is EM-inspired but carries no general guarantee of monotonic improvement in the IWAE bound or marginal likelihood. Full formulae are in App. A.2; iteration dynamics are in App. C.

4 Experimental Setup

We evaluate `LeAct` on six imperfect-information games (Leduc Hold'em at three scales, Liar's Dice, 3-Player Leduc, Flop Hold'em) and the BuilderBench [8] robotics cube-stacking benchmark, summarised in Table 1; per-domain rules and oracle construction are in App. H.1, F.

Recipe and baselines. We use Qwen3-8B [34] as the student LLM throughout. Fixing the backbone is by design: it isolates supervision quality from model scale, Qwen3-8B is a representative open-weight student, and 8B fits within the compute budget required to fully train all seven settings. The base model lacks the game-specific forward/backward competence to seed reasoning traces directly, so the two iterative methods (CoT ExIt and `LeAct`) both initialise from a frontier-LLM-generated coldstart corpus and run 3 training rounds. The coldstart is shared per-setting between the two, so any gap between them is attributable to the iteration phase. Full setup details are in App. H.

- **NoCoT BC** skips the CoT-candidate step: a single SFT pass on $(x, \pi^*(\cdot \mid x))$ pairs without a reasoning channel. Because it produces no CoT, it is not directly comparable on a data-centric basis with the iterative methods; we report it as a memorisation anchor.

Table 1: **Evaluation spans four orders of magnitude in scale across game families and a robotics domain.** Per-setting oracle, metric, and regime.

Domain	Scale	Oracle	Metric	Regime (split)
Leduc 6r2s	4,032	`CFR`	Exploitability	Imitation (full tree)
Leduc 10r4s	47,040	`CFR`	Exploitability	Both (full / rank-split)
Leduc 13r4s	79,872	`CFR`	Exploitability	Imitation (full tree)
Liar's Dice	24,576	`CFR`	Exploitability	Imitation (full tree)
3-Player Leduc	13,878	`CFR`	`NashConv`	Imitation (full tree)
Flop Hold'em (FHP)	$\sim 10^9$	`DeepCFR`	KL + mbb/g	Generalisation (20K hold-out)
BuilderBench	46 tasks	Frontier-LLM traj.	task progress	Both (in-domain / OOD)

"Both" = the game is evaluated under both imitation (full tree) and generalisation (held-out split) regimes.

- **CoT ExIt** is standard expert iteration [2, 40] with a thinking model: each round samples $N{=}8$ forward rollouts (z, y) per state, ranks them by KL of the model's parsed action distribution to the oracle, retains the top-$K{=}2$ as next-round SFT data, and uses the parsed action distribution itself (not π^*) as the action target (the standard expert-iteration practice; we ablate this in §6).

- `LeAct` samples $N{=}8$ backward CoT candidates per state conditioned on the expert action, ranks them by forward delta against the student's own self-baseline ($M{=}4$ parallel forward decodings, no gold-action conditioning), retains the top-$K{=}2$ as next-round SFT data, and uses π^* as the action target (expert-policy forcing). The ideal action-space objective motivates this target choice (§3.2); the practical textual-SFT choice is evaluated in §6.

Two reasons we do not implement RL/policy-gradient baselines (e.g., `GRPO` [10]): this work targets SFT / mid-training, and CoT ExIt already approximates a coarse-grained REINFORCE through its rollout-and-bootstrap loop. Each round retains up to $K{=}2$ CoTs per state; `LeAct` also requires positive delta. CoT ExIt trains on the forward direction only (up to $K|\mathcal{X}|$ examples per round), while `LeAct` jointly trains forward and backward directions (up to $2K|\mathcal{X}|$ per round). For each (method, setting) we report at the round (out of 3) with the best evaluation metric. Per-round dynamics, iteration diagnostics, and R1-only separation evidence are in App. C.

Two output formats. The seven settings split into two regimes by what the model emits at decoding time. The six imperfect-information games (Leduc, Liar's Dice, 3-Player Leduc, FHP) ask the model to output a mixed-strategy *distribution* per state. Each forward decode emits a parsed action distribution whose KL to the oracle π^* is the unit of supervision. BuilderBench instead asks for a single JSON action per turn over a multi-turn cube-stacking rollout: success is multi-step progress. Absolute scores are not comparable across regimes; only within-regime method ordering is. Because BuilderBench's per-state KL is unavailable when the oracle commits to a single action, we replace forward-delta scoring (and CoT ExIt's ranking) with a heuristic action-match reward against the oracle's action y^* for both methods (App. D.2.1); the rest of the recipe is unchanged.

Metrics. For two-player zero-sum games (Leduc, Liar's Dice) we report *exploitability* (best-response gap to Nash; $0 =$ Nash). For 3-Player Leduc we report `NashConv` [19], an n-player generalisation that reduces to $2\times$ exploitability when $n{=}2$. At FHP we report mean KL to the `DeepCFR` teacher and chip-outcome win rate in *mbb/g* (milli-big-blinds per game, the standard heads-up play-strength unit; App. D.3). On BuilderBench we report *cube-placement progress* (per-task fraction of cubes placed at target positions, averaged over 64 inference rollouts per task and then across tasks). All reasoning-game metrics are reported under **single** (one CoT $\rightarrow$ one action) and **BoN** (per infoset, the candidate with smallest KL to π^* across N samples; oracle-aware), an oracle-assisted candidate-pool diagnostic rather than an oracle-free inference policy. All are lower-is-better, except BuilderBench's cube-placement progress.

Evaluation regime. Imitation is evaluated on the full game tree at enumerable scales (Leduc, Liar's Dice, 3-Player Leduc) and on BuilderBench's in-domain 26-task partition (where the trajectory pool produced at least one fully-successful episode). Generalisation is evaluated on a held-out infoset split at FHP, a feature-disjoint rank-split at Leduc 10r4s (test ranks never appear in training), and BuilderBench's OOD 20-task partition (where the best-of-3 frontier-model trajectory pool produced no successful trajectory; App. H.2).

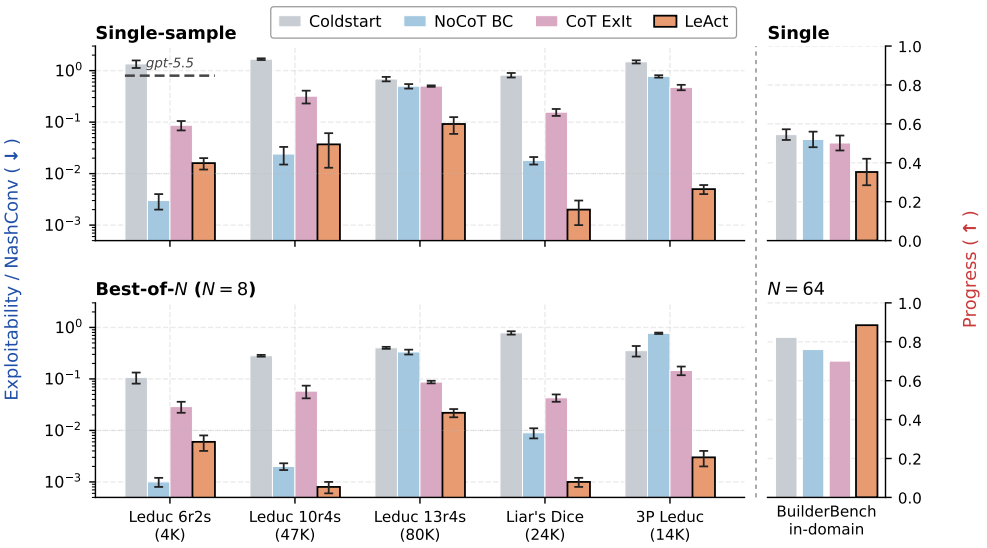


Figure 2: **Imitation quality: `LeAct` matches NoCoT BC where memorisation fits, and dominates where it doesn't.** Five game settings (4K–80K infosets, log y-axis, lower better, BoN uses $N{=}8$) and BuilderBench in-domain 26 tasks (linear y-axis, higher better; $N{=}64$). Numbers in Table 8.

5 Results

`LeAct` treats expert actions as imitation supervision while producing a latent CoT alongside each action; the empirical question is whether this latent helps relative to imitation without latent (NoCoT BC) or with self-generated latent that bypasses our backward-and-score loop (CoT ExIt). We organise the comparison along two axes. **Imitation quality** (§5.1): with shared coldstart and training states for the iterative methods, how closely does the trained forward policy track the expert? We measure across enumerable game scales (Leduc 4K–80K), player counts (3-player Leduc), game families (Liar's Dice), and domains (BuilderBench cube-stacking). **Generalisation** (§5.2): does the advantage persist on states the model never saw during training, where memorisation alone would not transfer? We test on the Flop Hold'em hold-out infoset, the Leduc rank-split, and BuilderBench's OOD task partition.

5.1 Imitation

`LeAct` matches or beats both baselines on BoN inference everywhere outside the 6r2s memorisation regime (Fig. 2). The BoN results suggest a contribution from candidate diversity: at Leduc 10r4s, `LeAct`'s single-to-BoN ratio is several times that of NoCoT's, and on BuilderBench in-domain `LeAct` starts at the lowest single-sample progress but leads at BoN across all four recipes. These results describe the quality of the candidate pool under oracle-aware selection, not an oracle-free deployment policy.

Scaling on poker. At 6r2s (4K infosets) the Nash table fits in model capacity and NoCoT's tabular memorisation suffices. At 13r4s (80K infosets) memorisation breaks: NoCoT BC also stalls (App. D.1), and CoT ExIt is $5.4\times$ worse on single-sample and $4\times$ worse on BoN than `LeAct` despite shared coldstart and training states. This compares the full recipes; selection-specific evidence is provided in §6. The trend is consistent: as scale grows, the value of selecting which CoTs to learn from grows.

Cross-domain results. Beyond two-player Leduc the same imitation gap appears in three further regimes (different game family, multi-player, robotics). On Liar's Dice (different game family), `LeAct` lands within solver-evaluation noise of the `CFR` oracle. On 3-player Leduc, `LeAct` lands within solver noise while ExIt stalls well above it. This is an empirical comparison; we do not claim a general convergence or monotonic-improvement guarantee for either training procedure. On BuilderBench cube-stacking in-domain set (using a frontier-LLM successful trajectory pool as

Table 2: `LeAct` **generalises across three held-out regimes.** (a) KL (↓) on a random infoset hold-out (FHP) and a feature-disjoint rank-split (Leduc 10r4s, test ranks never appear in training); cube-placement progress (↑) on the BuilderBench (BB) OOD partition. Two frontier-anchor rows: gpt-5.5 pass@1 on FHP (a comparator, not data source); the best-of-3 frontier-LLM trajectory pool used as BuilderBench SFT data, evaluated as a strength baseline on the same OOD partition. (b) FHP chip outcome in mbb/g over 50K paired hands (head-to-head; ± standard error). Details in App. D.3, D.2.2.

(a) Held-out generalisation.

	Leduc 10r4s (↓)		FHP 20K (↓)		BB OOD (↑)
Method	Single	BoN	Single	BoN	Progress
gpt-5.5	—	—	0.7530	—	—
Frontier traj.	—	—	—	—	0.4923
Coldstart	2.462	0.332	0.0321	0.0018	0.4815
NoCoT BC	2.110	0.574	0.0145	0.0049	0.3980
CoT ExIt	2.071	0.459	0.0053	0.0011	0.4361
`LeAct`	**0.716**	**0.168**	**0.0019**	**0.0005**	**0.5777**

(b) FHP chip outcome.

	NoCoT	CoT	`LeAct`
NoCoT	—	$+41 \pm 31$	-60 ± 13
CoT	-41 ± 31	—	-62 ± 28
`LeAct`	$+60 \pm 13$	$+62 \pm 28$	—

oracle), `LeAct` is the only training recipe whose BoN progress clears the joint coldstart anchor; the matched CoT ExIt comparator even regresses below it (Table 8).

Absolute KL and exploitability values are not directly comparable across these games: action-space size and Nash-policy concentration both differ, and large game trees like FHP have many infosets where the optimal move is near-deterministic, pulling the average KL floor lower. What is interpretable across rows is therefore the within-game ordering rather than the absolute scale.

5.2 Generalisation

`LeAct` carries its advantage to states the model never saw during training, across three regimes: a random infoset hold-out at million-scale (FHP), a feature-disjoint rank-split at medium scale (Leduc 10r4s), and a task-level OOD partition on BuilderBench. Random hold-outs test that the trained policy is not memorising the training infosets; feature-disjoint splits stress transfer along an axis the model never saw; the BuilderBench OOD partition stresses generalisation beyond the supervised demonstration distribution.

Random hold-out at scale: FHP. Flop Hold'em has on the order of 10^9 playable infosets, several orders of magnitude beyond the 80K training infosets, so even random sampling exposes the policy to a state distribution it has never been trained on, making FHP a natural generalisation testbed even under the random hold-out regime. The absolute KL floor is small because most infosets have near-trivial optimal strategy (preflop fold lines on weak hands). The within-regime ordering is nonetheless sharp: `LeAct` beats both baselines, and a frontier zero-shot anchor (gpt-5.5; Table 2a) lands orders of magnitude worse. Even a strong general reasoner is therefore severely under-optimised at producing calibrated numerical mixed strategies for a fine-grained game. Chip-outcome head-to-head (Table 2b) also favors `LeAct` over both trained baselines.

Feature-disjoint shift: Leduc 10r4s rank-split. The rank-split (train ranks $\{2, 4, 6, 8, \mathrm{T}\}$, test $\{3, 5, 7, 9, \mathrm{J}\}$) forces the model to transfer along a feature axis it never saw during training, a stronger distribution-shift test than random hold-out. All three methods are retrained from scratch for this split: the joint coldstart corpus is regenerated on the train ranks alone, and CoT ExIt and `LeAct` then iterate from this rank-restricted coldstart base. The test ranks $\{3, 5, 7, 9, \mathrm{J}\}$ are never seen during coldstart or any iteration round. `LeAct`'s $\sim 3\times$ lead over CoT ExIt on the held-out ranks (Table 2a) provides evidence of transfer beyond the ranks used for training.

Task-level OOD on BuilderBench. The OOD partition is 20 tasks that no frontier model in the best-of-3 trajectory pool fully solved, so the SFT data contains no golden trajectory for any of these. We report *cube-placement progress*, a partial-solve metric (fraction of cubes correctly placed), so non-zero scores correspond to placing some cubes correctly even when the task is not solved end-to-end. `LeAct` achieves the highest mean progress on this partition (Table 2a), whose tasks have no fully successful demonstration in the frontier-model trajectory pool.

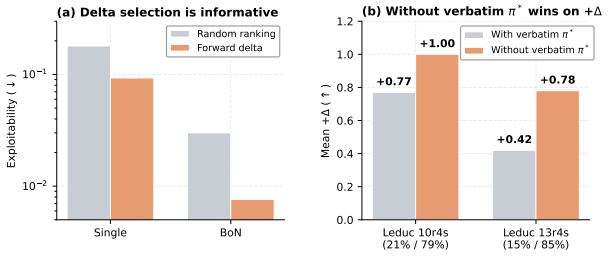


Figure 3: **Two falsifiers of the shortcut hypothesis for backward-delta selection.** (a) Forward-delta vs. random ranking on Single and BoN (Leduc 10r4s). (b) Mean $+\Delta$ by verbatim-π^* status.

Method	Regime	Single (↓)	BoN (↓)
Coldstart	—	1.6701	0.2824
CoT ExIt	EPF	1.0278	0.6191
CoT ExIt	no-EPF	0.4432	0.0599
`LeAct`	EPF	**0.1566**	**0.0019**
`LeAct`	no-EPF	1.0565	0.1940

Table 3: **Method and supervision target are entangled** We cross method (CoT ExIt vs `LeAct`) with *expert-policy forcing* (use the oracle's π^* vs the model's own action) on Leduc `10r4s`. Matched pairings: `LeAct`+EPF and CoT ExIt+no-EPF.

6 Analysis

§5.1 reports `LeAct`'s gap over the CoT ExIt comparator. The analyses below examine selection and target choice. This section runs four probes that test whether the gap might instead come from one of four alternatives, and closes with a compute-overhead analysis. (1) Could any ranking rule have done equally well? (2) Does our parsed-string proxy faithfully measure the formal Δ? (3) Is the gain just memorising surface features (e.g. Nash numbers)? (4) Or is it the supervision-target choice (not backward delta) doing the work?

Delta selection is informative, not random. To test this, we replace forward-delta ranking with random ranking on Leduc 10r4s, sharpened to top-1 (so any gap comes from the ranking rule, not the data budget). Forward-delta beats random on both single and BoN exploitability (Fig. 3a). `LeAct`'s gain therefore comes from *which* CoTs we keep, not just from keeping any $K = 2$ subset.

The textual proxy has partial agreement with a token-score diagnostic. `LeAct` scores CoT candidates by reading the model's parsed policy distribution, not by computing the token-level log-likelihoods that the formal Δ is defined over. This is a proxy. On Leduc 13r4s, the token-score top-1 candidate is positive under the parsed-policy score in 89–98% of infosets containing at least one parsed-score-positive candidate. This is a conditional positivity rate, not top-1 agreement or equivalence to the formal Δ (App. A.4, Table 5).

The selection signal is not memorised Nash numbers. If `LeAct`'s scorer were just rewarding CoTs that quote oracle probabilities verbatim (e.g., "fold 0.376"), then CoTs without such verbatim numbers would score worse. We split the backward pool by verbatim-π^* presence: at both Leduc 10r4s and 13r4s, the larger "no verbatim" stratum has a higher mean positive delta than the smaller "with verbatim" stratum (Fig. 3b). The held-out generalisation results (§5.2) corroborate: a memorisation shortcut would not transfer to unseen states.

Method and supervision target interact. To separate target choice from method choice, we run a 2×2 crossing method (CoT ExIt vs `LeAct`) with target (*expert-policy forcing* or EPF, vs the model's own action). Each cell is a separate run from the shared Leduc 10r4s coldstart, iterated for 2 rounds at matched data budget. `LeAct`+EPF gives the best Single and BoN results in Table 3. All four cells improve on coldstart in Single; three improve in BoN, with CoT ExIt+EPF the exception. The diagonal comparison (0.16 vs 0.44 Single; 0.0019 vs 0.06 BoN) changes both method and target, so it does not isolate backward-delta scoring. The random-ranking ablation above provides the more direct selection comparison. The coherence diagnostic offers a possible explanation of the interaction (App. E.1); BuilderBench provides a cross-domain comparison (App. D.2.2).

Compute overhead. `LeAct` and CoT ExIt both generate N candidate CoTs per state (backward and forward, respectively). `LeAct` additionally uses M forward baseline decodings, candidate scoring, and backward-direction SFT. These are distinct compute costs; matched candidate counts do not establish equal total training compute.

7 Conclusion

The key idea of this paper is that many domains already have a silent domain-specific expert, and that the reasoning trace these experts cannot articulate can be recovered by the student itself and

filtered against the expert's own actions. This makes action-only expert systems a source of reasoning supervision, and the oracle itself need not be exact. Our experiments still use frontier-LLM coldstart data, and BuilderBench additionally uses frontier-model trajectories as its oracle. At Flop Hold'em we score against a `DeepCFR` teacher, and on BuilderBench against a pool of frontier-model trajectories, in both cases without modification to the recipe.

Other silent experts (protein structure predictors, formal proof checkers, robotic-control libraries) are natural candidates for the same recipe, and the most direct algorithmic extension is merging `LeAct` with online RL, where the oracle's action distribution can serve as a process-level baseline for credit assignment. How `LeAct` interacts with frontier-scale students, beyond the 8B fixed here, is another direct extension.

Acknowledgments and Disclosure of Funding

Chi Jin acknowledges support from NSF grants OAC-2411299 and IIS-2239297 and the Princeton AI Lab Seed Fund.

References

[1] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[2] Thomas Anthony, Zheng Tian, and David Barber. Thinking fast and slow with deep learning and tree search. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

[3] Jörg Bornschein and Yoshua Bengio. Reweighted wake-sleep. In *International Conference on Learning Representations (ICLR)*, 2015. arXiv:1406.2751.

[4] Noam Brown and Tuomas Sandholm. Superhuman AI for multiplayer poker. *Science*, 365 (6456):885–890, 2019.

[5] Noam Brown, Adam Lerer, Sam Gross, and Tuomas Sandholm. Deep counterfactual regret minimization. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *PMLR*, pages 793–802, 2019.

[6] Yuri Burda, Roger Grosse, and Ruslan Salakhutdinov. Importance weighted autoencoders. In *International Conference on Learning Representations*, 2016.

[7] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[8] Raj Ghugare, Roger Creus Castanyer, Catherine Ji, Kathryn Wantlin, Jin Schofield, Karthik Narasimhan, and Benjamin Eysenbach. BuilderBench: The building blocks of intelligent agents, 2025. URL `https://arxiv.org/abs/2510.06288`.

[9] Caglar Gulcehre, Tom Le Paine, Srivatsan Srinivasan, Ksenia Konyushkova, Lotte Weerts, Abhishek Sharma, Aditya Siddhant, Alex Ahern, Miaosen Wang, Chenjie Gu, Wolfgang Macherey, Arnaud Doucet, Orhan Firat, and Nando de Freitas. Reinforced self-training (ReST) for language modeling. *arXiv preprint arXiv:2308.08998*, 2023.

[10] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. DeepSeek-R1 incentivizes reasoning in LLMs through reinforcement learning. *Nature*, 645(8081):633–638, 2025. doi: 10.1038/s41586-025-09422-z. arXiv:2501.12948.

[11] Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*, 2023.

[12] Alex Havrilla, Yuqing Du, Sharath Chandra Raparthy, Christoforos Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, Sainbayar Sukhbaatar, and Roberta Raileanu. Teaching large language models to reason with reinforcement learning. *arXiv preprint arXiv:2403.04642*, 2024.

[13] Namgyu Ho, Laura Schmid, and Se-Young Yun. Large language models are reasoning teachers. *arXiv preprint arXiv:2212.10071*, 2023.

[14] Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron Courville, Alessandro Sordoni, and Rishabh Agarwal. V-STaR: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*, 2024.

[15] Thomas Hubert, Rishi Mehta, Laurent Sartran, et al. Olympiad-level formal mathematical reasoning with reinforcement learning. *Nature*, 651:607–613, 2025. doi: 10.1038/s41586-025-09833-y.

[16] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in Neural Information Processing Systems*, 35:22199–22213, 2022.

[17] Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, et al. Training language models to self-correct via reinforcement learning. *arXiv preprint arXiv:2409.12917*, 2024.

[18] Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, et al. Tulu 3: Pushing frontiers in open language model post-training. *arXiv preprint arXiv:2411.15124*, 2024.

[19] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 30, pages 4191–4204, 2017.

[20] Marc Lanctot, Edward Lockhart, Jean-Baptiste Lespiau, Vinicius Zambaldi, Satyaki Upadhyay, Julien Pérolat, Sriram Srinivasan, Finbarr Timbers, Karl Tuyls, Shayegan Omidshafiei, et al. OpenSpiel: A framework for reinforcement learning in games. *arXiv preprint arXiv:1908.09453*, 2019.

[21] Tamera Lanham, Anna Chen, Ansh Radhakrishnan, Benoit Steiner, Carson Denison, Danny Hernandez, Dustin Li, Esin Durmus, Evan Hubinger, Jackson Kernion, et al. Measuring faithfulness in chain-of-thought reasoning. *arXiv preprint arXiv:2307.13702*, 2023.

[22] Zenan Li, Ziran Yang, Deyuan He, Haoyu Zhao, Andrew Zhao, Shange Tang, Kaiyu Yang, Aarti Gupta, Zhendong Su, and Chi Jin. Goedel-Code-Prover: Hierarchical proof search for open state-of-the-art code verification. *arXiv preprint arXiv:2603.19329*, 2026.

[23] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2305.20050.

[24] Yong Lin, Shange Tang, Bohan Lyu, Jiayun Wu, Hongzhou Lin, Kaiyu Yang, Jia Li, Mengzhou Xia, Danqi Chen, Sanjeev Arora, and Chi Jin. Goedel-Prover: A frontier model for open-source automated theorem proving. *arXiv preprint arXiv:2502.07640*, 2025.

[25] Yong Lin, Shange Tang, Bohan Lyu, Ziran Yang, Jui-Hui Chung, Haoyu Zhao, Lai Jiang, Yihan Geng, Jiawei Ge, Jingruo Sun, Jiayun Wu, et al. Goedel-Prover-V2: Scaling formal theorem proving with scaffolded data synthesis and self-correction. *arXiv preprint arXiv:2508.03613*, 2025.

[26] Hao Liu, Carmelo Sferrazza, and Pieter Abbeel. Chain of hindsight aligns language models with feedback. In *International Conference on Learning Representations (ICLR)*, 2024. arXiv:2302.02676.

[27] Trung Quoc Luong, Xinbo Zhang, Zhanming Jie, Peng Sun, Xiaoran Jin, and Hang Li. ReFT: Reasoning with reinforced fine-tuning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, 2024. arXiv:2401.08967.

[28] Chengdong Ma, Ziran Yang, Hai Ci, Jun Gao, Minquan Gao, Xuehai Pan, and Yaodong Yang. Evolving diverse red-team language models in multi-round multi-agent games. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2026. doi: 10.1109/TPAMI.2026.3694367.

[29] Lucie Charlotte Magister, Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. Teaching small language models to reason. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 1773–1781, 2023. arXiv:2212.08410.

[30] Meta Fundamental AI Research Diplomacy Team (FAIR). Human-level play in the game of Diplomacy by combining language models with strategic reasoning. *Science*, 378(6624): 1067–1074, 2022.

[31] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisý, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017.

[32] OpenAI. OpenAI o1 system card, 2024. URL `https://arxiv.org/abs/2412.16720`.

[33] Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving. *arXiv preprint arXiv:2009.03393*, 2020.

[34] Qwen Team. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[35] Geoffrey Roeder, Yuhuai Wu, and David K Duvenaud. Sticking the landing: Simple, lower-variance gradient estimators for variational inference. In *Advances in Neural Information Processing Systems*, 2017.

[36] Yangjun Ruan, Neil Band, Chris J Maddison, and Tatsunori Hashimoto. Reasoning to learn from latent thoughts. *arXiv preprint arXiv:2503.18866*, 2025.

[37] Amrith Setlur, Chirag Nagpal, Adam Fisch, Xinyang Geng, Jacob Eisenstein, Rishabh Agarwal, Alekh Agarwal, Jonathan Berant, and Aviral Kumar. Rewarding progress: Scaling automated process verifiers for LLM reasoning. *arXiv preprint arXiv:2410.08146*, 2024.

[38] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y.K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[39] Chengshuai Shi, Wenzhe Li, Xinran Liang, Yizhou Lu, Wenjia Yang, Ruirong Feng, Seth Karten, Ziran Yang, Zihan Ding, Gabriel Sarch, Danqi Chen, Karthik Narasimhan, and Chi Jin. Odysseus: Scaling VLMs to 100+ turn decision-making in games via reinforcement learning. *arXiv preprint arXiv:2605.00347*, 2026.

[40] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science*, 362(6419):1140–1144, 2018.

[41] Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, et al. Beyond human data: Scaling self-training for problem-solving with language models. *Transactions on Machine Learning Research*, 2024.

[42] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

[43] Finnegan Southey, Michael Bowling, Bryce Larson, Carmelo Piccione, Neil Burch, Darse Billings, and Chris Rayner. Bayes' bluff: Opponent modelling in poker. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 550–558, 2005.

[44] Oskari Tammelin. Solving large imperfect information games using CFR+. *arXiv preprint arXiv:1407.5042*, 2014.

[45] Marc Toussaint. Logic-geometric programming: An optimization-based approach to combined task and motion planning. In *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 1930–1936, 2015.

[46] Hugo Touvron, Louis Martin, Kevin Stone, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[47] Trieu H. Trinh, Yuhuai Wu, Quoc V. Le, He He, and Thang Luong. Solving olympiad geometry without human demonstrations. *Nature*, 625(7995):476–482, 2024. doi: 10.1038/s41586-023-06747-5.

[48] George Tucker, Dieterich Lawson, Shixiang Gu, and Chris J Maddison. Doubly reparameterized gradient estimators for Monte Carlo objectives. In *International Conference on Learning Representations*, 2019.

[49] Miles Turpin, Julian Michael, Ethan Perez, and Samuel R. Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2305.04388.

[50] Jonathan Uesato, Nate Kushman, Ramana Kumar, Francis Song, Noah Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *arXiv preprint arXiv:2211.14275*, 2022.

[51] Huaijie Wang, Shibo Hao, Hanze Dong, Shenao Zhang, Yilin Bao, Ziran Yang, and Yi Wu. Offline reinforcement learning for LLM multi-step reasoning. *arXiv preprint arXiv:2412.16145*, 2024.

[52] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-Shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 9426–9439, 2024. arXiv:2312.08935.

[53] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 35:24824–24837, 2022.

[54] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. ReAct: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*, 2023. arXiv:2210.03629.

[55] Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*, 2023.

[56] Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D Goodman. STaR: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022.

[57] Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D Goodman. Quiet-STaR: Language models can teach themselves to think before speaking. *arXiv preprint arXiv:2403.09629*, 2024.

[58] Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. ReST-MCTS*: LLM self-training via process reward guided tree search. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. arXiv:2406.03816.

[59] Haoyu Zhao, Ziran Yang, Jiawei Li, Deyuan He, Zenan Li, Chi Jin, Venugopal V. Veeravalli, Aarti Gupta, and Sanjeev Arora. AlgoVeri: An aligned benchmark for verified code generation on classical algorithms. *arXiv preprint arXiv:2602.09464*, 2026.

[60] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Advances in Neural Information Processing Systems*, volume 20, 2007.

A Formulation: Derivations

This appendix gives the derivations summarised in §3.2: (i) the IWAE bound, per-sample importance-weight factorisation, and rho-dropped soft-weight approximation that define the `LeAct` training objective (App. A.1); (ii) the M-step target-choice derivation for expert-policy forcing (App. A.2); (iii) the correspondence between formal quantities and `LeAct` algorithm components (App. A.3); and (iv) the textual policy proxy that bridges the formal Δ to its parsed-policy implementation (App. A.4).

A.1 From the ELBO to the LeAct Objective

This appendix expands §3.2 line by line. The derivation specialises the importance-weighted variational bound (IWAE) of Burda et al. [6] to a backward proposal $q_{\mathrm{bwd}}(z \mid x, y)$ held fixed within a round. The practical procedure then replaces the exact importance weights with a task-score surrogate. The same IWAE bound was recently applied to latent reasoning data synthesis on unlabelled text corpora by Ruan et al. [36]; here the target is the conditional likelihood $\log p_\theta(y \mid x)$ for an expert action $y \sim \pi^*$. Step 3 introduces the forward-delta surrogate. Steps 1–2 give the importance-weighted bound and weight factorisation. Steps 3–4 introduce the rho-dropped, hard-filtered SFT surrogate; Step 5 motivates the auxiliary backward loss.

Step 1: IWAE bound via a backward proposal. The marginal log-likelihood of an expert action $y \sim \pi^*(\cdot \mid x)$ under the student is $\log p_\theta(y \mid x) = \log \sum_z p_\theta(z \mid x)\, p_\theta(y \mid x, z)$, intractable over natural-language traces. Introduce a backward proposal $q_{\mathrm{bwd}}(z \mid x, y)$ (instantiated by prompting the LLM to generate a CoT given (x, y)) and draw N candidates $z_1, \ldots, z_N \sim q_{\mathrm{bwd}}(\cdot \mid x, y)$ i.i.d. The N-sample importance-weighted bound of Burda et al. [6] is

$$\log p_\theta(y \mid x) \;\geq\; \mathbb{E}_{z_{1..N} \sim q_{\mathrm{bwd}}} \left[\log \frac{1}{N} \sum_{i=1}^{N} \frac{p_\theta(z_i \mid x)\, p_\theta(y \mid x, z_i)}{q_{\mathrm{bwd}}(z_i \mid x, y)} \right], \tag{7}$$

no looser than the $N{=}1$ ELBO under the assumptions of Burda et al. [6]. Taking the outer expectation over x and $y \sim \pi^*$ gives the population IWAE bound,

$$\mathcal{J}(\theta) \;\geq\; \mathbb{E}_x\, \mathbb{E}_{y \sim \pi^*}\, \mathbb{E}_{z_{1..N} \sim q_{\mathrm{bwd}}} \left[\log \frac{1}{N} \sum_{i=1}^{N} \frac{p_\theta(z_i, y \mid x)}{q_{\mathrm{bwd}}(z_i \mid x, y)} \right]. \tag{8}$$

Burda et al. [6] show (their Eq. 8) that the gradient of Eq. (8) reduces to a per-sample SFT update with normalised soft weights,

$$\nabla_\theta \mathcal{L}_N \;=\; \mathbb{E}_{z_{1..N}} \left[\sum_{i=1}^{N} \widetilde{w}_i\, \nabla_\theta \log p_\theta(z_i, y \mid x) \right], \quad \widetilde{w}_i = \frac{w_i}{\sum_{j=1}^{N} w_j}, \quad w_i = \frac{p_\theta(z_i, y \mid x)}{q_{\mathrm{bwd}}(z_i \mid x, y)}, \tag{9}$$

with q_{bwd} held fixed at θ_{r-1} within the round (so $\nabla_\theta \log q_{\mathrm{bwd}} = 0$ for the per-round bound); the inner term $\log p_\theta(z_i, y \mid x)$ is exactly the joint SFT loss on (x, z_i, y).

A fixed proposal can be used at any N. The per-sample weights $w_i = p_\theta(z_i, y \mid x)/q_{\mathrm{bwd}}(z_i \mid x, y)$ inside the log-mean are importance weights between the joint $p_\theta(z, y \mid x)$ and the proposal $q_{\mathrm{bwd}}(z \mid x, y)$; Bayes' rule factorises them in Step 2. Within each `LeAct` round, q_{bwd} is held fixed at θ_{r-1}; across rounds it evolves alongside the shared model parameters that the M-step trains, recovering an approximate-EM alternation (the correspondence is summarised in App. A.3). The practical pipeline additionally augments the M-step with an explicit backward SFT term (§3.3), which arises as the reverse-KL update on q_{bwd} in the reweighted-wake-sleep regime; Step 5 below derives this.

Step 2: per-sample importance-weight factorisation. For each sample z_i with $q_{\mathrm{bwd}}(z_i \mid x, y) > 0$, Bayes' rule factorises the unnormalised IWAE weight $w_i = p_\theta(z_i, y \mid x)/q_{\mathrm{bwd}}(z_i \mid x, y)$ as

$$w_i \;=\; \underbrace{\frac{p_\theta(y \mid x, z_i)}{p_\theta(y \mid x)}}_{=\exp \Delta(z_i; x, y)} \cdot \underbrace{\frac{p_\theta(z_i \mid x)}{q_{\mathrm{bwd}}(z_i \mid x, y)}}_{\rho(z_i; x, y)} \cdot p_\theta(y \mid x), \tag{10}$$

with $\Delta(z; x, y) := \log p_\theta(y \mid x, z) - \log p_\theta(y \mid x)$. The factor $p_\theta(y \mid x)$ does not depend on i and so cancels in the normalised weight $\widetilde{w}_i = w_i / \sum_j w_j$; only $\exp \Delta_i$ and ρ_i remain. The

first factor $\exp \Delta_i$ is computable from the student's own logits: $\log p_\theta(y \mid x, z_i)$ is a teacher-forced log-likelihood of the action under the student conditioned on z_i, and $\log p_\theta(y \mid x) = \log \mathbb{E}_{z' \sim p_\theta(\cdot|x)}[p_\theta(y \mid x, z')]$ is estimable by Monte Carlo from a small number of forward samples (§3.3).

Step 3: a task-score surrogate for the soft weight. The log per-sample weight admits the additive decomposition

$$\log w_i = \Delta(z_i;\, x, y) + \log \rho(z_i;\, x, y) + \log p_\theta(y \mid x).$$

The two terms defining Δ can be evaluated or estimated from the student's probabilities: $\log p_\theta(y \mid x, z_i)$ is a teacher-forced action log-likelihood, and $\log p_\theta(y \mid x) = \log \mathbb{E}_{z' \sim p_\theta(\cdot|x)}[p_\theta(y \mid x, z')]$ is estimable by Monte Carlo from a small number of forward samples (§3.3). We omit ρ as a scoring design choice. `LeAct` therefore approximates the unnormalised IWAE soft weight by

$$w_{\mathrm{LeAct}}(z_i;\, x, y) \;:=\; \exp \Delta(z_i;\, x, y), \tag{11}$$

i.e. replaces the soft weight $\widetilde{w}_i \propto e^{\Delta_i} \cdot \rho_i$ by the rho-dropped surrogate $\widetilde{w}_i^{\mathrm{LeAct}} \propto e^{\Delta_i}$. The expert action provides a task-grounded score for this surrogate; it does not make the omitted ρ factor vanish. §6 verifies empirically that the resulting Δ-only filter is load-bearing.

Dropping ρ as a heuristic. Dropping ρ is a heuristic, not an identity. It preserves ranking when ρ is constant across candidates; more generally, a pairwise ordering is preserved when the corresponding difference in $\log \rho$ is smaller in magnitude than the score gap. Low correlation between $\log \rho$ and Δ alone does not ensure ranking preservation. The parsed-policy versus token-score audit in App. A.4 evaluates the scoring proxy, not the omitted proposal-ratio factor.

Step 4: the LeAct objective and the hard top-K sparsification. The rho-dropped surrogate $\widetilde{w}_i^{\mathrm{LeAct}} \propto e^{\Delta_i}$ motivates the following weighted-SFT update. For this theoretical surrogate, the candidate scores Δ_i and traces are treated as fixed:

$$\max_\theta \; \mathbb{E}_x \, \mathbb{E}_{y \sim \pi^*} \, \mathbb{E}_{z_{1..N} \sim q_{\mathrm{bwd}}} \left[\sum_{i=1}^{N} \widetilde{w}_i^{\mathrm{LeAct}} \log p_\theta(z_i,\, y \mid x) \right], \quad \widetilde{w}_i^{\mathrm{LeAct}} = \frac{e^{\Delta_i}}{\sum_{j=1}^{N} e^{\Delta_j}}, \tag{12}$$

This is the surrogate objective in Eq. (5). `LeAct` replaces the softmax weighting $\widetilde{w}_i^{\mathrm{LeAct}}$ with a hard top-K filter on the average delta $\bar{\Delta}(z;\, x) := \mathbb{E}_{y \sim \pi^*}[\Delta(z;\, x, y)]$ (a sparse approximation that keeps only the high-weight candidates): train on $(x, z, \pi^*(\cdot \mid x))$ tuples with $z \in \mathcal{Z}_x = \text{top-}K\{z_j \sim q_{\mathrm{bwd}} : \bar{\Delta}(z_j; x) > 0\}$, and use the full oracle distribution $\pi^*(\cdot \mid x)$ as the action target, the *expert-policy-forcing* M-step derived in App. A.2. Omitting ρ and replacing action-specific soft weights with a hard filter on the action-averaged score are heuristic steps. In general, averaging over y does not commute with softmax weighting; the practical objective is an EM-inspired surrogate.

Step 5: joint backward supervision as the reverse-KL update on the proposal. Steps 1–4 fix q_{bwd} within a round and only update the shared model parameters in p_θ. The IWAE bound is tightest when q_{bwd} matches the true posterior $p_\theta(z \mid x, y)$: at $N{=}1$ the gap is exactly $\mathrm{KL}\big(q_{\mathrm{bwd}}(\cdot \mid x, y) \,\|\, p_\theta(\cdot \mid x, y)\big)$, and at $N{>}1$ the gap is a soft generalisation of the same divergence [6]. Reducing this gap requires a separate update step on q_{bwd}. The natural choice (gradient ascent on the IWAE bound itself with respect to q_{bwd}'s parameters) is the "wake-phase" update of variational EM, but its signal-to-noise ratio degrades as q_{bwd} approaches the posterior, requiring the STL/DReG corrections of Roeder et al. [35], Tucker et al. [48]; for an LLM-parameterised proposal this is also expensive (each gradient step requires backprop through the IWAE log-mean). Reweighted wake-sleep [3] replaces the wake-phase update with a reverse-direction KL,

$$\min_{q_{\mathrm{bwd}}} \; \mathrm{KL}\big(p_\theta(\cdot \mid x, y) \,\big\|\, q_{\mathrm{bwd}}(\cdot \mid x, y)\big) \;=\; \min_{q_{\mathrm{bwd}}} \; \mathbb{E}_{z \sim p_\theta(\cdot|x,y)}\big[-\log q_{\mathrm{bwd}}(z \mid x, y)\big] + \mathrm{const}, \tag{13}$$

which is a maximum-likelihood objective on q_{bwd} with samples drawn from the posterior. The posterior is itself intractable, so we importance-sample from q_{bwd}: $\mathbb{E}_{z \sim p_\theta(\cdot|x,y)}\big[-\log q_{\mathrm{bwd}}(z \mid x, y)\big] \approx \mathbb{E}_{z_{1..N} \sim q_{\mathrm{bwd}}}\big[\sum_i \widetilde{w}_i \, (-\log q_{\mathrm{bwd}}(z_i \mid x, y))\big]$, recovering the same self-normalised IWAE weights $\widetilde{w}_i$ used in Step 4. The right-hand side is a finite-sample self-normalised importance-sampling approximation and is generally biased. Substituting the rho-dropped surrogate $\widetilde{w}_i^{\mathrm{LeAct}} \propto e^{\Delta_i}$ and then the hard top-K indicator on $\mathcal{Z}_x$ used by the M-step yields the practical update

$$\mathcal{L}_{\mathrm{bwd}}(\theta) \;=\; -\,\mathbb{E}_{(x,z) \in \mathcal{D}_r}\big[\log p_\theta(z \mid x, \tilde{\pi}^*)\big], \tag{14}$$

i.e., supervised cross-entropy training of q_{bwd} on the same selected positive-Δ traces $\mathcal{D}_r$ that drive the M-step (with $\tilde{\pi}^*$ the redacted-distribution prompt as defined in §3.2). Because q_{bwd} shares parameters with p_θ (one decoder, two prompts), the wake-sleep update lands on the same θ and Eq. (14) is the joint backward term in §3.3. Backward SFT trains the proposal to reproduce the selected traces. Because selection uses a proposal-ratio-free score, hard truncation, and shared parameters, this update is EM-inspired but carries no general guarantee of monotonic improvement in the IWAE bound or marginal likelihood. The empirical effect of including this term is ablated in App. C.

EM as a special case. Setting $N = 1$ reduces Eq. (8) to the standard ELBO. Choosing $q_{\text{bwd}}(\cdot \mid x, y) = p_{\theta^{(t-1)}}(\cdot \mid x, y)$ gives the exact EM E-step. At $\theta = \theta^{(t-1)}$, Bayes' rule yields $\rho(z; x, y) = \exp[-\Delta(z; x, y)]$, so $\exp(\Delta)\rho = 1$, although neither factor is generally one individually. Thus $w_i = p_{\theta^{(t-1)}}(y \mid x)$, all normalised weights equal $1/N$, and the bound is tight for any $N \geq 1$. Separately, for $N = 1$ the single normalised weight equals one for any valid proposal. `LeAct` instead uses an LLM-defined backward proposal and the ρ-dropped, hard-filtered surrogate described above; its EM-inspired alternation does not inherit the exact-EM monotonicity guarantee. The `LeAct`–EM correspondence is summarised in App. A.3.

A.2 Expert-Policy Forcing: Target Choice and Textual SFT

For a fixed (x, z), the ideal action-space cross-entropy admits sampled-action and full-distribution estimators. This identity motivates target choice; it does not make textual policy-line SFT an exact implementation of Eq. (5), whose proposals and weights also depend on y. The practical procedure instead selects a shared trace pool using a policy-conditioned proposal and an averaged score.

Sampled expert-action target. Draw $\tilde{y} \sim \pi^*$ and minimise $-\log p_\theta(\tilde{y} \mid x, z)$. This is an unbiased estimator of the action-space cross-entropy for fixed (x, z). Our no-EPF baseline instead uses the model-decoded policy, which is a different target.

Expert-distribution target (expert-policy forcing). Because the oracle supplies the *full* π^*, the outer expectation can be evaluated exactly:

$$\mathbb{E}_{y \sim \pi^*}\big[-\log p_\theta(y \mid x, z)\big] \;=\; H(\pi^*) + D_{\text{KL}}\big(\pi^*(\cdot \mid x) \,\|\, p_\theta(\cdot \mid x, z)\big), \tag{15}$$

so minimising this ideal action-space loss is equivalent to minimising $D_{\text{KL}}(\pi^* \,\|\, p_\theta(\cdot \mid x, z))$. The textual-SFT surrogate below uses the oracle policy as its output target.

When the choices matter. Sampled expert actions and the full oracle distribution have the same expected action-space gradient, although their estimator variance differs. Using the model-decoded policy changes the target and generally changes that expected gradient. Delta scoring (Eq. (4)) already evaluates Δ against the action distribution under π^*, so taking π^* as the SFT target keeps the E- and M-steps aligned on the same distribution. §6 shows this alignment is load-bearing: removing it reverses which algorithm wins at 47K information sets.

The two-part loss. The selected traces $\mathcal{D}_r$ are reused in two prompt formats that share the same θ. The idealised forward loss

$$\mathcal{L}_{\text{fwd}}(\theta;\, \mathcal{D}_r) \;=\; -\,\mathbb{E}_{(x,\, z) \in \mathcal{D}_r}\Big[\log p_\theta(z \mid x) \;+\; \mathbb{E}_{y \sim \pi^*(\cdot \mid x)}\big[\log p_\theta(y \mid x, z)\big]\Big] \tag{16}$$

is the ideal action-space objective. The policy-line SFT described in App. A.4 is a surrogate for this loss. The backward loss

$$\mathcal{L}_{\text{bwd}}(\theta;\, \mathcal{D}_r) \;=\; -\,\mathbb{E}_{(x,\, z) \in \mathcal{D}_r}\big[\log p_\theta(z \mid x,\, \tilde{\pi}^*)\big] \tag{17}$$

retrains the proposal q_{bwd} on the same selected z given the $(x, \tilde{\pi}^*)$ prompt, the wake-sleep update on q_{bwd} derived in App. A.1 Step 5. The described practical update combines forward and backward SFT on the selected traces; the action-space loss above provides an idealised motivation.

A.3 Algorithm–Formulation Correspondence

Table 4 maps each formal quantity from App. A.1 to its `LeAct` counterpart.

The full procedure is collected in Algorithm 1 (main text §3.3).

Table 4: **Mapping from the IWAE bound to the `LeAct` algorithm.** Each formal quantity (App. A.1) corresponds to a concrete pipeline component.

Formal quantity	`LeAct`'s approximation	Algorithm component
Proposal $q_{\mathrm{bwd}}(z \mid x, y)$	Backward decode at θ_{r-1}, redacted $\tilde{\pi}^*$	Backward generation
Per-sample IWAE weight $w_i = e^{\Delta_i}\rho_i\, p_\theta(y \mid x)$	$w_i^{\mathrm{LeAct}} = e^{\Delta_i}$ (ρ_i dropped)	Forward delta scoring
Softmax weight $\widetilde{w}_i \propto e^{\Delta_i}$	Hard top-K on positive $\bar{\Delta}(z; x)$	Top-K selection
Joint loss $\log p_\theta(z, y \mid x)$	SFT on (x, z, y) tuples	Supervised fine-tuning
Ideal action-level cross-entropy	SFT on the oracle policy's textual rendering	EPF surrogate

A.4 Textual Policy Proxy: From Δ to Implementation

The formal definition $\Delta(z;\, x, y) = \log p_\theta(y \mid x, z) - \log p_\theta(y \mid x)$ in Eq. (4) reads $\log p_\theta(y \mid x, z)$ as the decoder's next-token log-likelihood of the action under the student conditioned on z. Our implementation does not score the action token by token. Instead, the model is supervised during SFT to emit a textual policy line of the form "`Action:` a `\n Policy:` $\{a_i{:}p_i\}_i$" alongside its reasoning trace, and the scoring pipeline reads Δ off this line. This appendix bridges the formal Δ used in §3.2 and App. A.1 to the parsed-policy quantity $\widehat{\Delta}$ that our pipeline actually evaluates.

The implementation quantity. At scoring time we parse the textual policy line into a distribution $\hat{\pi}_\theta(\cdot \mid x, z)$ over legal actions and compute the average-delta proxy

$$\widehat{\Delta}(z;\, x) \;=\; -\,\mathrm{KL}\big(\pi^*(\cdot \mid x) \,\|\, \hat{\pi}_\theta(\cdot \mid x, z)\big) \;-\; \big[-\,\mathrm{KL}\big(\pi^*(\cdot \mid x) \,\|\, \hat{\pi}_\theta(\cdot \mid x)\big)\big],$$

where $\hat{\pi}_\theta(\cdot \mid x)$ is averaged over Monte-Carlo forward samples without conditioning on a backward z.

Equivalence to $\bar{\Delta}$ under proxy correctness. Up to the entropy term $H(\pi^*)$ that cancels between the two KLs,

$$\widehat{\Delta}(z;\, x) \;=\; \mathbb{E}_{y\sim\pi^*(\cdot|x)}\big[\log \hat{\pi}_\theta(y \mid x, z) \;-\; \log \hat{\pi}_\theta(y \mid x)\big],$$

equals the formal action-averaged delta only if both the conditional parsed policy and the baseline match the corresponding action distributions, and no clipping changes the KL values. Finite forward sampling, parsing, and clipping introduce additional approximation.

When the proxy diverges from the decoder distribution. The textual policy is a self-report: the model is free to write rounded numbers, to disagree with the action token it just emitted, or to omit minor mass that the decoder would still place on rare actions. A token-level evaluation $\log p_\theta(y \mid x, z)$ via teacher forcing would instead read the next-token logit at the action position and is the literal quantity in Eq. (4). The two coincide only when the textual line faithfully matches the decoder's softmax over the action vocabulary.

Rationale and limits of the proxy. Training and scoring use the same textual policy format, and the selected traces retain oracle-policy supervision. These choices motivate the proxy but do not make it lossless: parsing or calibration errors need not cancel between candidate and baseline, and scoring error can change which traces are selected.

Selection-Fidelity Audit. We quantify the agreement between $\bar{\Delta}$ (the parsed-policy proxy used by the M-step) and a token-level estimator $\Delta_{\mathrm{ll}}(z;\, x) := \log p_\theta(t^*(x) \mid x, z) - \frac{1}{M}\sum_m \log p_\theta(t^*(x) \mid x, z_m^{\mathrm{fwd}})$, where $t^*(x)$ is the canonical policy line `Policy:`$\{a_i : p_i^*\}_i$ and z_m^{fwd} are $M{=}4$ student-decoded forward CoTs scored without backward conditioning. This token-level score measures a canonical policy string and subtracts a mean log-likelihood; it is not the formal action-averaged delta with a log-marginal baseline. The reported diagnostic is whether its top-1 candidate is positive under the parsed-policy score, conditional on an available positive candidate. It does not measure agreement between the two top-1 choices. On a 100K-candidate audit over 16,977 multi-candidate Leduc-13r4s infosets, when Δ_{ll}'s within-infoset top-1 falls on an infoset containing some positive-$\bar{\Delta}$ candidate, that selection is also positive under $\bar{\Delta}$ in 89% of cases (base coldstart); on well-trained `LeAct` iterates this conditional rate reaches 98% (Table 5). Global rank correlation across all candidates is moderate ($\rho \approx 0.65$), reflecting scale disagreement on candidates the M-step never sees; this is a secondary diagnostic rather than a load-bearing fidelity number.

Boundary cases. We clip parsed KL values at 10 (rare deterministic outputs against a mixed π^* would otherwise diverge), and we treat candidates with malformed policy lines as having no information by

Audit subset	N_{inf}	positive retention	positive recall
100K candidates, base coldstart	16,977	0.89	0.80
5K candidates, base coldstart	74	0.93	0.81
5K candidates, best LeAct iterate	74	0.98	0.84

Table 5: **Token-score top-1 candidates are parsed-score positive in 89–98% of eligible infosets.** Selection-Fidelity Audit on Leduc-13r4s, decisive-Nash subset ($h(\pi^*)/\log|A| \leq 0.95$). N_{inf} counts multi-candidate infosets. Positive retention is the fraction of Δ_{ll}-top-1 picks with positive $\bar{\Delta}$, conditional on the infoset containing at least one positive-$\bar{\Delta}$ candidate. Positive recall is $\Pr(\bar{\Delta}>0 \mid \Delta_{\text{ll}}>0)$. Neither metric measures equality of the two top-1 choices.

routing them through a deterministic fallback on the parsed action. Evaluating action-token likelihood scoring in place of the parsed-policy proxy remains future work; the present audit does not establish unchanged training outcomes under that substitution.

B Ablation Studies: Selection, Scoring, and Supervision

Top-K selection rate ablation (Leduc 13r4s). We ablate the selection count $K \in \{1, 2, 4\}$ in the Leduc 13r4s R2 pool with N=8 per infoset and matched hyperparameters (coldstart base, lr $= 5 \times 10^{-6}$, matched training budget). Table 6 reports per-epoch results. K=1 underperforms $K \geq 2$ across all checkpoints, confirming that aggressive top-1 selection discards useful gradient. Between K=2 and K=4 the cells are close: K=2 wins at ep1 (single 0.41 vs 0.61, BoN 0.107 vs 0.110) and the gap is similar in the opposite direction at ep2–3, with neither setting dominating across all six cells. Because the two are comparable on selection quality and K=2 halves the per-round SFT data budget, we standardise on K=2 across the paper (Table 11) for compute efficiency.

Table 6: K=1 **underperforms** $K \geq 2$**;** K=2 **and** K=4 **are comparable.** Top-K selection ablation on the Leduc 13r4s R2 pool (N=8, matched HP). Per-epoch single and BoN exploitability; **bold** = best per metric across all cells. All variants share the same R2 pool, base (Leduc 10r4s joint coldstart), lr $= 5 \times 10^{-6}$, 3 epochs.

	Single ↓			BoN ↓		
K	ep1	ep2	ep3	ep1	ep2	ep3
1	0.7891	1.0900	0.4863	0.1285	0.1121	0.1392
2	0.4125	0.3915	0.4004	0.1069	0.0753	0.1282
4	0.6073	**0.3248**	0.3932	0.1097	0.0563	**0.0518**

Panel C: Within-`LeAct` component ablation on Liar's Dice. A within-`LeAct` ablation isolates the relative contribution of the joint forward–backward coldstart vs. the backward delta scoring stage. With the joint coldstart fixed at 100% infoset coverage, removing the scoring stage stops the policy at exploitability 0.439 (coverage-only, no scoring); restoring the scoring stage brings the policy to 0.019 (R1) and 0.002 (R2). The 219× R2 gain is attributable to selection alone, once the coldstart contribution is held constant.

Table 7: **Backward scoring drives the 219× R2 gain on Liar's Dice; coverage alone stops at 0.439.** Joint coldstart fixed at 100% infoset coverage; only the backward delta scoring stage is toggled.

Component	Variant	Scale	Single	BoN
Backward scoring	Joint coldstart, no scoring	LD	0.439	0.251
	Joint coldstart + scoring (R2)	LD	**0.002**	**0.001**

C Iteration Dynamics

A distinctive feature of `LeAct` is its ability to iterate: the improved model both generates better backward reasoning and serves as a tighter forward scorer. Per-domain headline numbers appear in Table 8; this appendix isolates two questions that table does not address: which E-step variant to run, and when to stop.

E-step variant comparison. We compare three E-step variants from the same R1 base on Leduc 10r4s: *DAgger* (expert backward prompts, model forward scorer), *Joint* (self-generated backward traces, joint SFT), and *REINFORCE* (self-generated backward traces, forward-only SFT). The ranking REINFORCE $>$ JOINT $>$ DAGGER is consistent across metrics. REINFORCE wins because the model's own backward traces are more compatible with its forward reasoning than expert-generated ones, and forward-only SFT avoids introducing noise from the backward task during training.

Iteration diagnostics and reported rounds. We inspect the following diagnostics of progress toward the oracle: (i) close-fraction (KL $<$ 0.1) saturating at 1.0 (Liar's Dice R2 reaches exploitability 0.002 and close-frac 1.0); (ii) single-sample exploitability relative to the prior round's BoN (Leduc 13r4s R2 single 0.092 remains above R1 BoN 0.036); (iii) single-sample KL falling below $\sim$0.002 at 10^9-scale (FHP R2 single KL 0.0019). The reported `LeAct` rounds are R1 for 3-Player Leduc, R2 for Leduc 6r2s/10r4s/13r4s, Liar's Dice, and FHP, and R1 for BuilderBench due to compute. The listed diagnostics do not establish a single prospective stopping rule across settings. In particular, the Leduc 13r4s comparison in (ii) does not show single-sample exploitability below the prior round's BoN.

`LeAct`'s contribution is the closed-loop backward-and-score algorithm itself, not a depth claim about iteration count. With shared coldstart and training states, R1 alone already separates `LeAct` from forward CoT ExIt across every reasoning-game setting (for instance, Leduc 13r4s R1 single 0.097 vs. CoT ExIt R1 0.501; Liar's Dice R1 0.019 vs. 0.671); iteration is a means of tightening once the oracle's distribution is matched, not the source of the imitation gap.

D Additional Results

D.1 Numerical Detail for Fig. 2

Table 8 reports the full numerical breakdown behind Fig. 2 of §5.1, including stds over multi-seed evaluation and the `gpt-5.5` zero-shot anchor at the smallest setting.

Table 8: **Numerical detail behind Fig. 2.** Game rows: exploitability or `NashConv` (lower better); BuilderBench row: cube-placement progress (higher better). BoN uses N=8 for games and N=64 for BuilderBench (single max per task, no across-seed std). **Bold** = best per metric per row. `gpt-5.5` is a frontier zero-shot anchor (Pass@1, medium reasoning effort) reported on the smallest setting only; FHP lives in Table 2a (generalisation-only at this scale).

		gpt-5.5	Single				BoN			
Setting (# infosets)	Metric		Cold.	NoCoT BC	CoT ExIt	`LeAct`	Cold.	NoCoT BC	CoT ExIt	`LeAct`
Leduc 6r2s (4K)	expl. (↓)	0.799	1.351 $\pm$0.226	**0.003** $\pm$0.001	0.087 $\pm$0.018	0.016 $\pm$0.004	0.107 $\pm$0.026	**0.001** $\pm$0.0002	0.029 $\pm$0.007	0.006 $\pm$0.002
Leduc 10r4s (47K)	expl. (↓)	—	1.670 $\pm$0.063	**0.024** $\pm$0.009	0.319 $\pm$0.089	0.037 $\pm$0.024	0.282 $\pm$0.011	0.002 $\pm$0.0003	0.058 $\pm$0.016	**0.0008** $\pm$0.0002
Leduc 13r4s (80K)	expl. (↓)	—	0.687 $\pm$0.068	0.498 $\pm$0.050	0.501 $\pm$0.016	**0.092** $\pm$0.033	0.402 $\pm$0.018	0.334 $\pm$0.036	0.087 $\pm$0.005	**0.022** $\pm$0.004
Liar's Dice (24K)	expl. (↓)	—	0.818 $\pm$0.082	0.018 $\pm$0.003	0.156 $\pm$0.024	**0.002** $\pm$0.001	0.784 $\pm$0.058	0.009 $\pm$0.002	0.043 $\pm$0.007	**0.001** $\pm$0.0002
3-Player Leduc (14K)	`NashConv` (↓)	—	1.482 $\pm$0.103	0.773 $\pm$0.041	0.473 $\pm$0.057	**0.005** $\pm$0.001	0.354 $\pm$0.082	0.770 $\pm$0.024	0.146 $\pm$0.028	**0.003** $\pm$0.001
BuilderBench (26 tasks)	Progress (↑)	—	**0.5448** $\pm$0.028	—	0.5017 $\pm$0.038	0.3527 $\pm$0.068	0.8229	—	0.7006	**0.8853**

Error Bar Calculation. For the five game-setting rows, error bars are standard deviations across 3 inference-rollout seeds of the single trained policy per cell (each seed reseeds the vLLM sampler, producing a distinct sampled policy on the full game tree; std is taken across the 3 resulting exploitability or `NashConv` values). BuilderBench is also single-trained per cell; we substitute an inference-side proxy with a larger seed count. Let $T = 26$ (number of in-domain tasks) and S be the number of inference seeds ($S = 64$ for trained cells; $S = 16$ for the joint coldstart eval). Define $x_{t,s} \in [0, 1]$ as the cube-placement ratio (fraction of cubes placed at target) on task t in seed s. The Single column reports $\bar{\mu} = \frac{1}{T}\sum_t \mu_t$ where $\mu_t = \frac{1}{S}\sum_s x_{t,s}$, with error bar $\sigma_{\text{single}} = \text{std}(\bar{x}_1, \bar{x}_2, \ldots, \bar{x}_S), \quad \bar{x}_s = \frac{1}{T}\sum_t x_{t,s}$, i.e. the per-seed cross-task mean is computed once per inference seed and the std is taken across the S resulting values. The BoN column uses

best-of-N=64, $\max_s x_{t,s}$ per task; this collapses the seed dimension to one value per task, so no across-seed std is defined and we omit error bars there.

D.2 BuilderBench: Partition and SFT-Headroom Diagnostic

BuilderBench serves as the cross-domain probe and the low-recall half of the EPF $\times$ method ablation (§6). The benchmark is a robotics cube-manipulation task (UR5e arm, MuJoCo, 51 tasks from cube-1 to cube-50). The expert oracle is a pool of golden trajectories from three frontier models (GPT-5.2 as a CoT agent, plus Claude Opus 4.6 and Gemini 3 Flash as reflexion agents, first successful episode per model). The model is Qwen3-8B, and the pipeline matches the one used for game domains (BuilderBench coldstart corpus details in App. H.3): joint forward–backward coldstart SFT on the frontier-model trajectories, backward generation on held-out trajectories, forward delta scoring with top-K selection, then SFT.

D.2.1 Heuristic Action-Match Reward

BuilderBench's expert oracle commits to a single JSON action y^* per state (action type, target cube id, and target position). Although a likelihood for this serialized action can be defined, our implementation uses an action-match reward instead of the likelihood score in Eq. (4). We substitute a heuristic action-match reward $r(y;\, y^*)$ that grades partial overlap on the three action attributes:

$$r(y;\, y^*) \;=\; \begin{cases} 1 & \text{action type and cube id matched, } \|\Delta\mathrm{pos}\|_2 \le 1\,\text{cm}, \\ \max\bigl(0,\, 1 - \|\Delta\mathrm{pos}\|_2\bigr) & \text{action type and cube id matched, larger position offset}, \\ 0.1 & \text{action type matched only}, \\ 0 & \text{otherwise}. \end{cases} \tag{18}$$

The candidate's forward delta becomes the with-CoT vs. without-CoT reward gap $\Delta(z;\, x, y^*) = r(y_z;\, y^*) - r(y_\emptyset;\, y^*)$, where y_z and $y_\emptyset$ are the actions the model decodes with and without the candidate CoT z in context. The rest of the M-step pipeline (positive-delta filter, top-K selection) is unchanged.

D.2.2 Cross-Domain Replication of the Poker 2×2

Coldstart anchor. After joint coldstart SFT (3 epochs), the model reaches aggregate pass@64 $= 15/46$ on the 46-task BB-51 denominator (5 context-overflow tasks excluded; partition details in App. H.2). The zero-shot Qwen3-8B base reaches $6/46$.

Method$\times$EPF interaction. Running the full `LeAct` loop (N=8 backward candidates per task, forward delta scoring, top-K selection, 3 SFT epochs) and crossing with the EPF axis gives the following aggregate pass@64 results: $(\texttt{LeAct}, \mathrm{EPF}) = 21/46$, $(\texttt{LeAct}, \text{no-EPF}) = 14/46$, $(\textsc{ExIt} + \textsc{CoT}, \mathrm{EPF}) = 10/46$, and $(\textsc{ExIt} + \textsc{CoT}, \text{no-EPF}) = 2/46$. Only the configuration that combines backward generation with EPF clears the coldstart anchor of $15/46$; the worst cell sits an order of magnitude below the best, separating the four configurations along both axes.

Diagnosis. On this domain, 25% of backward candidates receive positive delta. Among the four evaluated configurations, only `LeAct` with EPF improves aggregate pass@64 over the coldstart anchor. Ablating either the backward pool (CoT ExIt pipeline) or the EPF target costs absolute pass@64, in either combination. This is the same dependence structure as poker (§6), now visible on a non-game domain whose oracle and metric have nothing to do with Nash policies. The positive-delta rate alone does not identify why the other configurations fall below coldstart. The four-cell comparison supports an interaction between the training method and supervision target, rather than a unique causal explanation.

Implication for `LeAct`'s scope. BuilderBench establishes that `LeAct` adds aggregate end-task value when the supervision target is the oracle distribution itself; without EPF the pipeline regresses on this domain. The same domain doubles as a low-recall mechanism probe: the cross-domain 2×2 on EPF $\times$ method (§6) uses BuilderBench as the low-recall half (continuous-valued action arguments rarely repeat across states), and the pass@64 ordering (`LeAct` + EPF $>$ `LeAct` + no-EPF $>$ ExIt + CoT + EPF $>$ ExIt + CoT + no-EPF) shows that the outcome depends on the full method–target configuration; it does not by itself identify the mechanism. BuilderBench therefore contributes both generalisation evidence (§5.2) and mechanism evidence (§6) to the paper's claims.

D.3 Flop Hold'em: Win-Rate Methodology

KL summary. The hold-out KL summary is in Table 2a of the main text (§5.2); this subsection focuses on the chip-outcome (win-rate) analysis that complements it.

Setup. Methodology recapped here for self-containment: paired self-play, model-vs-model (not model-vs-teacher), chip outcomes averaged over $N{=}50{,}000$ paired hands with two seatings ($25{,}000$ each, identical card sequences, alternating Button vs Big Blind to remove positional bias). Outcomes in milli-big-blinds per game (mbb/g; one game = one hand in 2-player poker). 95% confidence intervals from per-hand chip-delta variance ($\mathrm{CI}_{95} = 1.96 \cdot \mathrm{SEM}$). Sampling: temperature 1.0, max-tokens $= 1024$. Both methods compared use expert-policy forcing (matched supervision target).

What the chip-outcome adds beyond KL. `LeAct` and CoT ExIt use matched supervision (both targets are the oracle Nash policy), so the chip-outcome gap isolates the contribution of backward delta scoring on top of forward KL-to-Nash ranking; this is the same controlled ablation as the KL Table 2a, but on play-strength rather than distribution-faithfulness.

Scope of the win-rate matrix. The matrix in Table 2b is restricted to trained-vs-trained comparisons across the three trained methods (NoCoT BC, CoT ExIt, `LeAct`) that share the same supervision target (the `DeepCFR` teacher's expert action). The SFT coldstart is reported in the hold-out KL summary (Table 2a; coldstart single-sample KL 0.0321 vs. trained-method 0.0019–0.0145) but excluded from the chip-outcome matrix because it is not the output of any iteration round.

E Mechanism Diagnostic: Nash-Number Recall in CoT and Delta Scoring

The qualitative redaction in §3.3 prevents the backward generator from copying π^* from its prompt, but trained CoTs may still contain Nash-plausible numbers as a side effect of expert-policy supervision: the M-step internalises π^*, and the same model reused as the backward proposal can re-emit Nash-like values without ever seeing them as input. This appendix records a CoT–policy coherence diagnostic examining a possible explanation of the EPF $\times$ method 2×2 pattern in §6.

E.1 CoT–Policy Coherence Diagnostic

§6 discusses CoT–policy coherence and Nash-number recall as a possible explanation of the 2×2 EPF $\times$ method interaction (Table 3). This subsection records the corresponding descriptive diagnostic.

CoT–policy coherence. ExIt+CoT's SFT target without EPF is an internally self-consistent (CoT, policy) trace: both components are produced by a single forward decode of the current model, so the gradient does not have to reconcile two independently produced signals. `LeAct`'s SFT target without EPF, by contrast, is a hybrid: the CoT was produced by the backward model (which saw π^* qualitatively), while the policy line is the forward model's decoded distribution. The prediction is that per-sample coherence (measured as $1 - D_{\mathrm{KL}}(\pi_{CoT} \,\|\, \pi_{\text{policy line}})$) should be higher for ExIt+CoT merge samples than for `LeAct` merge samples in the no-EPF regime, and inspecting merged SFT data suffices to test it.

Diagnostic (R1 merge files, $\sim$188K lines per cell). We measure $D_{\mathrm{KL}}(\pi_{CoT} \,\|\, \pi_{\text{policy line}})$ on the R1 SFT merge data for all four cells.

EPF row. Cell A's per-sample KL distribution is right-shifted relative to cell C across the bulk of the support. This distributional comparison is descriptive rather than a causal test.

No-EPF row. The surface ordering is reversed (cell B is more coherent than cell D), but inspection shows that the large majority (93.6%) of `LeAct`'s no-EPF backward CoTs restate π^* verbatim, because the backward generator saw the qualitative Nash context and the coldstart model was already near-Nash, so the CoT and the model's decoded policy line trivially agree. This "Nash-recall shortcut" is a degenerate form of coherence: the CoT does not analyse, it echoes, and because the SFT action target in the no-EPF regime is the model's own decode rather than π^*, the gradient may reinforce the model's residual Nash errors instead of correcting them. The symmetric flip is cell C (ExIt+CoT with EPF), where replacing the self-decoded (CoT, policy) pair with π^* can break the trace's internal consistency.

Synthesis. These observations are consistent with an interaction between CoT–policy coherence and supervision-target choice. They do not establish that this interaction causes the performance ordering of cells A–D.

F Oracle Construction for Large Games

We document oracle construction per game scale: tabular `CFR`/`CFR+` via `OpenSpiel`'s tabular enumeration for games up to ∼80K infosets (App. F.1), and a streaming `DeepCFR` enumeration for Flop Hold'em where tabular enumeration exhausts memory (App. F.2).

F.1 Tabular CFR for Small-to-Medium Games

For the enumerable game settings (Leduc 6r2s through 13r4s, Liar's Dice, and 3-player Leduc), we use vanilla `CFR` or `CFR+` to construct oracle policies and export their full information-set policy tables through `OpenSpiel`'s `to_tabular()` function. We store the resulting $|\mathcal{X}| \times |\mathcal{A}|$ policy matrix on disk. Full tabular enumeration does not certify an exact Nash equilibrium. Finite-iteration solver policies may retain nonzero exploitability, and the standard two-player zero-sum CFR convergence guarantee does not extend to general three-player games. This is feasible up to ∼80,000 information sets on a standard 256 GB RAM node; larger games cause out-of-memory failures during tabular enumeration.

F.2 DeepCFR Streaming Enumeration for Flop Hold'em Poker

For Flop Hold'em Poker (FHP), the standard tabular enumeration via `to_tabular()` exhausts available RAM at 1.5 TB MaxRSS before completing, making full tabular policy construction infeasible on current hardware. We developed a trajectory-based streaming enumeration method that avoids materialising the full game tree in memory.

Method. We run DeepCFR [5] on FHP using OpenSpiel's PyTorch implementation (`open_spiel.python.pytorch.deep_cfr`), which trains neural network value/strategy approximators from sampled game trajectories. The game is specified via OpenSpiel's `universal_poker` configuration matching the Brown et al. FHP benchmark (2-player limit hold'em, 2 betting rounds, blinds 50/100, raise size 100, 200 BB stacks). Rather than tabularising the final strategy network by querying it at all information sets simultaneously, we enumerate infosets *incrementally*: during each CFR traversal we record the (infoset, policy) pair encountered in the trajectory, deduplicate across traversals via a hash table, and write policy entries to disk as they are first seen. This streaming approach decouples memory consumption from game-tree size, requiring only $O(|\text{trajectory depth}|)$ live memory per traversal rather than $O(|\mathcal{X}|)$.

Configuration. We run $1{,}000$ outer DeepCFR iterations on OpenSpiel's PyTorch implementation, then draw 10^6 on-policy trajectories from the converged policy for streaming enumeration; end-to-end wall-clock is approximately 22 minutes on a single CPU node and produces a ∼570 MB jsonl teacher table that we use throughout the paper.

Results. The streaming enumeration produced $2{,}343{,}732$ unique infoset policies, substantially more than any other game in our experiments (see Table 9). Policy coverage converges monotonically along policy-induced trajectories: after the first 50K traversals, coverage stalls around 60% of newly-encountered infosets; ∼500K traversals already saturate at >99% in-support coverage, and we extend the run to 10^6 trajectories for margin. The full FHP game tree (which contains $\sim 10^9$ unique infosets [5] including off-policy subtrees not visited by external-sampling MCCFR) is therefore larger than the teacher's $2{,}343{,}732$-infoset table. This affects only out-of-table queries when the teacher would itself be queried as a self-play opponent (see App. D.3); KL evaluation is computed exclusively on the held-out subset of the teacher's $2{,}343{,}732$ infosets and is therefore unaffected.

Limitation. DeepCFR provides an *approximate* Nash policy rather than an exact one; the approximation error decreases with more traversals but does not reach zero. For the purposes of `LeAct` oracle supervision, we treat the converged DeepCFR policy as a high-quality proxy for π^*, analogous to how approximate CFR-family solvers (continual re-solving with neural counterfactual value networks [31], MCCFR-blueprint [4]) are used in large-scale poker AI.

Table 9: **Tabular CFR scales to 80K infosets; FHP requires streaming** `DeepCFR`**.** Tabular and neural solver outputs are used as oracle policies; tabular representation alone does not certify zero exploitability.

Game	# Infosets	Oracle	Method
Leduc 6r2s	4,032	Tabular `CFR`	`to_tabular()`
Leduc 10r4s	47,040	Tabular CFR	`to_tabular()`
Leduc 13r4s	79,872	Tabular CFR	`to_tabular()`
Liar's Dice	24,576	Tabular CFR	`to_tabular()`
3-Player Leduc	13,878	Tabular CFR	`to_tabular()`
Flop Hold'em Poker	2,343,732	DeepCFR (approx.)	Streaming enumeration

G Prompt Format and Representative Examples

We show the prompt interface and one matched pair of backward-generated traces. The pair illustrates what the forward-delta filter rewards and rejects; it is not used as quantitative evidence.

G.1 Prompt and Output Format

The forward prompt provides the information state and legal actions and asks the student for a reasoning trace followed by a policy. The backward prompt uses the same state but reveals only a qualitative summary of the expert action and asks for reasoning alone. Exact oracle probabilities are not shown to the backward generator.

Forward prompt and output schema

```
user: Information state: [Round 2][Player: 0][Private: 7h] [Public:
3d][Sequences: cb]
Legal actions: [Fold, Call, Raise]
What is your action?
assistant: <think>...reasoning...</think>
Policy: {Fold: p_f, Call: p_c, Raise: p_r}
```

Backward prompt and output schema

```
user: Information state: [Round 2][Player: 0][Private: 7h] [Public:
3d][Sequences: cb]
Legal actions: [Fold, Call, Raise]
Optimal strategy description: Fold often; call sometimes; raise never.
Explain the reasoning without quoting probabilities.
assistant: <think>...reasoning only...</think>
```

The M-step target $\pi^*(\cdot \mid x)$ is supplied separately and is never parsed from the backward output. A candidate survives only when conditioning the forward model on it increases the per-token log-probability of the expert policy relative to the student's self-generated reasoning baseline (App. A.4).

G.2 Matched Positive- and Negative-Delta Traces

For the state above, the expert qualitatively prefers folding, mixes in calls, and avoids raising. The following positive-delta trace connects observable features to that action structure:

Positive-delta candidate

<think> The private card does not pair the board, and the opponent's bet after a check is evidence of a stronger range. Calling retains some value against bluffs and weaker unpaired hands, but raising is unattractive because stronger hands continue and weaker hands often fold. The action should therefore lean toward folding, mix in calls, and rarely raise. </think>

By contrast, a circular restatement of the revealed action contains no state-dependent explanation and receives negligible or negative delta:

Non-positive-delta candidate

`<think>` The optimal strategy is to fold most of the time, call occasionally, and avoid raising because this is the optimal strategy. `</think>`

These examples isolate the intended distinction: the filter rewards reasoning that makes the expert action more predictable from the state, not prose that merely repeats the action description.

H Additional Experimental Details

H.1 Domain Summary

Table 10: **Domain summary with infoset count, oracle, and metric.** Leduc "NrMs" is N ranks $\times$ M suits per rank (so 10r4s = 40-card deck); Liar's Dice uses the `1d6f` variant; 3-Player Leduc uses `3r2s` parameters. †FHP's infoset count is the full-tree estimate [5]; the streaming `DeepCFR` teacher covers 2,343,732 unique infosets along policy-induced trajectories (App. F).

Domain	# Info Sets	Players	Expert Oracle	Evaluation Metric
Leduc 6r2s	4,032	2	CFR Nash	Exploitability
Leduc 10r4s	47,040	2	CFR Nash	Exploitability
Leduc 13r4s	79,872	2	CFR Nash	Exploitability
Liar's Dice (1d6f)	24,576	2	CFR Nash	Exploitability
3-Player Leduc 3r2s	13,878	3	CFR Nash	NashConv
Flop Hold'em (FHP)†	$\sim 10^9$	2	DeepCFR	KL to teacher
BuilderBench	51 tasks	1	Frontier-model trajectories	Pass@K, cube placement

H.2 BuilderBench Task Partition

We evaluate on 46 of the 51 BuilderBench tasks (5 are excluded for exceeding the 8,192-token context window), split by trajectory-pool success. **In-domain** (26 tasks): at least one of the three frontier oracles (Claude Opus 4.6, Gemini 3 Flash, GPT-5.2) produced a fully-successful episode in the trajectory pool, so the SFT data contains a golden trace. **OOD** (20 tasks): the best-of-3 frontier-model trajectory pool produced no successful trajectory, so any progress on these tasks is net-new training signal beyond the teacher pool. Per-oracle solve counts: Claude Opus 4.6 24/51; Gemini 3 Flash 24/51; GPT-5.2 15/51; any-oracle union 26/51. Solve-status lists are fixed by oracle data collection and do not change across `LeAct` iterations.

H.3 Coldstart Construction

The coldstart phase fine-tunes Qwen3-8B on a domain-specific format-priming corpus. Three coldstart variants are used across our experiments: NoCoT (state $\rightarrow$ action only), CoT (state $\rightarrow$ forward CoT $\rightarrow$ action), and joint forward+backward (one SFT mix containing both forward training pairs and backward state, action $\rightarrow$ CoT pairs). The corpus source depends on what the oracle provides.

Leduc, Liar's Dice, 3-player Leduc. Frontier-LLM sub-agents (Claude, Gemini, GPT-5) are prompted to produce candidate (CoT, action) pairs at every infoset; the action target is then replaced by the `CFR` solver's oracle policy before SFT. For joint variants, the backward direction is prompted with the oracle's *qualitative* action summary (e.g., "mostly fold with occasional calls") rather than exact probabilities, the same redaction protocol used in `LeAct`'s E-step (§3.3) to discourage Nash-number recall. Coldstart corpus sizes per setting: 4,032 examples for Leduc 6r2s (NoCoT, full coverage of all 4,032 infosets), 2,800 for Leduc 10r4s joint (1,400 forward + 1,400 backward, a sampled subset of the 47,040 infosets to keep frontier-LLM API generation cost bounded), $\approx$49,000 for Liar's Dice joint (full coverage of all 24,576 infosets in both directions), and 27,756 for 3-player Leduc joint (13,878 forward + 13,878 backward, full coverage). Leduc 13r4s reuses the Leduc

10r4s joint-coldstart checkpoint rather than running a fresh coldstart, isolating the scaling effect from coldstart variance.

Flop Hold'em. Exact `CFR` is intractable, so coldstart CoTs are generated off-GPU by frontier-model sub-agents (Claude Opus 4.6 / Sonnet 4.6 / Haiku 4.5) prompted with the infoset description and the `DeepCFR` teacher's policy. Each sub-agent emits a ⟨reasoning, action, policy⟩ triple in a single forward pass; the reasoning block embeds both the forward justification and the backward explanation, so no separate backward direction is generated. Outputs are validated for hole-card faithfulness (both hole cards named or referenced by exact-card token) and flop-rank coverage before inclusion, guarding against Leduc-style framing. The resulting corpus contains $1{,}510$ NoCoT and $1{,}510$ CoT examples ($3{,}020$ merged for joint).

BuilderBench. The coldstart corpus is the frontier-model trajectory pool itself: the first successful episode per (task, model) pair from three frontier agents (GPT-5.2 as a CoT-and-act agent, Claude Opus 4.6 and Gemini 3 Flash as reflexion agents). The joint forward+backward SFT mix contains $14{,}144$ examples across the 51 BuilderBench tasks; the resulting checkpoint is the "Coldstart" anchor discussed in App. D.2.2.

All coldstart SFT runs use 3 epochs at lr 1×10^{-4} for poker domains and BuilderBench, 2×10^{-5} for FHP; remaining hyperparameters follow Table 11.

H.4 Hyperparameters

Table 11 summarises the training hyperparameters used across all domains and methods. All experiments use Qwen3-8B as the base model, LLaMA-Factory for SFT with FSDP `full_shard` on 2–4 NVIDIA H200 GPUs, and vLLM for inference. Sampling parameters: temperature $T = 1.0$, top-$p = 0.95$, top-k disabled, frequency penalty 0, repetition penalty 1 (vLLM defaults), max generation $4{,}096$ tokens for game domains and $8{,}192$ tokens for BuilderBench.

Table 11: **Training hyperparameters per experimental condition.** Learning rates and epoch counts are scale-tuned: small Leduc variants tolerate aggressive schedules under full coverage, while 13r4s and FHP use longer warmup to avoid collapse on a wider distribution. We standardise on top-2 selection across the paper; N=8–16 for backward sampling.

Setting	Method	Epochs	LR	Selection K	Gen. N	GPUs
Leduc 6r2s	ExIt / `LeAct`	1–3	2e-6 – 5e-6	top-2	8–32	2
Leduc 10r4s	NoCoT BC	1	2e-6	top-2	8–32	2
Leduc 10r4s	`LeAct`	1–3	2e-6 – 5e-6	top-2	8	2–4
Leduc 13r4s	NoCoT BC	3	1e-5	top-2	8	4
Leduc 13r4s	`LeAct`	3	5e-6	top-2	8–16	4–8
Liar's Dice	`LeAct`	3	2e-6	top-2	8	2
3-Player Leduc	`LeAct`	3	2e-6	top-2	8	2
FHP ($\sim10^9$)	`LeAct` (`DeepCFR`)	3	5e-5	top-2	8	8
BuilderBench	`LeAct`	1–3	1e-5	top-2	8–32	4

All experiments share the following defaults: AdamW ($\beta_1 = 0.9$, $\beta_2 = 0.999$, weight decay 0.01) with cosine LR decay and a 3% linear warmup; max sequence length 2048 tokens for game domains and 8192 for BuilderBench; per-device batch 4–8, giving an effective batch of 16–64 depending on GPU count; bf16 mixed precision under PyTorch FSDP `full_shard` with `Qwen3DecoderLayer` wrapping. Each headline cell is trained once per (method, setting) and evaluated under 3 inference seeds; reported single-sample / BoN metrics are best-across-epochs on the full game tree, with std across the 3 evaluations reported in Table 8.

H.5 E-Step Load-Bearing Ablation Study

Two ablations test whether backward-delta selection contributes signal beyond what coverage alone provides; both share the production training recipe summarised above.

Random vs forward-delta ranking (K=1 sharpening). We compare random ranking and forward-delta ranking on the same `LeAct` R2 backward pool at Leduc 10r4s, sharpened to K=1 per infoset (within-pool IoU ≈ 0.13 instead of ≈ 0.50 at K=4) so that each arm commits to a single CoT per infoset and exposes the within-positive ranking signal. Both arms train from the same coldstart

base on the same R2 backward pool with hyperparameters held matched; the comparison is between selection rules, not between training pipelines. Reported metrics (Fig. 3a) are full-tree single-sample and BoN-of-8 exploitability on all 47K Leduc 10r4s infosets at the best epoch in each arm.

Recall-stratified delta on backward pools. On the `LeAct` R2 backward pools at Leduc 10r4s (N=374,272 candidates) and Leduc 13r4s (N=638,976 candidates; N=393,512 after filtering trivial-policy states), we compute mean positive forward delta partitioned by whether each candidate CoT contains at least one verbatim Nash probability. Stratification is on the full unfiltered candidate pool (not on selected CoTs), so the partition spans the entire pre-selection distribution; per-stratum positive-delta means are reported in Fig. 3b. Verbatim detection uses exact substring matching against the per-state π^* probabilities at three-decimal resolution.

H.6 Compute Budget

Table 12 estimates the GPU-hours for each phase of the experimental pipeline. Generation and scoring are the dominant costs.

Table 12: **Per-phase GPU-hour estimates on NVIDIA H200 (141GB).** Generation and scoring scale with infoset count and dominate at small to mid scales; SFT becomes the largest single phase only at FHP. Units suppressed in cells; FHP figures are observed R1 wall-times. † FHP coldstart CoT was generated off-GPU by frontier-model agents (Claude Opus 4.6 / Sonnet 4.6 / Haiku 4.5); the <1 figure is the action-only forward pass over the coldstart trace pool.

Phase	Leduc 10r4s	Leduc 13r4s	Liar's Dice	FHP
`CFR` solver (CPU)	0.5h	2h	0.5h	`DeepCFR` (pre-trained)
Coldstart data gen	4	8	4	$<1^{\dagger}$
Coldstart SFT	2	4	2	16
ExIt generation (N=8)	8	16	8	10
ExIt SFT (per round)	2	8	2	16
`LeAct` backward gen (N=8)	16	32	8	22
`LeAct` forward scoring	16	32	8	8
`LeAct` SFT (per round)	4	16	4	30
Evaluation (per checkpoint)	2	4	2	4
Total per `LeAct` round	∼38	∼84	∼22	∼64

The total compute for the reported experiments (including all iterations, ablations, and baselines) is approximately 1,500 GPU-hours on H200s. The full research project, including failed experiments and hyperparameter searches, consumed approximately 5,000 GPU-hours.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Abstract and introduction claims are supported by Tables 8 and 2 in Sections 5–6; the stated scope is action-oracle domains, evaluated on six imperfect-information games and a robotics cube-stacking benchmark.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Section 7 discusses the central scope limitation: all experiments share a single 8B student to keep cross-domain comparisons compute-controlled, leaving the interaction between `LeAct` and frontier-scale students as a direct extension.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper is empirical and presents no formal theorems. The IWAE-derived delta-scoring objective (Equation 4, full derivation in Appendix A.1) is a definition and approximation, not a theorem requiring proof.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Section 4 specifies the base model (Qwen3-8B), domains, oracles, evaluation metrics, and the `LeAct` algorithm (Algorithm 1); training hyperparameters are in Table 11. Game environments use OpenSpiel.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We will release the complete codebase (generation, scoring, merging, evaluation, and training scripts) with reproduction instructions. Game environments use OpenSpiel (Apache 2.0); the base model Qwen3-8B is publicly available; training data is synthetic and reproducible from the released code.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: Section 4 and Appendix H.4 (Table 11) cover the training pipeline (coldstart, CoT ExIt, `LeAct`), per-domain hyperparameters, optimizer (AdamW with cosine schedule and linear warmup), FSDP configuration, and evaluation metrics.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

Justification: Table 8 reports standard deviations across three inference-rollout seeds of one trained policy per (method, setting); these error bars do not measure variability across independent training runs. Table 2b reports FHP chip outcomes with ± standard error. Methodology is described in Appendix D.1.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Hardware (NVIDIA H200, FSDP) and software (LLaMA-Factory for SFT, vLLM for inference) are specified in Section 4; per-phase GPU-hours and total compute are in Appendix H.4.

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Training uses synthetic data from game-theoretic solvers and frontier-LLM trajectories; no human subjects, personal data, or sensitive content is involved. The base model (Qwen3-8B) and game frameworks (OpenSpiel) are publicly available.

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The primary impact is positive: enabling LLMs to learn reasoning from existing expert systems (planners, theorem provers, robotic-control libraries) without additional human annotation (Section 7). Trained models are domain-specific (poker variants and a cube-manipulation simulator) with no general-purpose capability uplift over the Qwen3-8B base.

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Trained models are domain-specific (poker variants and a cube-stacking simulator) with no general-purpose capabilities beyond the narrow tasks they are trained on; the base model Qwen3-8B is publicly available and our fine-tuning does not add general capabilities.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All assets are cited at first mention with permissive licenses: OpenSpiel [20] (Apache 2.0), Qwen3-8B [34] (Apache 2.0), LLaMA-Factory and vLLM (Apache 2.0), `DeepCFR` [5] for the FHP teacher, BuilderBench [8] (MuJoCo, Apache 2.0); Leduc poker follows Southey et al. [43].

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

Justification: We release the `LeAct` codebase (generation, scoring, merging, evaluation, training configurations) and pre-computed Nash policy tables, with reproduction scripts for the main experiments.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: No crowdsourcing or human subjects; all training data is synthetic (CFR/DeepCFR solvers and language model outputs).

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve research with human subjects.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are the central subject. Qwen3-8B is the experimental student fine-tuned end-to-end under `LeAct`. For BuilderBench, frontier LLMs (Claude Opus 4.6, Gemini 3 Flash, GPT-5.2) serve as the action oracle through their trajectory pool (Appendix H.3). Full description in Sections 3–4.