Self-Play Search Distillation for Large Language Model Reasoning

Lorenzo Molfetta[1] **Wai-Chung Kwan**[2] **Giacomo Frisoni**[1]
Luca Ragazzi[1] **Gianluca Moro**[1] **Pasquale Minervini**[2,3]
[1]University of Bologna [2]University of Edinburgh [3]Miniml.AI
{lorenzo.molfetta, giacomo.frisoni, l.ragazzi, gianluca.moro}@unibo.it
{wkwan, p.minervini}@ed.ac.uk

Abstract

Reasoning post-training for large language models is constrained by the quality and availability of human-annotated data. Human supervision remains scarce at scale, and even rich collections anchor the student to human-level intermediate decisions rather than superhuman ones. Synthetic data can, in principle, ease the availability bottleneck, but controlled reasoning-oriented simulation settings for generating such data remain limited. Recent work shows that zero-shot reasoning can improve when models are trained in simulated settings that expose search-like structure, especially executable board-game environments where self-play produces delayed consequences, exact verification, and strong search teachers. However, most such pipelines keep the learner inside the simulation loop, so data quality remains entangled with the student policy itself. We propose *self-play search distillation (SPSD)*, an offline framework that uses external search experts in executable environments to generate verifiable decisions and search traces, then distils them into language supervision for a separate student model. Furthermore, we introduce SPSD-CoT, a trace-linearization format that converts searched alternatives, the opponent's replies, and replayed terminal evidence into compact reasoning traces. The main study trains Qwen3-4B on five source games and nine controlled variants per source game, then evaluates transfer on external reasoning benchmarks and 15 held-out evaluation games. The reshaped evaluation tests whether SPSD transfers reasoning from five source games and nine variants per source, whether transfer grows with game coverage, whether SPSD-CoT adds value beyond final-action imitation, and whether trained reasoning traces become more MCTS-shaped on held-out games and external reasoning benchmarks.

1 Introduction

Large language models often struggle when reasoning requires sustained state tracking, comparison among plausible alternatives, or decisions whose consequences only become clear several steps later; these failures appear in planning benchmarks, world-state tracking tasks, and other settings where correctness depends on preserving structured intermediate state across several steps (Li et al., 2023; Nanda et al., 2023; Valmeekam et al., 2023a,b; Zeng et al., 2025). Improving this behaviour, therefore, requires supervision beyond final-answer imitation. The training signal should expose good intermediate decisions: how alternatives are compared, how state is preserved, and how delayed consequences are handled. Human demonstrations remain the most reliable source of such supervision, but they are expensive, difficult to scale, and bounded by annotator skill (Ouyang et al., 2022; Wang et al., 2023).

A cheaper alternative is to let models generate part of their own training signal, for example, by producing tasks, opponents, preferences, or solution traces. However, when the same model family supplies both the supervision and the learner being optimised, the resulting data can preserve the learner's current errors or biases (Chen et al., 2024; Liu et al., 2025a; Zhao et al., 2025). To isolate the effect of the supervision itself, the data generator should be independent of the language model being trained, and its outputs should be measurable and verifiable. Executable board games provide this control: they expose exact states, legal actions, and terminal outcomes; their difficulty can be varied through rules, openings, board sizes, and opponents; and they support strong search-based experts whose quality can be improved independently of the language model.

Expert Iteration (ExIt) formalises this source of supervision as a loop between a slow planning

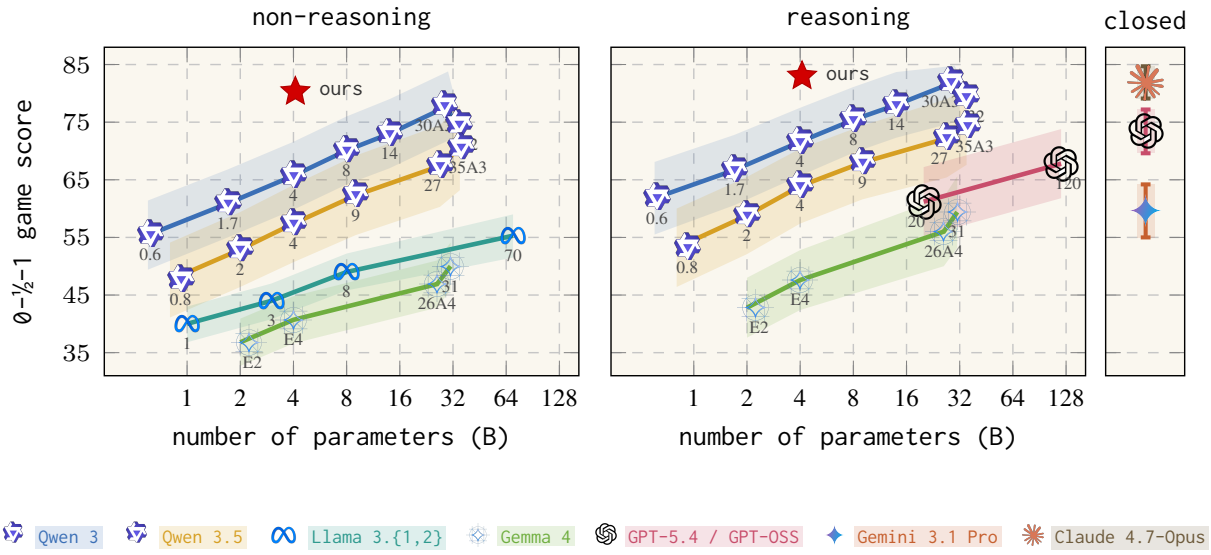


Figure 1: **Model performance on held-out puzzle games.** Win-rate estimates with 95% confidence intervals are grouped by open-weight non-reasoning, open-weight reasoning, and closed-source models.

expert and a fast apprentice that generalises the plans (Anthony et al., 2017; Anthony, 2021). Self-play systems such as AlphaGo (Silver et al., 2016), AlphaZero (Silver et al., 2017a), MuZero (Schrittwieser et al., 2020), and EfficientZero (Ye et al., 2021) can be viewed as prominent MCTS-based instances of this pattern: search improves a policy locally, and learned components make later search stronger. For our purposes, their value is not game performance itself, but the decision data produced by search. A final expert move gives an action label, but it hides the rejected alternatives, threats, and delayed consequences that made the move preferable. Recent work on procedure cloning and search-based supervision suggests that such intermediate decision structure can itself be a useful training target (Yang et al., 2022; Gandhi et al., 2024; Kim et al., 2025).

We test whether exposing this search structure improves LLM post-training in an offline distillation setting. For each executable game environment, we train or select an ExIt-trained planning expert, freeze its checkpoint, and export both its final decisions and compact search traces to supervise a separate language model. Our source corpus contains five selected game families—Connect4, Tic-Tac-Chess, 3 6 9, Domineering, and First Attack—with nine controlled variants per family, yielding 50 trainable environments. The language model student never acts in the simulator during post-training—it learns only from rendered conversations grounded in verified game states. This separation lets us ask whether transfer comes only from stronger final actions, or also from exposing searched alternatives, opponent replies, and replayed terminal evidence.

Figure 1 shows that even strong LLMs remain brittle on held-out board-game play, motivating board games as a compact stress test for state tracking and branch comparison. Figure 2 summarises the pipeline from simulator-grounded search to language supervision. We make three contributions.

- We propose *self-play search distillation (SPSD)*, an offline data-generation framework in which ExIt-trained planning experts generate verifiable state-action and trace supervision while the language model remains a downstream student, decoupling data generation from student optimisation.
- We introduce *SPSD-CoT* (Sections 4.3 and 4.4), an MCTS trace-linearization procedure that turns searched alternatives, opponent replies, and replayed terminal evidence into language supervision while keeping the expert's selected action separate from the continuations used to justify it.
- We evaluate five research questions covering transfer, generalization beyond memorization, MCTS trace supervision, MCTS-depth distillation, and reasoning-dynamics alignment against SPIRAL-style self-play baselines.

2 Related Work

Game reasoning. Game-based reasoning work uses executable environments both to measure

model competence and to generate supervision. ChessBench (Feng et al., 2023) and related game-evaluation studies (Zhang et al., 2025; Karvonen, 2024) test whether language models can maintain board state, follow legal moves, and reason about delayed consequences. Other benchmarks extend this analysis to strategic world-state acquisition and longer interactive settings (Ruoss et al., 2024; Wang et al., 2025). A parallel line treats games as training environments rather than only as tests (Duan et al., 2024; Hu et al., 2025). Follow-up systems enlarge that agenda with broader game-based transfer, rule-grounded environment synthesis, and planning-centric agents (Shi et al., 2025; Guertler et al., 2025; Guo et al., 2024; Schultz et al., 2025). SPIRAL (Liu et al., 2025a) is the clearest example here of learner-coupled self-play for reasoning transfer, while adjacent work studies code self-play and explicit planning agents (Bachrach et al., 2025). However, these works mostly emphasise online improvement inside the learner, leaving offline distillation from a separate search teacher less directly characterised.

Distilling symbolic and search structure into LLMs. Structured synthetic reasoning work seeks supervision richer than answer imitation. One branch replaces human annotation with formal tasks, verifiers, and synthetic curricula so that correctness can be checked automatically. This includes large-scale verified mathematics and puzzle-generation pipelines (Trinh et al., 2024; Chen et al., 2025), as well as self-generated curriculum systems such as Absolute Zero (Zhao et al., 2025). Related work adds tool-mediated verification and automated filtering of synthetic reasoning traces (Wong et al., 2025; Stojanovski et al., 2025). Another branch treats the reasoning process itself as the supervision target: chain-of-thought prompting (Wei et al., 2022), Tree of Thoughts (Yao et al., 2023), and ASTRO (Kim et al., 2025) all expose intermediate search or deliberation structure rather than only final answers. Variants of this idea study procedure cloning, search-dynamics bootstrapping, and related trajectory-level supervision (Yang et al., 2022; Gandhi et al., 2024; Hao et al., 2025; Wu et al., 2025). Related results in algorithm distillation, world-representation learning, and planning analysis suggest that structured latent procedures can transfer beyond the source domain (Laskin et al., 2023; Li et al., 2023; Nanda et al., 2023; Valmeekam et al., 2023a,b). However, these methods usually teach the model to imitate an internal reasoning process, whereas our setting distills simulator-grounded evidence attached to a fixed expert action.

Self-play and RL for LLMs. Modern LLM post-training increasingly relies on scalable feedback rather than purely human-written demonstrations. RLHF-style alignment pipelines established the basic recipe (Christiano et al., 2017; Ouyang et al., 2022; Wang et al., 2023). DPO (Rafailov et al., 2023), classical distillation (Hinton et al., 2015), and self-improvement setups (Shao et al., 2024) then reduced reliance on costly online preference collection. Reasoning-oriented extensions add process rewards and verifiable practice signals (Singh et al., 2024; Liu et al., 2025b; Zeng et al., 2025). Reinforcement-based reasoning pipelines also explore larger self-play or self-generated curricula (Chen et al., 2024). SPIRAL (Liu et al., 2025a) belongs to that broader movement but keeps the learner inside the self-play loop. In parallel, executable-environment frameworks and open-ended curriculum generators make automatically scored supervision controllable at scale (Bellemare et al., 2013; Côté et al., 2018; Cobbe et al., 2020; 'Eric Piette et al., 2020). Related benchmark and curriculum work extends that perspective to procedurally generated tasks and adversarial environment construction (Lanctot et al., 2019; Wang et al., 2019; Dennis et al., 2020; Parker-Holder et al., 2022). However, this literature usually studies how learners improve inside their own feedback loop, not how a separate self-play search process can be used purely as an offline data generator.

3 Background

Expert Iteration. Expert Iteration (ExIt) decomposes reinforcement learning into two coupled operations: an *expert* performs slow online planning at particular states, and an *apprentice* learns a fast policy or value approximation that generalises those plans to new states (Anthony et al., 2017, 2019; Anthony, 2021). The next round of expert planning is then guided by the apprentice, for example through policy priors, value estimates, rollout policies, or learned dynamics. This separates local policy improvement from global generalisation. It also connects ExIt to approximate modified policy iteration: planning supplies an approximate greedy improvement step, while supervised fitting supplies the approximate evaluation/generalisation step over

the states that were searched (Anthony et al., 2019; Anthony, 2021). Let F_{ϕ_k} denote the apprentice available at ExIt iteration k, and let $\mathcal{E}$ denote the simulator or model used by planning. From a state s, the expert planner returns a search-improved policy, optional value, and retained planning evidence:

$$\mathrm{Plan}(s; F_{\phi_k}, \mathcal{E}) = \big(\pi_k^E(\cdot \mid s), v_k^E(s), \mathcal{B}_k^E(s)\big),$$

where π_k^E is the search-improved policy, v_k^E is an optional value estimate, and $\mathcal{B}_k^E$ is retained planning evidence. The apprentice is then trained by supervised learning to match the expert policy target and, when a value head is used, a corresponding value target $\hat{v}_k(s)$:

$$\begin{aligned}\mathcal{L}_k(\phi) = \sum_{s \in \mathcal{D}_k} \Big[& \ell_\pi\big(F_\phi^\pi(s), \pi_k^E(\cdot \mid s)\big) \\ & + \lambda_v \ell_v\big(F_\phi^v(s), \hat{v}_k(s)\big)\Big], \\ \phi_{k+1} \leftarrow \arg\min_\phi & \mathcal{L}_k(\phi).\end{aligned} \tag{1}$$

Here $\mathcal{D}_k$ is the set of states searched at iteration k, ℓ_π and ℓ_v are supervised policy and value losses, and λ_v weights the value term. The target $\hat{v}_k(s)$ may be the planner's value estimate, an observed return, or another bootstrapped target, depending on the ExIt instance. Equation 1 is the supervised apprentice-fitting step that later lets the search expert become a compact data generator. The original ExIt formulation from Anthony et al. (2017) used tree search as the expert and a neural network as the apprentice, while later ExIt variants in the same thesis instantiate the planner with both MCTS and policy-gradient search (Anthony et al., 2019; Anthony, 2021). Our language model is not the ExIt apprentice—instead, we first train or select compact ExIt-trained planning experts inside games, freeze them, and use their exported records as offline supervision for a downstream language-model student.

MCTS-based ExIt instances. AlphaGo (Silver et al., 2016) and AlphaGo Zero (Silver et al., 2017b) established the MCTS version of this expert/apprentice pattern at scale: search improves move selection in self-play, and neural policy/value networks make later searches stronger. AlphaZero (Silver et al., 2017a) generalised the same MCTS-guided self-play recipe to Chess and Shogi. MuZero (Schrittwieser et al., 2020) is am ExIt algorithm in which the expert planner searches through a learned latent dynamics model rather than through a hand-coded transition model exposed to the network. EfficientZero (Ye et al., 2021) keeps the same ExIt contract but adds sample-efficiency mechanisms around the MuZero latent planning. In this work, MuZero and EfficientZero are two ways to implement the ExIt planner in Section 3: representation and dynamics models provide the planning substrate, and policy/value heads provide the apprentice guidance used by MCTS. At a search node s, let $Q(s,a)$ be the current action-value estimate, $P(s,a)$ the policy prior assigned when the node is expanded, and $N(s,a)$ the number of simulations that selected action a. For the MuZero/EfficientZero experts used in our implementation, the MCTS tree policy follows the standard PUCT form. Writing $N_s = \sum_b N(s,b)$ for the sibling visit count at node s, the next action in a simulation is:

$$a_k = \arg\max_a \left[Q(s,a) + P(s,a)\, U(s,a)\right], \tag{2}$$

$$U(s,a) = \frac{\sqrt{N_s}}{1 + N(s,a)} \left(c_1 + \log \frac{N_s + c_2 + 1}{c_2}\right).$$

Here c_1 and c_2 are exploration constants, and b ranges over sibling actions available at node s. Equation 2 turns apprentice priors and value estimates into an expert tree policy. After a fixed simulation budget, the root visit counts define the improved policy target. Writing T_{mcts} for the MCTS temperature used at the root, this target is:

$$\pi(a \mid s) = \frac{N(s,a)^{1/T_{\mathrm{mcts}}}}{\sum_{b \in \mathcal{A}(s)} N(s,b)^{1/T_{\mathrm{mcts}}}}. \tag{3}$$

The resulting distribution in Equation 3 is the search-improved ExIt policy exported from the game expert. It determines how search allocates support across legal actions, while the root v and retained continuations identify the evidence that becomes available to the language trace renderer. Follow-up work improved this search-learning substrate through stochastic or sampled policy improvement (Antonoglou et al., 2022; Danihelka et al., 2022), reanalysis, and sample-efficiency mechanisms (Schrittwieser et al., 2021a,b; Ye et al., 2021; Wang et al., 2024; Mei et al., 2023). Other extensions adapt the same recipe to regularized or hybrid policy optimization, large action spaces, multi-agent games, and world models (Grill et al., 2020; Hessel et al., 2021; Hubert et al., 2021a,b; Hao et al., 2024; Schmid et al., 2023; Ozair et al., 2021; Luo et al., 2025; Pu et al., 2025b,a). Open implementations such as LightZero (Niu et al., 2023)

make these agents practical data generators rather than only benchmark systems. These developments make ExIt search useful here not as the object of study, but as a compact generator of stronger-than-student trajectories and auditable search traces.

SPIRAL self-play. SPIRAL is the online language-model self-play baseline in our study because it improves the same LLM policy that generates the game trajectories (Liu et al., 2025a). For a sampled zero-sum game G, a policy π_θ plays both roles, receives antisymmetric terminal returns $R_0(\tau) = -R_1(\tau)$, and updates from role-conditioned advantages that subtract a moving baseline for each game-role pair. This makes SPIRAL a strong comparison for self-play transfer, but it also couples the curriculum, opponent, trace quality, and student policy in one online optimization loop. SPSD keeps the self-play search source but decouples it from the LLM: compact ExIt-trained experts generate trajectories offline, and the language model sees only rendered state-action or trace-supervision examples. The comparison with SPIRAL therefore asks whether a separate search teacher can supply stronger and more interpretable supervision than learner-coupled self-play.

4 Method

SPSD separates search from language-model training. A compact game expert plans inside an executable simulator, while the language model sees only rendered supervised conversations. This decoupling fixes the student objective and lets us vary the source game, expert strength, search budget, or trace renderer without changing the downstream interface. The pipeline has four stages: define an executable game interface, train or select a frozen ExIt-family teacher, export one compact search record per expert move, and render that record as either SPSD final-action supervision or SPSD-CoT procedure supervision.

4.1 Problem setting

Formally, we define $\mathcal{G}$ as a collection of finite executable games. Each game $g \in \mathcal{G}$ defines a state space $\mathcal{S}_g$, a legal-action function $\mathcal{A}_g(s)$, a transition function $T_g(s, a)$, a terminal predicate, and an outcome function $R_g(s_{\text{term}}) \in [-1, 1]$. The experiments use two-player games, but the interface only requires that the simulator can advance legal actions and verify terminal outcomes. Each game also has a renderer ρ_g that converts a simulator state into a language-model prompt containing the board, the side to move, and the legal actions. Each supervised example pairs a rendered prompt $x = \rho_g(s)$ with an assistant response y that must end with a legal action. The training signal is useful only when the final action comes from a stronger policy and, for trace supervision, the response exposes transferable operations such as state preservation, branch comparison, threat detection, and delayed-reward reasoning.

4.2 External search teachers

For each source game g, we train or select an ExIt-family expert policy, with MuZero and EfficientZero MCTS as concrete implementations. The expert is trained by self-play inside the executable simulator and then frozen for data generation. At a non-terminal state s, the expert runs MCTS from a root node $r(s)$ and returns a selected action $a^\star$. The search also produces the root visit distribution $\pi(\cdot \mid s)$, a root value estimate $v(s)$, ranked root actions, and replayable branch prefixes. We write the exported object for one expert decision as

$$m(s) = \big(s, \mathcal{A}_g(s), a^\star, \pi(\cdot \mid s), v(s), \mathcal{B}(s)\big), \quad (4)$$

where $\mathcal{B}(s)$ contains the retained branch metadata needed to reconstruct searched continuations. The tuple in Equation 4 is the only contract between the game expert and the language-data renderer. The student is never optimized with reinforcement learning in the game environment; it is trained only on rendered pairs derived from $m(s)$.

4.3 Supervision formats

We render each retained expert decision in one of two formats that share the same prompt $x = \rho_g(s)$ and differ only in the assistant response y.

SPSD supervision. The direct renderer imitates the expert's final decision. It produces a short response that identifies $a^\star$ and ends with the required boxed move. This format transfers the search-improved action label but not the search procedure that produced it.

SPSD-CoT supervision. The SPSD-CoT renderer uses the same selected action but conditions the response on the retained search record $\mathcal{B}(s)$. It verbalizes a small number of rejected alternatives, searched opponent replies or follow-up moves, replay-derived terminal evidence when available, and the final commitment to $a^\star$. Raw visits, priors,

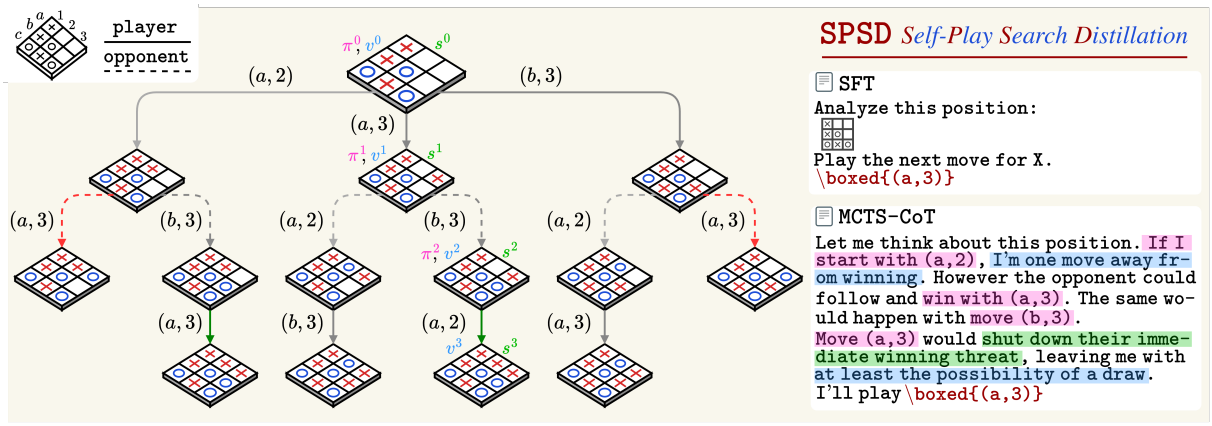


Figure 2: **Self-play search data generation pipeline.** Self-play experts generate direct SPSD state-action examples and SPSD-CoT procedure-cloning traces from the same executable game state.

and Q values select retained branches but are not exposed in the surface text. SPSD-CoT is therefore a compressed procedure-cloning target. It is intended to preserve the comparison structure of search without asking the language model to imitate simulator internals or arbitrary rationales.

4.4 Linearizing MCTS traces

The full MCTS tree is too large and too implementation-specific to render directly. The SPSD-CoT linearizer keeps only the search evidence needed to make one expert decision inspectable. Given a root $r(s_0)$ and selected action $a^\star$, it retains ranked root candidates C, bounded search paths $\mathcal{P}$, shared-topology summaries S, and relative branch features F. These objects preserve three invariants: the final answer remains $a^\star$, every verbalized branch is replayable as a legal simulator prefix, and failed branches are deduplicated at the level of decision-relevant alternatives. The renderer then uses terminal replay evidence when available, otherwise falls back to relative root-candidate comparisons. This is narrower than ASTRO-style search linearization (Kim et al., 2025): ASTRO linearizes a broader search process that ends at a correct terminal node, whereas SPSD-CoT starts from the expert's chosen root action and explains why that action was selected. In all cases, the final line follows the same boxed-action contract as SPSD, so format comparisons isolate trace structure rather than answer formatting.

5 Experimental Setup

We evaluate SPSD as a data-generation mechanism rather than as a new game-playing algorithm. Every condition uses the same downstream answer interface: the model receives the rules, current state, side to move, and legal actions, and the response must end with a boxed legal move. We define one evaluation instance as a rendered game state or external benchmark item paired with the model response produced under that fixed answer contract. This fixed interface isolates the experimental factors of interest: source-game coverage, trace format, and MCTS teacher budget.

5.1 Game Corpus

The training pool contains 50 executable two-player environments. The parent games were selected from the Ludii game repository, ported into the maintained executable board-game interface, and retained only after deterministic checks of legal actions, state transitions, terminal outcomes, and prompt rendering. The five selected parents cover distinct reasoning primitives: Connect4 stresses alignment planning under gravity, Tic-Tac-Chess stresses rule-conditioned tactics, 3 6 9 stresses counting over scored lines, Domineering stresses spatial tiling and blocking, and First Attack stresses constraint satisfaction over attack lines. These capability icons are used again in Figure 5, where marker shape identifies the game family behind each plotted point. The final source corpus contains the five parent games—Connect4, Tic-Tac-Chess, 3 6 9, Domineering, and First Attack—and nine validated variants for each parent. The resulting 45 variants preserve the parent observation shape, board topology, and action-space size while perturbing local rules, including scoring thresholds, winning directions, openings, locked cells, movement rules, and misere objectives. This design separates two forms of

generalization: transfer across nearby variants of a known game family and transfer to unseen game families. The parent games and held-out set are documented in Section A, while the variant protocol and corpus statistics are detailed in Section B. For cross-family transfer, 15 additional games are reserved as the held-out set and are never used for SPSD data generation.

5.2 Models and Baselines

The SPSD student is Qwen3-4B trained by supervised fine-tuning on rendered game conversations. We compare two supervision formats under the same prompt and boxed-answer contract. The SPSD condition clones only the final expert action. The SPSD-CoT condition uses the same expert action but exposes a compact search record containing retained alternatives, opponent replies, and replayed terminal evidence. Thus, the comparison between SPSD and SPSD-CoT tests whether procedure-level search supervision adds value beyond final-action imitation.

We include base Qwen3 checkpoints and SPIRAL as the main language-model baselines because SPIRAL is the closest online LLM self-play comparison. For the direct-control setting, we train a Qwen3-4B-Instruct SPIRAL run on the same five parent games used by SPSD. For the reasoning-dynamics analysis, we first report SPIRAL-Qwen3-8B, SPIRAL-OctoThinker-8B, SPIRAL-Llama3-8B, and SPIRAL-Qwen3-4B as completed baselines; the final comparison will add Qwen3-4B-SPSD-CoT once its run is available. This separates online learner-coupled self-play from offline search-teacher supervision while keeping the held-out game prompts and analysis pipeline fixed.

5.3 Metrics and Analysis Groups

External transfer metrics report cross-domain reasoning performance. They report the average score over MATH500, AIME24, AIME25, AMC23, OlympiadBench, MinervaMath, GPQA, and MMLU-Pro, together with the per-benchmark scores in Table 1. Executable-game transfer metrics report direct transfer to board-game play. Figure 3 reports win rate on the 15 held-out evaluation games and on held-out states drawn from the training families. Environment-scale metrics report the source-pool sweep, where we evaluate 5, 10, 20, 30, 40, and 50 source environments; confidence intervals in Figure 3 are 95% intervals over the corresponding non-base curves. Search-geometry metrics define the RQ5 trace analysis. Each visible model trace is split into numbered paragraphs and mapped to a search-tree JSON schema before metric computation. The completed SPIRAL-family analysis reported in Figure 5 uses the package's conservative local explicit-action extractor, which creates tree nodes only from explicit mentions of benchmark legal actions. The GPT-5.5 mapper described in Section E is the stricter LLM-reviewed protocol reserved for the final SPSD-CoT comparison. Both mapping routes constrain root nodes to the benchmark legal-action list and count only simulated future moves as future-depth mentions, so references to current-board coordinates do not inflate search depth. Invalid or empty trees are retained as invalid records rather than manually repaired. β denotes legal root-action coverage, the code-level M_{breadth}, computed as the number of distinct root actions recovered in the trace divided by the number of legal actions in the benchmark state. γ denotes root-subtree Gini concentration, the code-level M_{focus}; higher values indicate that follow-up reasoning is concentrated under fewer candidate actions. δ denotes chosen-subtree depth, the code-level M_{depth}, defined as the maximum future-action depth under the model's selected root action. ω denotes episode win rate in the outcome panel. Replayed value metrics use the 50-simulation EfficientZero replay budget. For Figure 5, the value panel reports $\mathrm{Q}50$, the MuZero chosen-action value at 50 simulations, δ, and EZA, the EfficientZero action-agreement indicator that the model's selected action equals the replayed search oracle action. Rows without successful value replay are omitted from the value panel rather than imputed. Appendix diagnostics include raw trace heuristics such as marker counts, repeated bigrams, and composite raw-search scores; these diagnostics are defined in Section F and kept out of the main quantitative claims. The MCTS-depth ablation keeps the student, renderer, and source pool fixed while varying the teacher search budget over 32, 64, 128, and 256 simulations per move.

5.4 Implementation Details

EfficientZero-family teachers generate the SPSD records after self-play training. For each retained move, the renderer exports either a direct state-action example or an SPSD-CoT conversation whose searched branches are replayable in the simulator. Renderer settings are held constant across ablations; additional hyperparameters are listed

in Section C. A representative rendered prompt is shown in Section D.

SPIRAL and SPSD fine-tuning use 4xH200 GPUs, EfficientZero teacher training uses 1xA100, and LLM game benchmarking uses 2xA100.

6 Results

We report five research-question-oriented analyses. Measurements that are not yet available are left blank in tables or explicitly marked as placeholders in figure captions. This convention separates completed baseline evidence from planned SPSD evaluations.

RQ1 Does SPSD transfer reasoning capabilities? The first analysis establishes the available external-reasoning baseline for the planned SPSD comparison. Table 1 reports the external reasoning benchmark comparison for available base and SPIRAL runs, while reserving SPSD and SPSD-CoT rows for pending measurements. For Qwen3-4B, SPIRAL-SFT increases the average score from 34.0 to 39.7, and SPIRAL-RL increases it to 44.5. For Qwen3-8B, the same comparison moves from 39.5 for the base model to 46.1 with SPIRAL-SFT and 49.6 with SPIRAL-RL. Figure 1 provides the corresponding game-facing motivation: even large LLMs remain weak on held-out board-game play, so the SPSD evaluation targets state tracking and branch comparison rather than ordinary answer imitation.

RQ2 Does SPSD generalize beyond memorization? The second analysis measures whether transfer changes as the source pool grows from the five parent games to the full parent-plus-variant corpus. Figure 3 reports the planned sweep over 5, 10, 20, 30, 40, and 50 training environments. The left panel tracks external reasoning score, and the right panel tracks win rate on the 15 held-out evaluation games. The shaded bands denote 95% confidence intervals; the current numerical trajectories and intervals are placeholders and are retained only to specify the final reporting format.

RQ3 Does SPSD-CoT improve reasoning abilities? The third analysis isolates the supervision format at matched game count. Table 1 leaves the matched Qwen3-4B SPSD and SPSD-CoT scores blank until final evaluation is available. Figure 3 specifies the planned format comparison by plotting SPSD and SPSD-CoT under identical source-pool sizes and the same held-out evaluation protocol. The completed result will test whether searched alternatives and replayed outcomes add measurable transfer beyond the expert's final action label.

RQ4 How does MCTS depth affect transfer? The fourth analysis varies the teacher search budget used to produce SPSD-CoT traces. Figure 4 is the planned MCTS-depth ablation: the placeholder bars increase from 52.8 average reasoning at 32 simulations to 58.4 at 256 simulations, with diminishing gains after 128 simulations. The completed ablation will report whether additional MCTS simulations improve transfer after controlling for the renderer and the student training recipe.

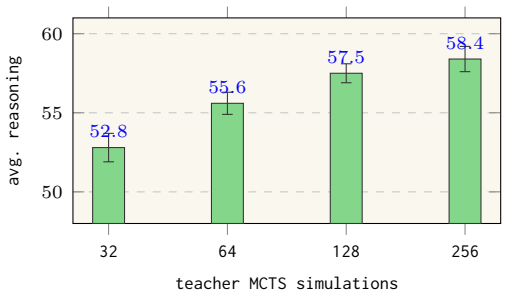


Figure 4: **MCTS-depth distillation ablation.** Bars show the planned SPSD-CoT comparison when teacher traces are generated with increasing MCTS simulation budgets. Values are random placeholders and must be replaced with final depth-sweep measurements. TODO

RQ5 Is reasoning shaped to follow MCTS? The fifth analysis evaluates whether visible model traces contain explicit search-tree structure. Figure 5 is the main reasoning-dynamics view for the model comparisons in this analysis and is rendered from the exact export produced by the RQ5 plotting pipeline. The present export contains the completed SPIRAL rows; SPSD and SPSD-CoT use the same encoding when their rows are added to the table. Color identifies the model, while marker shape identifies the game family: Connect4, Tic-Tac-Chess, 3 6 9, Domineering, or First Attack. The legend also recalls the intended reasoning primitive for each family: alignment planning, rule-conditioned tactics, counting, spatial tiling, or constraint satisfaction. Panels (a) and (b) contain 20 model-game centroids, and panel (c) contains 18 centroids because value replay is unavailable for two model-game pairs. Panel (a) shows that the Qwen and OctoThinker SPIRAL variants expose more recoverable search geometry than Llama3-8B. Across the five game cen-

Student	Training condition	Avg.	MATH500	AIME24	AIME25	AMC23	Olympiad	Minerva	GPQA	MMLU-Pro
Qwen3-4B	∅ base	34.0	73.4	9.6	6.2	42.4	33.3	29.4	30.6	47.2
	⟳ SPIRAL –SFT	39.7	74.2	13.7	11.7	51.1	37.6	40.1	37.8	51.3
	⟳ SPIRAL –RL	44.5	78.2	19.7	13.3	61.6	41.8	42.6	40.1	58.5
	▶ SDPS	--	--	--	--	--	--	--	--	--
	♦ SDPS-CoT	--	--	--	--	--	--	--	--	--
Qwen3-8B	∅ base	39.5	77.0	12.1	11.2	50.6	33.5	38.2	38.0	55.7
	⟳ SPIRAL –SFT	46.1	82.8	19.9	15.6	63.5	45.9	40.8	41.6	58.8
	⟳ SPIRAL –RL	49.6	86.6	26.2	16.8	65.2	49.6	46.3	44.6	61.1

Table 1: **Main reasoning benchmark results.** Average and per-benchmark scores for available base Qwen3 and SPIRAL runs on the external reasoning suite, with SPSD and SPSD-CoT rows kept as planned comparison conditions. Dashes indicate measurements that are not yet available and must be filled only after final evaluation. TODO

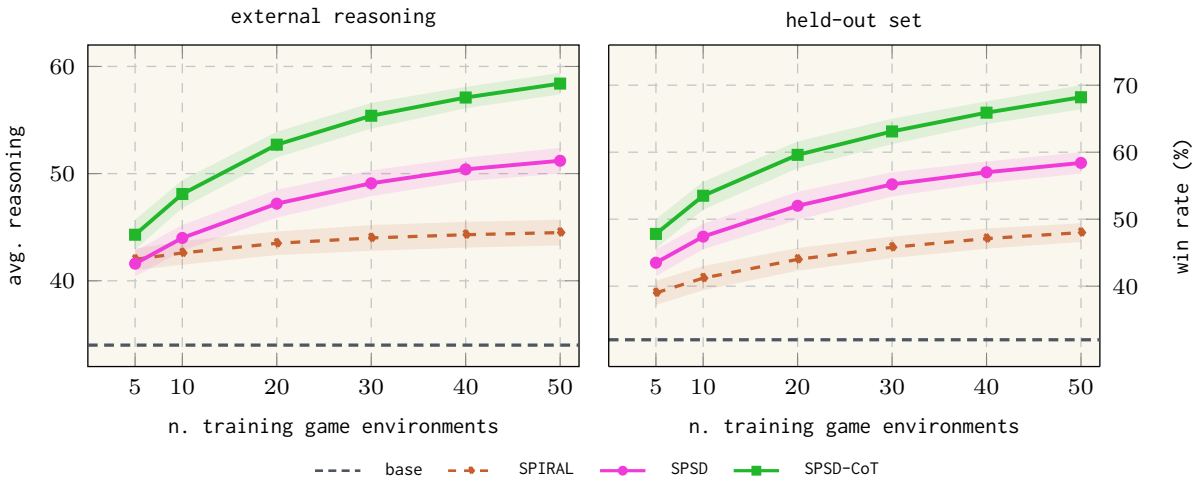


Figure 3: **Training-game scale and held-out transfer.** The left panel reports external reasoning performance as the SPSD source pool grows from five parent games to 50 parent-plus-variant environments. The right panel reports the win rate on the 15 held-out evaluation games under the same sweep. Shaded regions denote 95% confidence intervals around each non-base curve. Curves and intervals use random placeholder values and must be replaced with final sweep measurements. TODO

troids, SPIRAL-Qwen3-4B has mean $(\beta, \gamma, \delta) = (0.58, 0.26, 3.50)$ and SPIRAL-Qwen3-8B has $(0.59, 0.25, 3.36)$. SPIRAL-OctoThinker-8B is similar in depth but lower in breadth, with $(0.51, 0.24, 3.35)$, while SPIRAL-Llama3-8B is shallow at $(0.36, 0.02, 1.17)$. Panel (b) shows that explicit tree geometry alone does not determine game success. The First Attack marker has high win rates for all four models, from 0.55 for SPIRAL-Qwen3-4B to 1.00 for SPIRAL-Llama3-8B, while Connect4, Tic-Tac-Chess, and 3 6 9 remain at 0.00 for every displayed model despite non-trivial search geometry. Panel (c) adds the EfficientZero replay check at 50 simulations, comparing Q50, δ, and EZA. EZA remains game-specific: First Attack reaches 0.34–0.45 for the value-scored non-Llama models, Domineering reaches 0.37 for SPIRAL-Qwen3-4B and 0.30 for SPIRAL-Qwen3-8B, but Connect4 remains at or below 0.04. Thus, the completed SPIRAL baseline provides a concrete comparison target: some SPIRAL traces expose recoverable branch structure, but (β, γ, δ), ω, and EZA separate by game family rather than moving as a single model-level property.

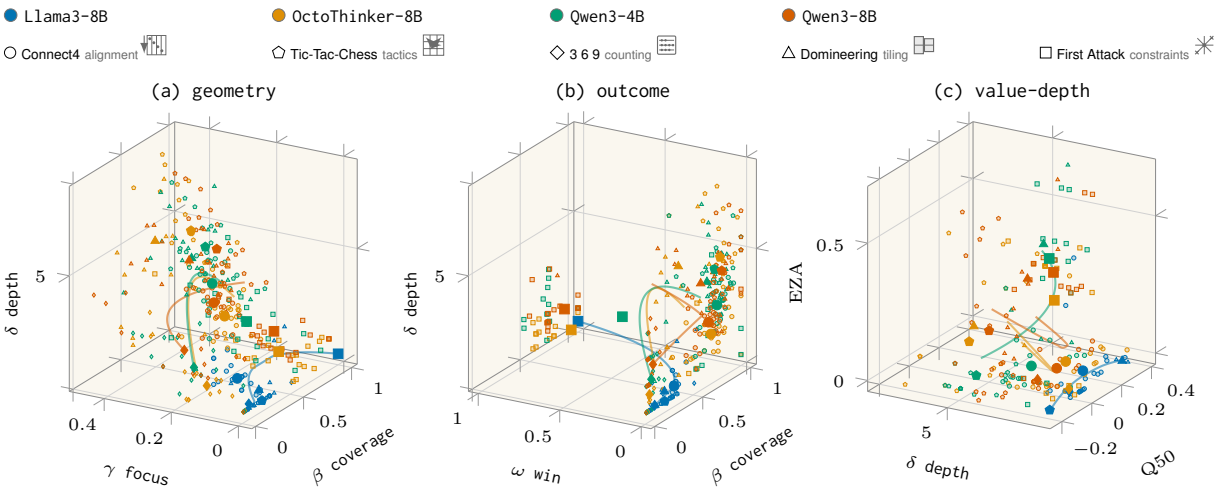


Figure 5: **Reasoning dynamics across game-trained language models.** Color denotes the model or training condition, marker shape the game family, and the trailing icon the reasoning primitive. Small points are episodes, curves are model trends, and large points are model-game centroids. Panel (a) shows root-action coverage β, root-subtree concentration γ, and chosen-subtree depth δ. Panel (b) replaces γ with win rate ω. Panel (c) shows Q50, the 50-simulation MuZero chosen-action value; δ; and EZA, EfficientZero oracle-action agreement.

7 Conclusion

Self-Play Search Distillation turns executable board-game search into post-training data for language models. The reshaped evaluation tests whether Qwen3-4B-SPSD transfers reasoning, whether transfer grows across the 50-game parent-plus-variant pool, whether SPSD-CoT improves over final-action imitation, and whether visible reasoning becomes more MCTS-shaped. The central hypothesis is that programmable self-play search can provide verifiable supervision beyond ordinary human demonstrations and beyond learner-coupled online self-play.

References

Thomas Anthony, Robert Nishihara, Philipp Moritz, Tim Salimans, and John Schulman. 2019. Policy gradient search: Online planning and expert iteration without search trees. *CoRR*, abs/1904.03646.

Thomas Anthony, Zheng Tian, and David Barber. 2017. Thinking fast and slow with deep learning and tree search. In *NIPS*.

Thomas William Anthony. 2021. *Expert Iteration*. Ph.D. thesis, University College London.

Ioannis Antonoglou, Julian Schrittwieser, Sherjil Ozair, Thomas K. Hubert, and David Silver. 2022. Planning in stochastic environments with a learned model. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Yoram Bachrach, Edan Toledo, Karen Hambardzumyan, Despoina Magka, Martin Josifoski, Minqi Jiang, Jakob N. Foerster, Roberta Raileanu, Tatiana Shavrina, Nicola Cancedda, Avraham Ruderman, Katie Millican, Andrei Lupu, and Rishi Hazra. 2025. Combining code generating large language models and self-play to iteratively refine strategies in games. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2025, Montreal, Canada, August 16-22, 2025*, pages 10999–11003. ijcai.org.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. 2013. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Intell. Res.*

Jiangjie Chen, Qianyu He, Siyu Yuan, Aili Chen, Zhicheng Cai, Weinan Dai, Hongli Yu, Qiying Yu, Xuefeng Li, Jiaze Chen, Hao Zhou, and Mingxuan Wang. 2025. Enigmata: Scaling logical reasoning in large language models with synthetic verifiable puzzles. *CoRR*, abs/2505.19914.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024. Self-play fine-tuning converts weak language models to strong language models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pages 6621–6642. PMLR / OpenReview.net.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 4299–4307.

Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. 2020. Leveraging procedural generation to benchmark reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, Proceedings of Machine Learning Research, pages 2048–2056. PMLR.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Matthew J. Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2018. Textworld: A learning environment for text-based games. In *CGW@IJCAI*.

Ivo Danihelka, Arthur Guez, Julian Schrittwieser, and David Silver. 2022. Policy improvement by planning with gumbel. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Michael Dennis, Natasha Jaques, Eugene Vinitsky, Alexandre M. Bayen, Stuart Russell, Andrew Critch, and Sergey Levine. 2020. Emergent complexity and zero-shot transfer via unsupervised environment design. In *NeurIPS*.

Jinhao Duan, Renming Zhang, James Diffenderfer, Bhavya Kailkhura, Lichao Sun, Elias Stengel-Eskin, Mohit Bansal, Tianlong Chen, and Kaidi Xu. 2024. Gtbench: Uncovering the strategic reasoning limitations of llms via game-theoretic evaluations. *CoRR*, abs/2402.12348.

'Eric Piette, Dennis J. N. J. Soemers, Matthew Stephenson, Chiara F. Sironi, Mark H. M. Winands, and Cameron Browne. 2020. Ludii - the ludemic general game system. In *ECAI 2020 - 24th European Conference on Artificial Intelligence, 29 August-8 September 2020, Santiago de Compostela, Spain, August 29 - September 8, 2020 - Including 10th Conference on Prestigious Applications of Artificial Intelligence (PAIS 2020)*, Frontiers in Artificial Intelligence and Applications, pages 411–418. IOS Press.

Xidong Feng, Yicheng Luo, Ziyan Wang, Hongrui Tang, Mengyue Yang, Kun Shao, David Mguni, Yali Du, and Jun Wang. 2023. Chessgpt: Bridging policy learning and language modeling. In *NeurIPS*.

Kanishk Gandhi, Denise Lee, Gabriel Grand, Muxin Liu, Winson Cheng, Archit Sharma, and Noah D. Goodman. 2024. Stream of search (sos): Learning to search in language. *CoRR*, abs/2404.03683.

Jean-Bastien Grill, Florent Altch'e, Yunhao Tang, Thomas Hubert, Michal Valko, Ioannis Antonoglou, and R'emi Munos. 2020. Monte-carlo tree search as regularized policy optimization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, Proceedings of Machine Learning Research, pages 3769–3778. PMLR.

Leon Guertler, Bobby Cheng, Simon Yu, Bo Liu, Leshem Choshen, and Cheston Tan. 2025. Textarena. *CoRR*, abs/2504.11442.

Hongyi Guo, Zhihan Liu, Yufeng Zhang, and Zhaoran Wang. 2024. Can large language models play games? A case study of A self-play approach. *CoRR*, abs/2403.05632.

Qianyue Hao, Sibo Li, Jian Yuan, and Yong Li. 2025. RL of thoughts: Navigating LLM reasoning with inference-time reinforcement learning. *CoRR*, abs/2505.14140.

Xiaotian Hao, Jianye Hao, Chenjun Xiao, Kai Li, Dong Li, and Yan Zheng. 2024. Multiagent gumbel muzero: Efficient planning in combinatorial action spaces. In *Thirty-Eighth AAAI Conference on Artificial Intelligence, AAAI 2024, Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence, IAAI 2024, Fourteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2014, February 20-27, 2024, Vancouver, Canada*, pages 12304–12312. AAAI Press.

Matteo Hessel, Ivo Danihelka, Fabio Viola, Arthur Guez, Simon Schmitt, Laurent Sifre, Theophane Weber, David Silver, and Hado van Hasselt. 2021. Muesli: Combining improvements in policy optimization. In *International Conference on Machine Learning, ICML 2021*, pages 4214–4226.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR*, abs/1503.02531.

Lanxiang Hu, Mingjia Huo, Yuxuan Zhang, Haoyang Yu, Eric P. Xing, Ion Stoica, Tajana Rosing, Haojian Jin, and Hao Zhang. 2025. lmgame-bench: How good are llms at playing games? *CoRR*, abs/2505.15146.

Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Mohammadamin Barekatain, Simon Schmitt, and David Silver. 2021a. Learning and planning in complex action spaces. In *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, Proceedings of Machine Learning Research, pages 4476–4486. PMLR.

Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Mohammadamin Barekatain, Simon Schmitt, and David Silver. 2021b. Learning and planning in complex action spaces. *CoRR*, abs/2104.06303.

Adam Karvonen. 2024. Emergent world models and latent variable estimation in chess-playing language models. *CoRR*.

Joongwon Kim, Anirudh Goyal, Liang Tan, Hannaneh Hajishirzi, Srinivasan Iyer, and Tianlu Wang. 2025. ASTRO: teaching language models to reason by reflecting and backtracking in-context. *CoRR*, abs/2507.00417.

Marc Lanctot, Edward Lockhart, Jean-Baptiste Lespiau, Vinícius Flores Zambaldi, Satyaki Upadhyay, Julien Pérolat, Sriram Srinivasan, Finbarr Timbers, Karl Tuyls, Shayegan Omidshafiei, Daniel Hennes, Dustin Morrill, Paul Muller, Timo Ewalds, Ryan Faulkner, János Kramár, Bart De Vylder, Brennan Saeta, James Bradbury, and 8 others. 2019. Openspiel: A framework for reinforcement learning in games. *CoRR*, abs/1908.09453.

Michael Laskin, Luyu Wang, Junhyuk Oh, Emilio Parisotto, Stephen Spencer, Richie Steigerwald, DJ Strouse, Steven Stenberg Hansen, Angelos Filos, Ethan Brooks, Maxime Gazeau, Himanshu Sahni, Satinder Singh, and Volodymyr Mnih. 2023. In-context reinforcement learning with algorithm distillation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Kenneth Li, Aspen K. Hopkins, David Bau, Fernanda B. Viégas, Hanspeter Pfister, and Martin Wattenberg. 2023. Emergent world representations: Exploring a sequence model trained on a synthetic task. In *ICLR*. OpenReview.net.

Bo Liu, Leon Guertler, Simon Yu, Zichen Liu, Penghui Qi, Daniel Balcells, Mickel Liu, Cheston Tan, Weiyan Shi, Min Lin, Wee Sun Lee, and Natasha Jaques. 2025a. Spiral: Self-play on zero-sum games incentivizes reasoning via multi-agent multi-turn reinforcement learning. *CoRR*, abs/2506.24119.

Mingjie Liu, Shizhe Diao, Ximing Lu, Jian Hu, Xin Dong, Yejin Choi, Jan Kautz, and Yi Dong. 2025b. Prorl: Prolonged reinforcement learning expands reasoning boundaries in large language models. *CoRR*, abs/2505.24864.

Baiting Luo, Ava Pettet, Aron Laszka, Abhishek Dubey, and Ayan Mukhopadhyay. 2025. Scalable decision-making in stochastic environments through learned temporal abstraction. In *International Conference on Learning Representations, ICLR 2025*.

Yixuan Mei, Jiaxuan Gao, Weirui Ye, Shaohuai Liu, Yang Gao, and Yi Wu. 2023. Speedyzero: Mastering atari with limited data and time. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Neel Nanda, Andrew Lee, and Martin Wattenberg. 2023. Emergent linear representations in world models of self-supervised sequence models. In *BlackboxNLP@EMNLP*.

Yazhe Niu, Yuan Pu, Zhenjie Yang, Xueyan Li, Tong Zhou, Jiyuan Ren, Shuai Hu, Hongsheng Li, and Yu Liu. 2023. Lightzero: A unified benchmark for monte carlo tree search in general sequential decision scenarios. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022,*

NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022.

Sherjil Ozair, Yazhe Li, Ali Razavi, Ioannis Antonoglou, Aäron van den Oord, and Oriol Vinyals. 2021. Vector quantized models for planning. In *ICML*.

Jack Parker-Holder, Minqi Jiang, Michael Dennis, Mikayel Samvelyan, Jakob N. Foerster, Edward Grefenstette, and Tim Rocktäschel. 2022. Evolving curricula with regret-based environment design. In *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, Proceedings of Machine Learning Research, pages 17473–17498. PMLR.

Yuan Pu, Yazhe Niu, Jia Tang, Junyu Xiong, Shuai Hu, and Hongsheng Li. 2025a. One model for all tasks: Leveraging efficient world models in multi-task planning. *CoRR*, abs/2509.07945.

Yuan Pu, Yazhe Niu, Zhenjie Yang, Jiyuan Ren, Hongsheng Li, and Yu Liu. 2025b. Unizero: Generalized and efficient planning with scalable latent world models. *Transactions on Machine Learning Research*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Anian Ruoss, Grégoire Delétang, Sourabh Medapati, Jordi Grau-Moya, Li Kevin Wenliang, Elliot Catt, John Reid, and Tim Genewein. 2024. Grandmaster-level chess without search. *CoRR*, abs/2402.04494.

Martin Schmid, Matej Moravčík, Neil Burch, Rudolf Kadlec, Josh Davidson, Kevin Waugh, Nolan Bard, Finbarr Timbers, Marc Lanctot, G. Zacharias Holland, Elnaz Davoodi, Alden Christianson, and Michael Bowling. 2023. Student of games: A unified learning algorithm for both perfect and imperfect information games. *Science Advances*, 9(46):eadg3256.

Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, Timothy P. Lillicrap, and David Silver. 2020. Mastering atari, go, chess and shogi by planning with a learned model. *Nat.*, 588(7839):604–609.

Julian Schrittwieser, Thomas Hubert, Amol Mandhane, Mohammadamin Barekatain, Ioannis Antonoglou, and David Silver. 2021a. Online and offline reinforcement learning by planning with a learned model. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pages 27580–27591.

Julian Schrittwieser, Thomas Hubert, Amol Mandhane, Mohammadamin Barekatain, Ioannis Antonoglou, and David Silver. 2021b. Online and offline reinforcement learning by planning with a learned model. *CoRR*, abs/2104.06294.

John Schultz, Jakub Adámek, Matej Jusup, Marc Lanctot, Michael Kaisers, Sarah Perrin, Daniel Hennes, Jeremy Shar, Cannada A. Lewis, Anian Ruoss, Tom Zahavy, Petar Velickovic, Laurel Prince, Satinder Singh, Eric Malmi, and Nenad Tomasev. 2025. Mastering board games by external and internal planning with language models. In *ICML*, Proceedings of Machine Learning Research. PMLR / OpenReview.net.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *CoRR*, abs/2402.03300.

Jiajun Shi, Jian Yang, Jiaheng Liu, Xingyuan Bu, Jiangjie Chen, Junting Zhou, Kaijing Ma, Zhoufutu Wen, Bingli Wang, Yancheng He, Liang Song, Hualei Zhu, Shilong Li, Xingjian Wang, Wei Zhang, Ruibin Yuan, Yifan Yao, Wenjun Yang, Yunli Wang, and 10 others. 2025. Korgym: A dynamic game platform for LLM reasoning evaluation. *CoRR*, abs/2505.14552.

David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of go with deep neural networks and tree search. *Nat.*

David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, Timothy P. Lillicrap, Karen Simonyan, and Demis Hassabis. 2017a. Mastering chess and shogi by self-play with a general reinforcement learning algorithm. *CoRR*, abs/1712.01815.

David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. 2017b. Mastering the game of go without human knowledge. *Nat.*

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron T. Parisi, Abhishek Kumar, Alexander A. Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Fathy Elsayed, Hanie Sedghi, and 22 others. 2024. Beyond human data: Scaling self-training for problem-solving with language models. *Trans. Mach. Learn. Res.*, 2024.

Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. 2025. REASONING GYM: reasoning environments for reinforcement learning with verifiable rewards. *CoRR*, abs/2505.24760.

Trieu H. Trinh, Yuhuai Wu, Quoc V. Le, He He, and Thang Luong. 2024. Solving olympiad geometry without human demonstrations. *Nat.*, 625(7995):476–482.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo Hernandez, Sarath Sreedharan, and Subbarao Kambhampati. 2023a. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2023b. On the planning abilities of large language models - A critical investigation. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O. Stanley. 2019. Paired open-ended trailblazer (POET): endlessly generating increasingly complex and diverse learning environments and their solutions. *CoRR*, abs/1901.01753.

Shengjie Wang, Shaohuai Liu, Weirui Ye, Jiacheng You, and Yang Gao. 2024. Efficientzero V2: mastering discrete and continuous control with limited data. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Shu Wang, Lei Ji, Renxi Wang, Wenxiao Zhao, Haokun Liu, Yifan Hou, and Ying Nian Wu. 2025. Explore the reasoning capability of llms in the chess testbed. In *NAACL*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 13484–13508. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Zhen Hao Wong, Jingwen Deng, Runming He, Zirong Chen, Qijie You, Hejun Dong, Hao Liang, Chengyu Shen, Bin Cui, and Wentao Zhang. 2025. Logicpuzzlerl: Cultivating robust mathematical reasoning in llms via reinforcement learning. *CoRR*, abs/2506.04821.

Haoyuan Wu, Xueyi Chen, Rui Ming, Jilong Gao, Shoubo Hu, Zhuolun He, and Bei Yu. 2025. Totrl: Unlock LLM tree-of-thoughts reasoning potential through puzzles solving. *CoRR*, abs/2505.12717.

Mengjiao Yang, Dale Schuurmans, Pieter Abbeel, and Ofir Nachum. 2022. Chain of thought imitation with procedure cloning. In *NeurIPS*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *NeurIPS*.

Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. 2021. Mastering atari games with limited data. In *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, virtual*, pages 25476–25488.

Thomas Zeng, Shuibai Zhang, Shutong Wu, Christian Classen, Daewon Chae, Ethan Ewer, Minjae Lee, Heeju Kim, Wonjun Kang, Jackson Kunde, Ying Fan, Jungtaek Kim, Hyung Il Koo, Kannan Ramchandran, Dimitris Papailiopoulos, and Kangwook Lee. 2025. Versaprm: Multi-domain process reward model via synthetic reasoning data. *CoRR*, abs/2502.06737.

Yinqi Zhang, Xintian Han, Haolong Li, Kedi Chen, and Shaohui Lin. 2025. Complete chess games enable LLM become A chess master. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2025 - Volume 2: Short Papers, Albuquerque, New Mexico, April 29 - May 4, 2025*, pages 1–7. Association for Computational Linguistics.

Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Yang Yue, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. 2025. Absolute zero: Reinforced self-play reasoning with zero data. *CoRR*, abs/2505.03335.

A Game Corpus

As a supplement to the main paper's experimental setup in Section 5, this appendix documents the game corpus used to select the five SPSD parent games and the held-out evaluation set. We constructed the corpus from the Ludii repository ('Eric Piette et al., 2020), which provides rule descriptions for a broad collection of board games, by selecting finite two-player games with compact board states, discrete legal actions, and terminal outcomes that can be rendered and checked by the SPSD executable interface. Each selected game was translated into a language-facing environment with a normalized board renderer, legal-action enumerator, transition function, terminal predicate, and outcome function. The five SPSD parent games are Connect4, Tic-Tac-Chess, 3 6 9, Domineering, and First Attack; the remaining games form the held-out evaluation set. Each card pairs a representative TikZ board sketch with the compact rules needed to interpret model prompts and legal actions. The cards are synchronized to the paper-facing game roster used by dataset generation and benchmark evaluation.

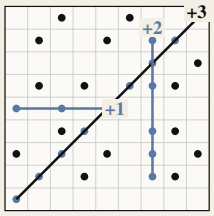
A.1 Game Cards

This subsection supports the main corpus description by listing the 20 selected games, their prompt-facing rules, and a representative board sketch drawn in the same TikZ style as the Connect4 card.

3 6 9

Shared-stone scoring game over lines of length three, six, and nine.

ACTION shared placement
GOAL higher score

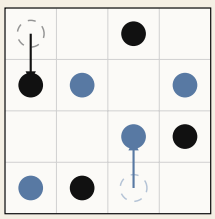


- Players share an empty 9×9 board; stones do not belong to either player after placement.
- On each turn, place one shared stone on any empty cell.
- The mover scores from non-contiguous occupied lines through the placed cell.
- Lines of exactly three, six, or nine occupied cells score one, two, or three points; the higher final score wins.

A Simple Game

Orthogonal movement game with a three-in-a-row objective.

ACTION orthogonal steps
GOAL line of three

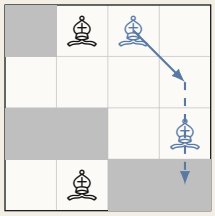


- Each player starts with four stones on alternating cells along the top and bottom edges.
- Move one own stone exactly one orthogonal step into an adjacent empty cell.
- Diagonal movement is not allowed, and there are no captures.
- A player wins immediately after forming a horizontal, vertical, or diagonal line of three stones.

Agapi

Bishop movement followed by a permanent orthogonal blocker shot.

ACTION bishop move + shot
GOAL immobilize

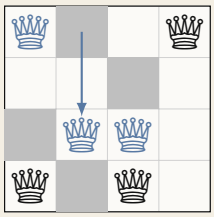


- Each player has two bishops on a 4×4 board.
- A turn first moves one bishop diagonally to the farthest reachable empty cell.
- From the destination, the mover shoots a permanent blocker orthogonally to the farthest reachable empty cell.
- If a player begins their move phase with no legal move-plus-shot option, the other player wins.

All Queens Chess

Queen-slide race to form a line before the board locks.

ACTION queen slides
GOAL line of three

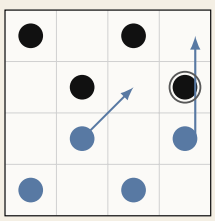


- Each player controls three queens on a 4×4 board.
- Choose a queen and slide it orthogonally through empty cells to the farthest reachable square.
- The square the queen left becomes a permanent blocker that no queen may enter or cross.
- A player wins by forming three queens in a row, column, or diagonal, or by leaving the opponent with no legal slide.

Alquerkonane

Checkers-like forward movement with orthogonal jump captures.

ACTION diagonal moves/jumps
GOAL immobilize

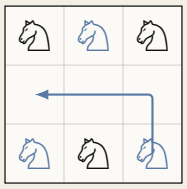


- The game starts from a fixed pawn pattern on a 4×4 board.
- A pawn may move one step diagonally forward to an empty square.
- A pawn may instead jump orthogonally over an adjacent enemy pawn to the empty square beyond, removing it.
- There is no promotion; a player wins by leaving the opponent with no legal move.

Aralzaa

Forward-leap race to occupy the opponent's home row.

ACTION forward leaps
GOAL opponent row

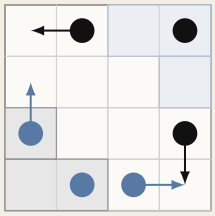


- Each player starts with three horse-shaped pieces on their home row.
- A piece leaps two squares forward and one square sideways into an empty square.
- The forward knight-style leap does not capture; a player with no legal leap passes.
- A player wins immediately after occupying all three squares of the opponent's starting row.

Bajr

Directional camp-race game on a compact square board.

ACTION forward steps
GOAL opposite camp

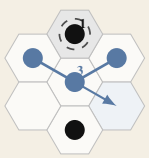


- Player 1 starts in the lower-left camp and Player 2 starts in the upper-right camp.
- Move one stone one orthogonal step into an empty cell.
- Player 1 may move upward or sideways toward the right; Player 2 may move downward or sideways toward the left.
- A player wins by occupying all three cells of the opponent's camp, or by leaving the opponent without a legal move.

Chains of Thought

Hex-chain game where legal moves depend on local group sizes.

ACTION place/merge/capture
GOAL largest chain

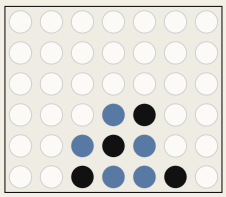


- Players use a radius-one hex board with seven cells.
- A turn either places a stone on a legal empty hex or moves one stone to an adjacent empty hex that merges friendly groups.
- Placement and movement legality depend on the relative sizes of adjacent friendly and opposing chains.
- After every move, smaller adjacent opposing chains can be captured; if no legal move remains, chain-size signatures decide the winner.

Connect 4

Vertical connection game on a 6×7 grid.

ACTION `column drop`
GOAL `connect four`

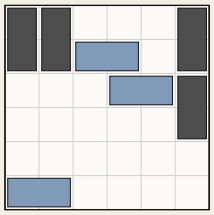


- Players use a vertical 6×7 grid and Player 1 moves first.
- Choose one non-full column and drop a disc into it; gravity sends it to the lowest empty cell.
- A move is illegal if the chosen column is already full.
- A player wins immediately after connecting four own discs horizontally, vertically, or diagonally.

Domineering

Asymmetric domino-placement game of move exhaustion.

ACTION `domino placement`
GOAL `last move`

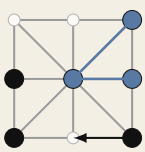


- Players share an initially empty 6×6 grid.
- The mover places one domino on two adjacent empty squares.
- One player may place only horizontal dominoes, while the other may place only vertical dominoes.
- A placement may not overlap or extend off the board; the first player with no legal placement loses.

Epelle

Adjacent-sliding alignment game where starting rows do not count.

ACTION `adjacent slides`
GOAL `new line of three`

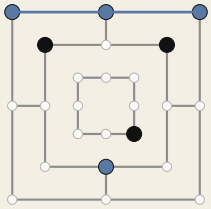


- Each player begins with three pieces already on opposite rows of a 3×3 intersection graph.
- Move one own piece to an adjacent connected empty intersection.
- Starting rows do not count as wins; the line must be created by play.
- A player wins by creating a line of three along the board lines.

Feldja

Alignment-and-capture game on a three-ring point graph.

ACTION `place/slide/remove`
GOAL `capture`

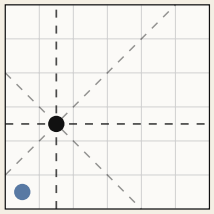


- Each player starts with nine reserve stones on a 24-point three-ring graph.
- Place reserve stones on empty points, then slide committed stones to adjacent empty points.
- A vacated point closes and cannot be reused; completing a marked line removes one opposing stone.
- A player wins by reducing the opponent to two or fewer total stones, or by leaving the opponent with no legal action.

First Attack

Non-attacking placement puzzle played as a last-move contest.

ACTION `non-attacking place`
GOAL `last move`

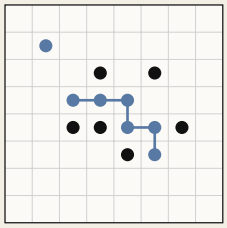


- Players share a 6×6 board and all placed stones are neutral.
- Place one shared stone on an empty cell.
- A legal cell shares no row, column, or diagonal with any occupied cell.
- The player who makes the last legal placement wins.

Groups

Connection game where friendly jumps help assemble one group.

ACTION `move/jump`
GOAL `connected group`

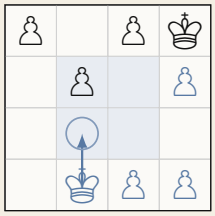


- The game starts from a fixed 8×8 position with seven Player 1 pieces and six Player 2 pieces.
- A piece either steps orthogonally to an adjacent empty square or jumps one friendly piece to the empty square beyond.
- Jumps do not capture; the jumped piece remains on the board.
- A player wins immediately after at least six own pieces form a single orthogonally connected group.

King's Valley

Sliding-piece race to bring the king into the central throne.

ACTION maximal slides
GOAL king to throne

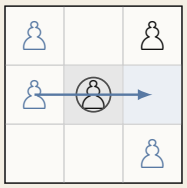


- Each player starts with one king and three guards on a 4×4 board.
- Choose one piece and one orthogonal direction; the piece slides as far as possible through empty cells.
- The four central cells form the throne zone.
- A player wins when their king lands on a throne-zone cell, or by leaving the opponent with no legal slide.

Moxie

Compact placement, movement, and forced-jump alignment game.

ACTION place/move/jump
GOAL line or capture

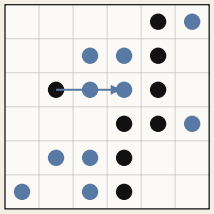


- Each player starts with four reserve stones on an empty 3×3 board.
- If no capture is available, place a reserve stone or move one stone one orthogonal step.
- If an orthogonal jump capture is available, the player must take one; captures do not chain.
- A player wins by forming three in a row, reducing the opponent to two or fewer total stones, or immobilizing the opponent.

Othello

Disc-flipping majority game with straight-line outflanks.

ACTION outflank flips
GOAL disc majority

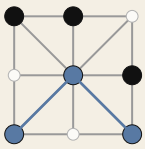


- The game uses a 6×6 board with the standard four-disc center start.
- Place a disc on an empty square that outflanks one or more straight lines of opposing discs.
- All outflanked discs immediately flip to the mover's color; a player with no legal placement passes.
- When the board is full or both players are stuck, the higher disc count wins.

Tapatan

Placement-then-sliding alignment game on a nine-point graph.

ACTION `place/slide`
GOAL `line of three`

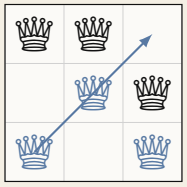


- Each player has three stones on a 3×3 point graph with diagonals.
- First place stones on empty intersections, then slide one stone to an adjacent connected empty intersection.
- Lines are checked along the board lines in both phases.
- A player wins immediately after forming a line of three.

Tic-Tac-Chess

Tic-tac-toe objective with queen movement after placement.

ACTION `place/queen moves`
GOAL `line of three`

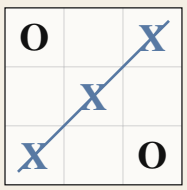


- Players first alternate placing three queen pieces each on empty cells of a 3×3 board.
- After all six pieces are placed, a queen may slide any distance in a straight line through empty cells.
- A queen may also hop over one adjacent enemy piece to the empty cell beyond; hops do not capture.
- A player wins immediately after forming a horizontal, vertical, or diagonal line of three pieces.

Tic-Tac-Toe

Classic mark-placement alignment game on a 3×3 grid.

ACTION `mark placement`
GOAL `line of three`



- Player 1 uses X and moves first; Player 2 uses O.
- Place one mark on any empty cell.
- A move is legal only if the chosen cell is empty.
- A player wins by forming a line of three; a full board without a line is a draw.

B Corpus Statistics

The SPSD training pool contains 50 environments: five parent games and 45 same-family variants. The held-out evaluation set contains 15 additional games, which are not used for SPSD student training. The statistics below summarize executable game dynamics for the 50 training environments and the variant-protocol categories used to create the 45 variants.

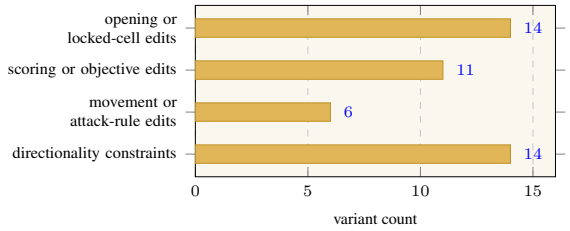


Figure 6: **Variant-family distribution.** The 45 controlled SPSD variants preserve each parent game's observation shape, board topology, and action-space size while perturbing openings or locked cells, scoring or objective rules, movement or attack rules, and directionality constraints.

Statistic	Min	Median	Mean	Max
Max game length	12	42	50.2	81
Action-space size	7	36	47.0	81
Board cells	9	36	40.8	81
Board rows	3	6	6.0	9
Board cols	3	6	6.2	9

Table 2: **SPSD training-pool statistics.** Summary over the exact 50 parent-plus-variant environments used for SPSD data generation.

Generation and validation workflow. The generation pipeline starts from Ludii rule descriptions ('Eric Piette et al., 2020) for candidate games. For each selected game, we normalize the rule text, define the prompt-facing state notation, enumerate legal actions, implement deterministic transition and terminal checks, and verify that simulator rollouts match the intended rules. GPT-5.4 is used at the language-facing authoring stages, including rule normalization, executable-specification drafting, and prompt scaffolding, but entry into the final corpus is decided by executable checks rather than by the model alone. Each candidate game must pass schema validation, legality checks, invariant rollouts, and downstream smoke training before it enters the supported library. Controlled variants then mutate one local factor at a time, typically a scoring threshold, winning direction, opening constraint, locked-cell pattern, movement rule, attack rule, or misere objective, so that transfer can be attributed to a specific perturbation. After training the ExIt-trained planning experts, the data renderer exports simulator-grounded trajectories and converts them into either SPSD state-action supervision or SPSD-CoT conversations with searched alternatives and terminal evidence.

C Training Hyperparameters

Table 3 records the default settings for the variant-authoring stage, the ExIt-trained planning experts, and the Qwen3-4B SPSD student. We keep renderer parameters fixed across ablations so that differences come from the source game, trace format, or teacher MCTS budget.

Component	Hyperparameter	Value
Variant authoring	Parent games	Connect4, Tic-Tac-Chess, 3 6 9, Domineering, First Attack
Variant authoring	Variants per parent	9
Held-out evaluation	Evaluation games	15 held-out games
ExIt-trained planning expert	MCTS simulations per move	32, 64, 128, 256
ExIt-trained planning expert	Exploration constant	$c_{\text{puct}} = 1.5$
Data renderer	Max retained root alternatives	$K = 3$
Data renderer	Max verbalized search depth	$D = 5$
Student SPSD	Base model	Qwen3-4B
Student SPSD	Sequence length	4096 tokens
Student SPSD	Learning rate	2×10^{-5}
Student SPSD	Epochs	2
Student SPSD	Effective batch size	128 examples

Table 3: **Training hyperparameters.** Settings for variant authoring, ExIt-trained planning experts, trace rendering, and Qwen3-4B SPSD post-training.

D Prompt Example

All training conversations follow the same contract. The system message contains the rules, the user message contains the rendered board plus legal actions, and the assistant must end with a boxed legal move. For readability, the example below shows a shortened Connect4-style SPSD-CoT prompt with the repo's ASCII board rendering.

Connect4 SPSD-CoT Training Example

```
System: Connect4 is a two-player connection
game on a vertical 6x7 grid. Players choose
a non-full column, the disc falls to the
lowest empty cell, and the first player to
connect four horizontally, vertically, or
diagonally wins.

User: Let's play Connect4. I'll be Player
2 and you be Player 1.
Description:  Gravity matters, central
columns support more future lines, and the
final answer must be a legal boxed move.
Board:
  6 . . . . . .
  5 . . . . . .
  4 . . . . . .
  3 X O X . . .
  2 O X O X . .
  1 X O X O X .
    a b c d e f g
Side to move: Player 1.
Legal actions: column a, column b, column
c, column d, column e, column f, column g.
Choose the expert move and finish with
\boxed{action}.

Assistant:  Column d is the cleanest
continuation  because  the  searched
alternatives leave weaker replies: column
c lets Player 2 answer in d and flatten
the central threat, while column e keeps
material but does not create an immediate
fork. After d, the retained rollout still
leaves Player 1 with a horizontal threat
on row 1 and a diagonal follow-up if Player
2 blocks only one line. Therefore the move
is \boxed{column d}.
```

Figure 7: **Representative training prompt.** The full system message contains the complete rules. The user turn supplies the rendered board and legal-action list. The assistant response ends with a boxed legal move.

E MCTS-Mapping Prompt

This appendix records the exact trace-to-tree mapping prompt used for the GPT-5.5 reviewed RQ5 mapping protocol described in Section 6. The completed SPIRAL-family baseline in Figure 5 uses the local explicit-action extractor and therefore does not call this prompt; both routes emit the same search-tree schema. For the LLM-reviewed route, the mapper is GPT-5.5, and the prompt is the search-analysis template referenced by the project skill and instantiated with each turn's task metadata and numbered reasoning paragraphs. The output contract requires strict JSON, constrains root actions to the legal-action list, and labels each paragraph so that the downstream validator can compute the metrics in Section F.

System template: `tree_extraction_system.md`

```
Your task is to convert one model reasoning trace into an explicit search tree.
The trace comes from a board-game turn. Legal root actions are supplied by the
task. A root action is a candidate first move considered by the model. Child
nodes are simulated replies or continuations mentioned by the model.

Return strict JSON only. Do not use Markdown fences. Use this schema:

{
  "turn_id": "...",
  "trees": [
    {"action": "legal action handle", "children": [
      {"action": "reply or continuation handle", "children": []}
    ]}
  ],
  "paragraphs": [
    {
      "para": 0,
      "type": "PREAMBLE",
      "branch_root": null,
      "mentions": []
    }
  ]
}

Allowed paragraph types:
- PREAMBLE: board parsing or threat scanning before a candidate branch
- BRANCH_START: first paragraph proposing one candidate first move
- BRANCH_ANALYSIS: continuation of that candidate branch
- BRANCH_CONCLUSION: local verdict on one branch
- COMPARISON: comparison across multiple candidate first moves
- FINAL_DECISION: final commitment to the chosen move
- META: generic thinking text not tied to a move

Only include a mention when the paragraph simulates a future move. Do not count
coordinates or handles that merely describe the current board state.
```

User template: `tree_extraction_user.md`

````
Task metadata:

```json
{task_json}
```

Numbered reasoning paragraphs:

<trace>
{numbered_paragraphs}
</trace>

Extract the search tree and paragraph labels. Root actions must come from the
task's legal_actions list. The chosen action is included so you can check
whether the model's final decision appears among the candidate first moves.
Return strict JSON only.
````

F Search-Analysis Metrics

This appendix defines the metrics produced by `src/search-analysis` and used by the RQ5 MCTS-mapped analysis. The main paper uses only annotation-derived geometry for Figure 5; raw-trace metrics are diagnostic and are kept in the appendix.

Analysis	Primary artifact	Main metric
RQ–1 transfer	Table 1 and Figure 1	external reasoning average and held-out game weakness of base models
RQ–2 generalization	Figure 3	reasoning and held-out game performance over 5–50 training environments
RQ–2 trace format	Table 1 and Figure 3	SPSD-CoT gain over direct SPSD at matched game count
RQ–4 MCTS depth	Figure 4	transfer change under 32, 64, 128, and 256 teacher simulations
RQ–5 reasoning dynamics	Figure 5	M_{depth}, M_{breadth}, and M_{focus} from annotation-derived search geometry

Table 4: **Metric map.** Primary artifact and metric for each reshaped research question. Raw heuristic metrics such as effective search score, search effort score, inefficiency penalties, and reasoning-nature labels are appendix-only diagnostics.

F.1 Annotation-Derived Tree Metrics

The annotation metrics are computed after either the GPT-5.5 mapper or the local explicit-action extractor returns a JSON object with `trees` and `paragraphs`. The validator normalizes action handles, checks legal root actions against the task metadata, records `annotation_source`, and preserves invalid annotations instead of silently repairing them.

Code key	Definition
`valid`, `errors`, `warnings`	Validation status, hard validation failures, and non-fatal warnings emitted while parsing the annotation.
`total_nodes`	Total number of nodes across all extracted root trees.
`root_breadth`	Number of distinct root trees, i.e., the number of candidate first moves represented in the annotation.
`max_depth`	Maximum depth reached by any extracted root tree, with each root node counted at depth 1.
`leaf_count`	Number of nodes with no children across all extracted trees.
`mean_leaf_depth`	Mean depth of all leaf nodes.
`root_subtree_gini`	Gini coefficient over root-subtree sizes. A larger value means that follow-up simulation is concentrated under fewer root actions.
`legal_action_coverage`	`root_breadth` divided by the number of legal actions supplied in the benchmark task.
`chosen_in_tree`	Boolean indicator that the task's chosen action appears among the extracted root actions.
`chosen_subtree_nodes`	Number of extracted nodes in the subtree rooted at the chosen action, or blank if the chosen action is absent.
`chosen_subtree_depth`	Maximum depth of the subtree rooted at the chosen action, or blank if the chosen action is absent. This is M_{depth} in the main paper.
`paragraph_count`	Number of paragraph labels returned by the mapper.
`comparison_paragraphs`	Number of paragraph labels with type `COMPARISON`.
`final_decision_paragraphs`	Number of paragraph labels with type `FINAL_DECISION`.
`nodes_depth_{d}`	Number of extracted tree nodes at depth `d`.
`mean_branching_depth_{d}`	Mean number of children among nodes at depth `d`.
`paragraph_{type}`	Count of paragraph labels for each observed label type, lower-cased in the output key. Valid types are `PREAMBLE`, `BRANCH_START`, `BRANCH_ANALYSIS`, `BRANCH_CONCLUSION`, `COMPARISON`, `FINAL_DECISION`, and `META`.
`mentions_depth_{d}`	Number of paragraph-level future-move mentions assigned to depth `d`.

Table 5: **Annotation-derived search-tree metrics.** Code-level fields produced by `search_analysis.metrics.compute_metrics`.

F.2 Raw-Trace Heuristic Metrics

The raw metrics operate directly on unannotated model reasoning traces. They are useful as diagnostics for verbosity, candidate enumeration, and marker-based search effort, but they are weaker evidence than annotation-derived tree geometry.

Code key	Definition
M_depth	Per-turn alias for chosen_subtree_depth.
M_breadth	Per-turn alias for legal_action_coverage.
M_focus	Per-turn alias for root_subtree_gini.
annotation_source	Provenance label for the mapper that produced the search-tree JSON, e.g., local_explicit_action_extractor_v1 for the completed SPIRAL baseline.
annotation_valid_turns, annotation_total_turns, annotation_valid_fraction	Number of valid mapped turns, number of candidate turns, and their ratio in the model-game aggregate.
mu_value	Per-turn alias for chosen_q_value when optional trained-MuZero value scoring is configured.
M_depth_mean, M_breadth_mean, M_focus_mean	Mean geometry values after grouping valid annotated turns by model and game, or by model and episode when the episode-level output is requested.
mu_value_mean	Mean mu_value over the same aggregation group.
win_rate	Mean episode win indicator over completed win/loss/draw outcomes in the aggregation group.
n_turns, n_episodes	Number of annotated turns and unique episodes contributing to the aggregated row.
value_score_status	Status of optional value scoring: ok, not_configured, error, unreplayable, or unavailable.
chosen_raw_action, oracle_action, policy_action	Raw chosen action, replayed oracle action, and policy action returned by the configured value scorer.
oracle_agreement	Boolean indicator that the chosen raw action equals the replayed oracle action.
chosen_oracle_rank	Rank of the chosen action among replayed candidate paths, if present.
chosen_q_value	Replayed MuZero Q-value for the chosen action, used as the source of mu_value.
chosen_visit_probability	Replayed root visit probability for the chosen action.
searched_value, predicted_value	Search value and model-predicted value returned by the value-scoring backend.
value_chosen_q_mean	Mean chosen-action Q at the maximum value-sensitivity simulation budget, 50 simulations in the current run.
value_stability_mean_q_range_across_budgets	Mean per-turn range of chosen-action Q across the value-sensitivity budgets, here 8, 25, and 50 simulations.
value_oracle_agreement_rate	Fraction of value-scored turns where the model's chosen action matches the EfficientZero oracle action at the maximum simulation budget.
value_ok_fraction, value_ok_turns, value_turns	Coverage of successful value scoring for the model-game row.
value_stability_high_q_instability_fraction	Fraction of multi-budget scored turns whose chosen-action Q range is at least 0.20.

Table 6: **Paper-geometry aggregation fields.** Code-level fields produced by search_analysis.paper_geometry and optional value scoring.

Code key	Definition
empty_action, illegal_action, forced_action	Enforcement flags from the benchmark turn. They determine the legality_factor.
legal_actions_count	Number of legal actions supplied by the benchmark task.
unique_legal_actions_mentioned	Number of legal actions whose normalized handle appears in the reasoning trace.
legal_actions_mentioned	List of mentioned legal actions.
legal_action_coverage	unique_legal_actions_mentioned divided by legal_actions_count.
chosen_action_mentioned	Boolean indicator that the chosen action appears anywhere in the trace.
chosen_action_final_paragraph	Boolean indicator that the chosen action appears in the final paragraph.
trace_present	Boolean indicator that the reasoning trace is non-empty.
trace_chars, trace_words	Character count and token count under the analyzer's word regex.
paragraph_count	Number of non-empty paragraphs split by blank lines.
sentence_count	Count of sentence delimiters or blank-paragraph delimiters in the trace.
reasoning_len	Raw reasoning-length field copied from the benchmark artifact when present.
repeated_bigram_fraction	One minus the ratio of unique adjacent word bigrams to all adjacent word bigrams; zero for traces with fewer than three words.
board_marker_count, rule_marker_count, candidate_marker_count, future_marker_count, comparison_marker_count, uncertainty_marker_count, self_correction_marker_count, final_marker_count	Counts of marker words or phrases from the marker families in Table 8.
{marker}_per_100_words	Per-100-word density for each marker count, using max(1, trace_words) as the denominator.
reasoning_nature	Rule-based trace category from classify_reasoning_nature; categories and thresholds are shown in Table 9.
outcome_bucket	Episode outcome normalized to win, loss, draw, invalid, exception, unknown, or the original non-empty result value.
episode_result, completion_status, episode_steps, reward_model, move_cap_draw, episode_empty_model_actions, episode_illegal_model_actions, episode_forced_legal_actions	Episode-level outcome and enforcement fields joined from the benchmark episode file.

Table 7: **Raw-trace structure metrics.** Code-level fields produced by search_analysis.raw_trace.analyze_turn before the composite scores are added.

Marker family	Exact markers used by the code
Board	board, grid, row, column, coordinate, cell, square, position, current state.
Rule	rule, objective, legal, illegal, win, draw, loss, capture, phase, turn.
Candidate	option, candidate, move, play, place, placing, choose, select, possibility.
Future	if, then, opponent, respond, reply, next, after, would, could, leads to, result, line, simulate, continuation.
Comparison	better, worse, best, compare, versus, instead, alternative, another, however, on the other hand.
Uncertainty	wait, maybe, perhaps, probably, seems, confusing, ambiguous, unclear, not sure, hard to, i don't know.
Self-correction	wait, actually, let me, double-check, re-read, however, but.
Final	therefore, thus, so the, best choice, correct, answer, choose, select, final.

Table 8: **Raw-trace marker families.** Exact marker lists used for the marker-count metrics in `search_analysis.raw_trace`.

Category	Code condition, evaluated in order
missing_trace	trace_present is false.
direct_answer	trace_words < 20.
confused_loop	trace_words > 450 and either uncertainty_marker_count >= 8 or repeated_bigram_fraction >= 0.18.
forward_search	future_marker_count >= 6 and either comparison_marker_count >= 2 or legal_action_coverage >= 0.35.
candidate_enumeration	legal_action_coverage >= 0.5 or candidate_marker_count >= 6.
rule_recap	rule_marker_count >= 5 and rule_marker_count >= board_marker_count.
board_parse	board_marker_count >= 5.
freeform_deliberation	Default category when none of the earlier conditions apply.

Table 9: **Reasoning-nature categories.** Rule order and thresholds used by `classify_reasoning_nature`.

Code key or helper	Definition
saturate(x, s)	max(0, min(1, x / s)), with invalid or missing values treated as zero.
candidate	max(legal_action_coverage, saturate(candidate_marker_count, 8)).
future	saturate(future_marker_count, 10).
comparison	saturate(comparison_marker_count, 6).
paragraph_structure	saturate(paragraph_count, 8).
final_commitment	saturate(final_marker_count, 4).
decision_focus	0.45 * chosen_action_mentioned + 0.35 * chosen_action_final_paragraph + 0.20 * final_commitment.
structure	0.25 * coverage + 0.20 * future + 0.20 * comparison + 0.15 * candidate + 0.10 * decision_focus + 0.10 * paragraph_structure.
uncertainty_penalty	min(0.45, uncertainty_marker_count / 30).
repetition_penalty	min(0.25, repeated_bigram_fraction).
verbosity_penalty	0.40 for empty traces, 0.25 for traces under 20 words, 0 for 20–1500 words, and min(0.25, (trace_words - 1500) / 6000) above 1500 words.
penalty	min(0.80, uncertainty_penalty + repetition_penalty + verbosity_penalty).
legality_factor	0.20 for empty or illegal actions, 0.50 for forced actions, and 1.0 otherwise.
search_effort_score	100 * structure.
inefficiency_penalty_score	100 * penalty.
effective_search_score and raw_search_score	100 * legality_factor * max(0, structure * (1 - penalty)). The two keys are identical; raw_search_score is retained as a compatibility alias.
decision_focus_score	100 * decision_focus.
efficiency_penalty	min(0.75, verbosity_penalty + 0.5 * uncertainty_penalty + repetition_penalty).
reasoning_efficiency_score	100 * legality_factor * max(0, (0.55 * structure + 0.45 * decision_focus) * (1 - efficiency_penalty)).
search_quality_band	strong_search for scores at least 75, structured_search for at least 55, shallow_search for at least 35, weak_search for positive scores, and no_search_signal otherwise.

Table 10: **Composite raw-search scores.** Formula-level definitions used by `search_analysis.raw_trace.compute_raw_scores`.

Aggregate key pattern	Definition
records	Number of turn-level rows in an aggregate group.
rate_{boolean_key}	Mean of each boolean field over the group for trace_present, empty_action, illegal_action, forced_action, chosen_action_mentioned, and chosen_action_final_paragraph.
mean_{numeric_key}	Mean of every numeric non-boolean metric present in the group.
median_raw_search_score, median_reasoning_efficiency_score, median_decision_focus_score	Median values additionally emitted for the three score keys named by the code.
game_balanced_mean_{metric} and game_balanced_median_{metric}	Mean and median over per-game/outcome means for the headline metrics: effective_search_score, raw_search_score, search_effort_score, inefficiency_penalty_score, decision_focus_score, reasoning_efficiency_score, legal_action_coverage, trace_words, repeated_bigram_fraction, uncertainty_marker_count, future_marker_count, comparison_marker_count, and unique_legal_actions_mentioned.
win_mean_{metric}, loss_mean_{metric}, win_minus_loss_{metric}, loss_minus_win_{metric}	Outcome-delta fields emitted when both win and loss rows are available for the same model and reasoning mode.
games, turns, win_games, loss_games, win_turns, loss_turns	Group sizes used by the game-balanced headline and outcome-delta summaries.

Table 11: **Raw-analysis aggregate fields.** Code-level summary fields produced by `summarize_raw_metrics`, `aggregate_rows`, `compute_game_balanced_headline`, and `compute_outcome_deltas`.