DIRECT OPTIMIZATION OF GENERATORS FOR SEARCH IN AUTOMATED THEOREM PROVING

Anonymous authors
Paper under double-blind review

ABSTRACT

Fine-tuned Large Language Models (LLMs) significantly advance Automated Theorem Proving (ATP), but are often deployed as guiding policies within tree search rather than for single-attempt generation. Recent work shows cross entropy is suboptimal for an LLM used in flat search strategies such as aggregation or filtering and that work has developed new loss functions to correct this misalignment. Extending this alignment to tree search is more challenging: proof discovery depends on exploration and recovery through off-trace states that supervised demonstrations do not reveal. We extend Compute-Aligned Training (CAT) to this setting through an abstraction of policy-guided search, deriving tractable, trace-supported losses. Alongside these search-aware losses, we introduce a search-agnostic uniform-allocation (UA) loss that accounts for the budget without specifying the specific search. Both induce scalar weights on per-tactic cross-entropy gradients. We characterize how off-trace behavior affects the search-aware weights, including conditions for vanishing approximation error at large budgets. On a Lean benchmark, both approaches achieve higher observed proof-success rates than cross-entropy across six search strategies, with strong results from a single shared UA adapter. Budget sweeps show larger gains over cross-entropy at 256 than at 16 expansions, implying CAT scales with test time compute.

1 INTRODUCTION

Automated Theorem Proving (ATP) is an important frontier in Artificial Intelligence (AI) (Yang et al., 2024), with applications in software and hardware verification (Bengio & Malkin, 2024). Formal verifiers such as Lean 4 (de Moura & Ullrich, 2021) allow generated proofs to be checked, but proof generation remains difficult because rewards are sparse and horizons are long (Poesia et al., 2024). Large Language Models (LLMs) predict proof steps (Polu & Sutskever, 2020), yet a single flawed step can invalidate an entire attempt. Recent approaches therefore pair LLMs with structured search (Lample et al., 2022; Polu et al., 2022), which retains partial proofs and explores alternative continuations rather than requiring every step to succeed on the first attempt.

LLMs driving search are typically trained through Supervised Fine-Tuning (SFT) with Cross-Entropy (CE). Recent work shows that CE can be suboptimal when deployment uses search (Ousherovitch & Tewari, 2026; Chen et al., 2025), but existing corrections focus on flat strategies such as candidate aggregation (Wang et al., 2023) or filtering (Chen et al., 2021; Cobbe et al., 2021). Extending alignment to tree search is fundamentally more difficult because success probability includes unsuccessful exploration followed by recovery, and depends on how compute is allocated across intermediate proof states. An exact objective therefore requires off-trace continuations that demonstrated proof traces do not supply.

We address this gap by formalizing policy-guided search and constructing tractable Compute-Aligned Training (CAT) losses for this kind of search. Given a demonstrated proof trace, we restrict the modeled search to observed states and collapse deviations into a trace-miss event. This yields *search-aware* objectives that preserve selected aspects of the strategy's compute allocation. Their gradients are weighted sums of the usual per-tactic CE gradients, emphasizing steps according to their influence on modeled proof success.

Alongside these objectives, we introduce a simple *search-agnostic* uniform-allocation (UA) loss. UA assigns a uniform local compute budget to each demonstrated step, accounting for repeated

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

sampling without specifying the deployed search rule. It thus provides a shared training objective for use across search algorithms, rather than an approximation tied to one particular deployed strategy.

Trace support introduces approximation error because demonstrations omit alternative proofs and the compute consumed by off-trace exploration. We characterize these effects through the per-tactic gradient weights and identify when search-aware weights scale better than CE with more budget.

On a Lean benchmark, both strategy-specific CAT and UA attain higher accuracy than an epoch-matched CE control under all six evaluated strategies (Table 1). A shared UA adapter performs strongly across strategies, while search-aware CAT yields an additional observed gain under Pass@N. Experiments also show larger gains over CE at $N = 256$ than at $N = 16$ (Table 2).

1.1 Related Work

Automated Theorem Proving. The application of deep learning to ATP has rapidly evolved alongside LLMs (Polu & Sutskever, 2020; Han et al., 2022). Modern neural provers target interactive verification environments like Lean (de Moura & Ullrich, 2021; Zheng et al., 2022), with open-weight milestones such as Lean Copilot (Song et al., 2025), Llemma (Azerbayev et al., 2024), and ReProver (Yang et al., 2023) powered almost exclusively by SFT. Our work is strictly situated within this SFT paradigm. We explicitly scope our analysis away from systems that interleave informal natural-language reasoning with formal verification (Jiang et al., 2023; Wu et al., 2022), as well as online Reinforcement Learning (RL) pipelines (AlphaProof and AlphaGeometry teams, 2024).

Test-Time Search Algorithms. The use of search in ATP falls in the wider trends of utilizing inference-time compute (Snell et al., 2024). Search in ATP is diverse. It includes uninformed traversal methods like Depth-First Search (DFS) (Pearl, 1984) or Pass@N (Chen et al., 2021), and the more complex policy-guided searches like Best-First Search (BFS) (Hart et al., 1968), most notably Levin Search and its modern neural adaptations (Orseau et al., 2018; Orseau & Lelis, 2021; Xin et al., 2025). While prior literature has extensively benchmarked these algorithms in isolation, it treats the language model as a static, search-agnostic prior. Crucially, the literature has never investigated how the structural assumptions of these distinct search rules interact with the training of the policy itself.

Test-Time Aligned Training. A growing body of work recognizes that standard supervised objectives fail to align models with inference-time search (Ousherovitch & Tewari, 2026; Chen et al., 2025; Chow et al., 2025; Balashankar et al., 2025; Tang et al., 2025). However, these efforts focus overwhelmingly on non-branching strategies where the global search objective can be expressed as an exact function of observed generation probabilities. Within ATP, the sole existing attempt at test-time aligned training is Direct Coverage Optimization (DCO) (Chen et al., 2025), which focused exclusively on Pass@N. Because the global success probability of Pass@N can be expressed as a closed-form function of on-path tactic probabilities, DCO sidesteps the structural approximation problem entirely. By contrast, state-of-the-art theorem provers rely on branching strategies whose objectives cannot be resolved purely from on-path data. This paper bridges that gap.

1.2 Contributions

- **A Framework for Search-Aligned Training.** We formalize policy-guided search and derive tractable losses for branching strategies. We also develop a budget-based UA loss independent of the specific search's structure. Both approaches induce scalar weights on per-tactic CE gradients.
- **Characterizing the Trace-Supported Approximation.** For Pass@N and best-first search, we analyze how unobserved alternative proofs and off-trace compute consumption affect search-aware gradient weights, including their dependence on the test-time budget.
- **Empirical Evaluation in Lean.** Across six search strategies, strategy-specific CAT and UA each attain higher success rates than CE. We also evaluate gains across deployment budgets.

2 A General Objective for Search-Aligned Training

This section formalizes LLM-guided search and derives a general aligned objective and the search-agnostic UA objective. We assume familiarity with Interactive Theorem Proving (ITP).[1]

[1] A full formalization is provided in Section B.

2.1 SETUP

Let the environment be a state space $\mathcal{X}$ (proof states), an action space $\mathcal{Y}$, and a transition $\mathcal{T} : \mathcal{X} \times \mathcal{Y} \to \mathcal{X} \cup \{x_{err}\}$, where x_{err} is the state produced by incorrect tactics. The LLM defines a stochastic policy $\pi_\theta(y \mid x)$. A dataset $\mathcal{D}$ provides valid traces; a single trace is

$$\tau^* = (x_0, y_1^*, x_1, \ldots, x_{L-1}, y_L^*, x_L), \qquad x_L = x_{QED},$$

where each demonstrated tactic y_t^* moves the environment from x_{t-1} to x_t. Write $p_t = \pi_\theta(y_t^* \mid x_{t-1})$. Standard SFT with CE maximizes the likelihood of these traces:

$$L_{CE}(\theta) = -\sum_{t=1}^{L} \log \pi_\theta(y_t^* \mid x_{t-1}) = -\sum_{t=1}^{L} \log p_t, \qquad L_{CE}^{(t)} := -\log p_t. \tag{1}$$

At test time, the policy is deployed inside a search algorithm $\mathcal{A}$ with a compute budget. We wish to optimize the probability that $\mathcal{A}$, using π_θ, finds a proof within that budget.

2.2 SEARCH AS CONFIGURATION DYNAMICS

$\mathcal{A}$ is defined by its interaction with memory, which we define via *configurations*: $c = \langle \mathcal{F}, M \rangle$.

The frontier $\mathcal{F}$ holds the nodes available for expansion with a heuristic score (insertion order, internal confidence, etc.); its elements are tuples (x, s) with $x \in \mathcal{X}$, $s \in \mathbb{R}$. We may write $x \in \mathcal{F}$ for $(x, s) \in \mathcal{F}$ for notational ease. M holds everything else: the policy, the *remaining budget*, the unavailable nodes, etc. Budget is most naturally interpreted as either tactic expansions or rollouts. Budget handling can be encoded in the specification of the strategy.

$\mathcal{A}$ selects a node $x \in \mathcal{F}$, samples a tactic $y \sim \pi_\theta(\cdot \mid x)$, executes it, and updates its configuration from the observation $z = \mathcal{T}(x, y)$. Excluding the initialization of c_0 from x_0 and the compilation of the returned proof from M, $\mathcal{A}$ is fully specified by two operations,

$$S_{\mathcal{A}}(x \mid c), \qquad U_{\mathcal{A}}(c, x, z),$$

a (possibly stochastic) selection rule $S_{\mathcal{A}}$ and a deterministic update rule $U_{\mathcal{A}}$.

A configuration is *solved* if it contains a completed proof, and *dead* if its budget is exhausted. Let $J^{\mathcal{A}}(c)$ be the true probability that $\mathcal{A}$, started from c, ever discovers a proof. It satisfies

$$J^{\mathcal{A}}(c) = \begin{cases} 1 \text{ if } c \text{ solved;} \quad 0 \text{ if } c \text{ dead (and unsolved),} \\ \sum_{x \in \mathcal{F}(c)} S_{\mathcal{A}}(x \mid c) \sum_{y \in \mathcal{Y}} \pi_\theta(y \mid x)\, J^{\mathcal{A}}\big(U_{\mathcal{A}}(c, x, \mathcal{T}(x, y))\big), & \text{otherwise} \end{cases} \tag{2}$$

and the deployed objective is $\mathcal{J}^{\mathcal{A}}(\theta) = J^{\mathcal{A}}(c_0)$, where c_0 encodes the start state x_0 and the full budget. Returning any valid trajectory from x_0 to x_{QED} counts as success.

Equation (2) is exact but cannot be evaluated from a supervised trace. Consider expanding a frontier node on the trace, say $x = x_{t-1}$. The tactic space partitions into three disjoint sets: the demonstrated tactic $\{y_t^*\}$, the invalid tactics $\mathcal{I}_t := \{y : \mathcal{T}(x_{t-1}, y) = x_{err}\}$, and the valid but off-trace tactics $\mathcal{O}_t := \{y : \mathcal{T}(x_{t-1}, y) \notin \{x_{err}, x_t\}\}$. With off-trace states defined as $x_{t,y} = \mathcal{T}(x_{t-1}, y)$ for $y \in \mathcal{O}_t$, the sum in Equation (2) then expands to:

$$\begin{aligned} &\sum_y \pi_\theta(y \mid x_{t-1})\, J^{\mathcal{A}}\big(U_{\mathcal{A}}(c, x_{t-1}, \mathcal{T}(x_{t-1}, y))\big) = p_t\, J^{\mathcal{A}}\big(U_{\mathcal{A}}(c, x_{t-1}, x_t)\big) \\ &+ \sum_{y \in \mathcal{I}_t} \pi_\theta(y \mid x_{t-1})\, J^{\mathcal{A}}\big(U_{\mathcal{A}}(c, x_{t-1}, x_{err})\big) + \sum_{y \in \mathcal{O}_t} \pi_\theta(y \mid x_{t-1})\, J^{\mathcal{A}}\big(U_{\mathcal{A}}(c, x_{t-1}, x_{t,y})\big). \end{aligned} \tag{3}$$

The dataset supplies neither probabilities within $\mathcal{I}_t \cup \mathcal{O}_t$ nor continuations $J^{\mathcal{A}}(U_{\mathcal{A}}(c, x_{t-1}, x_{t,y}))$ containing unobserved off-trace states. Off-trace tactics may lead to alternate proofs or rejoin the trace so the exact objective depends on the search space outside of our dataset.

2.3 THE TRACE-SUPPORTED SURROGATE

To obtain a tractable objective, we restrict the search to states from the demonstrated trace. Let $\widehat{\mathcal{C}}_{\tau^*}$ be the configurations whose frontier and memory reference only $x_0, \ldots, x_L$. We replace the true transition by an offline version,

$$\widehat{\mathcal{T}}_{\tau^*}(x_{t-1}, y) = \begin{cases} x_t, & y = y_t^*, \\ \bot_\tau, & y \neq y_t^*, \end{cases} \tag{4}$$

where $\perp_\tau$ marks any deviation from the trace: valid off-trace and invalid tactics are collapsed into one training-time event. This transition induces two projected update rules,

$$\widehat{U}^+_{\mathcal{A}}(\widehat{c}, x), \qquad \widehat{U}^-_{\mathcal{A}}(\widehat{c}, x),$$

the positive update when the selected node advances $x_{t-1} \to x_t$, and the negative update following a trace miss. Both updates map to $\widehat{\mathcal{C}}_{\tau^*}$. At a demonstrated state, the model either samples y^*_t with probability p_t or deviates with probability $1 - p_t$.

Trace-supported recurrence. The surrogate success probability is

$$\widehat{J}_{\mathcal{A}}(\widehat{c}; p) = \begin{cases} 1 \text{ if } \widehat{c} \text{ solved}; \quad 0 \text{ if } \widehat{c} \text{ dead}, \\ \sum_{x \in \widehat{\mathcal{F}}(\widehat{c})} S_{\mathcal{A}}(x \mid \widehat{c}; p) \Big[p_{t(x)}\, \widehat{J}_{\mathcal{A}}\big(\widehat{U}^+_{\mathcal{A}}(\widehat{c}, x); p\big) + (1 - p_{t(x)})\, \widehat{J}_{\mathcal{A}}\big(\widehat{U}^-_{\mathcal{A}}(\widehat{c}, x); p\big) \Big], \quad \text{else} \end{cases} \tag{5}$$

where $t(x)$ is the index of x in τ^*.

We call this loss Compute Aligned Training (CAT), following Ousherovitch & Tewari (2026)

$$\widehat{\mathcal{J}}^{\mathcal{A}}_{\tau^*}(\theta) = \widehat{J}_{\mathcal{A}}\big(\widehat{c}_0;\, p(\theta)\big), \qquad \widehat{L}^{\mathcal{A}}_{CAT}(\theta) = -\log \widehat{\mathcal{J}}^{\mathcal{A}}_{\tau^*}(\theta), \tag{6}$$

where $\widehat{c}_0$ encodes x_0 and the initial budget. We will assume the recursion is finite.[2]

2.4 The CAT Gradient

The following Proposition shows search strategy can be completely specified for training by its triple $(S_{\mathcal{A}}, \widehat{U}^+_{\mathcal{A}}, \widehat{U}^-_{\mathcal{A}})$, and that its loss and per-tactic gradient weights follow mechanically.

Proposition 1 (CAT Loss)**.** *Fix a strategy $\mathcal{A}$ given by $(S_{\mathcal{A}}, \widehat{U}^+_{\mathcal{A}}, \widehat{U}^-_{\mathcal{A}})$. Then:*

(i) Equation (5) *has a unique solution, and $\widehat{\mathcal{J}}^{\mathcal{A}}_{\tau^*}(\theta)$ depends on θ only through $p = (p_1, \ldots, p_L)$.*[3]

(ii) The CAT gradient is a weighted sum of per-step cross-entropy gradients,

$$\nabla_\theta \widehat{L}^{\mathcal{A}}_{CAT}(\theta) = \sum_{t=1}^{L} \widehat{w}^{\mathcal{A}}_t \nabla_\theta L^{(t)}_{CE}, \qquad \widehat{w}^{\mathcal{A}}_t = \frac{p_t}{\widehat{\mathcal{J}}^{\mathcal{A}}_{\tau^*}} \frac{\partial \widehat{\mathcal{J}}^{\mathcal{A}}_{\tau^*}}{\partial p_t}. \tag{7}$$

Proof. (i) Under the assumption the budget is finite and decrements with each update, it terminates after finitely many steps and assigns a unique value to each configuration. All θ-dependence enters only through $p_{t(v)}$ or possibly through the scores $s_{\mathcal{A}}(\cdot; p)$.

(ii) By (i), $\nabla_\theta \widehat{\mathcal{J}}^{\mathcal{A}}_{\tau^*} = \sum_t \partial_{p_t} \widehat{\mathcal{J}}^{\mathcal{A}}_{\tau^*}\, \nabla_\theta p_t$. From $L^{(t)}_{CE} = -\log p_t$ we have $\nabla_\theta p_t = -p_t\, \nabla_\theta L^{(t)}_{CE}$, so

$$\nabla_\theta \big[-\log \widehat{\mathcal{J}}^{\mathcal{A}}_{\tau^*} \big] = -\frac{1}{\widehat{\mathcal{J}}^{\mathcal{A}}_{\tau^*}} \sum_{t=1}^{L} \frac{\partial \widehat{\mathcal{J}}^{\mathcal{A}}_{\tau^*}}{\partial p_t} \big(-\, p_t\, \nabla_\theta L^{(t)}_{CE} \big) = \sum_{t=1}^{L} \frac{p_t}{\widehat{\mathcal{J}}^{\mathcal{A}}_{\tau^*}} \frac{\partial \widehat{\mathcal{J}}^{\mathcal{A}}_{\tau^*}}{\partial p_t}\, \nabla_\theta L^{(t)}_{CE}. \qquad \square$$

As noted by Chen et al. (2025) for the case of Pass@N, separating $\widehat{w}^{\mathcal{A}}_t$ lets one filter low-weight gradients before they corrupt batch normalization statistics, stabilizing training.

2.5 A Search-Agnostic Uniform-Allocation Objective

The same gradient form permits a compute-aware objective without specifying $\mathcal{A}$. For an expansion budget $N \geq L$, UA assigns each demonstrated step a uniform local budget $\bar{K} = N/L$ and uses

$$\widehat{\mathcal{J}}^{\mathrm{UA}}(p; N) = \prod_{t=1}^{L} \big[1 - (1 - p_t)^{\bar{K}} \big], \qquad \widehat{L}^{\mathrm{UA}}_{CAT} = -\log \widehat{\mathcal{J}}^{\mathrm{UA}}. \tag{8}$$

[2] Sufficient conditions for this include decrements of budget every expansion or decrements every rollout and a finite demonstrated trace.

[3] Differentiability naturally follows if selection is smooth in θ.

For integer $\bar{K}$, this is the probability that every demonstrated step succeeds within its own quota of independent attempts, without transferring unused quota between steps. For noninteger N/L, we use the continuous extension of the formula. This allocation is a modeling choice, not a claim about how a particular deployed algorithm distributes its budget. By Proposition 1,

$$\widehat{w}_t^{\mathrm{UA}} = \frac{\bar{K} p_t (1-p_t)^{\bar{K}-1}}{1-(1-p_t)^{\bar{K}}}. \tag{9}$$

The rule depends on N, L, and p_t, but not on $\mathcal{A}$. At $N = L$ it recovers CE; additional local attempts reduce the emphasis on tactics already likely to be sampled. Thus UA is search-agnostic, but not compute-agnostic. Its connection to the shared-budget formulation is detailed in Section J.

3 Strategy-Specific Objectives

We instantiate the construction for Pass@N and best-first search using Proposition 1.[4]

3.1 Pass@N

Pass@N uses up to N independent rollouts (Figure 3); here N counts *rollouts*, not tactic expansions. In the trace-supported model, let $\widehat{c} = (r, t)$, where r counts remaining rollouts, including the current one, and t indexes the next demonstrated tactic. Advancement gives $(r, t+1)$; a miss abandons the rollout and gives $(r-1, 1)$. Thus Equation (5) becomes

$$\widehat{J}(r,t) = p_t \widehat{J}(r, t+1) + (1-p_t)\widehat{J}(r-1, 1), \tag{10}$$

with $\widehat{J}(r, L+1) = 1$ and $\widehat{J}(0, t) = 0$ for $t \le L$. A rollout completes the demonstrated trace with probability $\pi_{\mathrm{proof}} := \prod_{t=1}^{L} p_t$, giving

$$\widehat{\mathcal{J}}_{\tau^*}^{\mathrm{Pass@}N}(\theta) = 1 - (1-\pi_{\mathrm{proof}})^N. \tag{11}$$

Applying Proposition 1 yields the same weight at every step of the trace:

$$\widehat{w}_t^{\mathrm{Pass@}N} = \frac{N \pi_{\mathrm{proof}} (1-\pi_{\mathrm{proof}})^{N-1}}{1-(1-\pi_{\mathrm{proof}})^N}.$$

3.2 Best-First Search

We consider policy-guided best-first search (BFS) (Orseau et al., 2018; Orseau & Lelis, 2021; Xin et al., 2025). A node x_k reached through tactics $y_1, \ldots, y_k$, $k \ge 1$, is scored by

$$s_{\mathrm{BFS}}(x_k; \pi_\theta) = \left(\prod_{j=1}^{k} \pi_\theta(y_j \mid x_{j-1}) \right)^{1/k}. \tag{12}$$

The algorithm pops the highest-scoring frontier node, samples B tactics there, and inserts valid children (Figure 4). Here N counts *tactic expansions*: each sample costs one unit.

Trace-supported instantiation. We use a retry surrogate with configuration $\widehat{c} = (b, t)$, where b is the remaining expansion budget and the only live node is x_{t-1}. Starting from $(N, 1)$, advancement gives $(b-1, t+1)$ and a miss gives $(b-1, t)$. Retaining the node after a miss is a modeling choice: this surrogate replaces the deployed finite per-pop sampling with retries until the demonstrated tactic is sampled or the budget is exhausted. Its singleton frontier also removes score-based competition. The recurrence is

$$\begin{aligned} \widehat{J}_{\mathrm{BFS}}(b,t) &= p_t \widehat{J}_{\mathrm{BFS}}(b-1, t+1) + (1-p_t)\widehat{J}_{\mathrm{BFS}}(b-1, t), \\ \widehat{J}_{\mathrm{BFS}}(b, L+1) &= 1, \qquad \widehat{J}_{\mathrm{BFS}}(0,t) = 0 \quad (t \le L). \end{aligned} \tag{13}$$

[4] Objectives for other strategies are given in Sections D and E.

Objective. Let $T_t \sim \mathrm{Geom}(p_t)$ independently count the samples needed at step t, including the successful sample, and write $S_L = \sum_{t=1}^{L} T_t$. For $N \geq L$, the surrogate success probability is

$$\widehat{\mathcal{J}}^{\mathrm{BFS}}_{\tau^*}(\theta) = \mathbb{P}(S_L \leq N). \tag{14}$$

The shared budget couples the per-tactic weights, which follow by differentiating Equation (13). The connection to UA's fixed local allocation is detailed in Section J.

4 Approximation Error

A demonstrated trace tells us how one proof succeeds, but not what happens when search leaves that path. At state x_{t-1}, we can compute the probability of missing the demonstrated tactic. This section intends to characterize what is lost in not having information about this event at training. We study this missing information through two channels: alternative proofs (the *bypass channel*) and compute spent on unsuccessful off-trace exploration (the *trap channel*). How do these unseen consequences change the importance of sampling the demonstrated tactic? Full derivations are in Section C.[5]

4.1 Effects of the Channels on the True Weight

Bypass: missing the trace isn't failing. A different tactic may lead to a valid proof without completing τ^*. The surrogate cannot see these successes. For Pass@N, accounting for alternative proofs reduces the deployed weight relative to the trace-supported weight.

Theorem 1 (Bypass Deflation)**.** *Under Assumption 1, for Pass@N with $N > 1$ and every step t,*

$$w_t^\star \leq \widehat{w}_t^{\mathrm{Pass@}N} < 1 = w_t^{\mathrm{CE}}. \tag{15}$$

Consequently, $|\widehat{w}_t^{\mathrm{Pass@}N} - w_t^\star| < |1 - w_t^\star|$.

The proof and decomposition of the missing alternative-success contributions are given in Theorem 4 and eq. (25).

Trap: a miss can cost much more than one attempt. A valid but unhelpful tactic can lead search into a branch that consumes many expansions before the demonstrated state is selected again. Unlike an immediate rejection, this excursion leaves less budget for completing the proof. Our approximation understates the cost of such misses.

To isolate this effect, we censor alternative proofs, so only completion of τ^* counts as success while off-trace exploration remains possible (Lemma 4). We quantify the resulting upward pressure on weights in a model with a common deterministic return cost.

Theorem 2 (Shared-Budget Bracketing under Deterministic Excursions)**.** *Under Assumption 2, let $N \geq L$ be an integer. Every miss costs $1 \leq \kappa \leq \bar{\kappa} < \infty$ expansions before returning to the same trace state; successful trials cost one, and κ is independent of p. Then*

$$w_t^{\mathrm{trap}}(N) \in \left[\widehat{w}_t(N),\ \widehat{w}_t\left(L + \frac{N-L}{\bar{\kappa}}\right)\right] \subseteq [\widehat{w}_t(N), 1], \tag{16}$$

where $w_t^{\mathrm{trap}} = w_t(J^{\mathrm{trap}})$ and noninteger budget arguments are rounded down.

The mechanism is a reduction in effective budget to $N_{\mathrm{eff}} = L + (N-L)/\kappa$ (Proposition 3). The surrogate assumes $\kappa = 1$: a miss costs one expansion. At the other extreme, if every miss prevents recovery, success requires following the trace without mistakes, recovering CE's unit weights. Finite excursion costs interpolate between these cases within this model.

What the trace cannot determine. Two environments can agree on every demonstrated transition, all p_s, the search rule, and the budget, yet differ in what a miss causes. When those off-trace differences produce distinct deployed weights, no rule using only these trace-level inputs can be exact in both environments (Proposition 4). Thus, solving the trace-supported recurrence exactly does not resolve the missing information about recovery.

[5] Throughout, $p_s \in (0,1)$ and $w_t(J) = p_t \partial_{p_t} \log J$ for differentiable $J > 0$. Derivatives follow Equation (19): all other demonstrated probabilities, conditional distributions over non-demonstrated tactics, and off-trace policy conditionals are held fixed.

4.2 Large-Compute Scaling: Can Search Recover from a Miss?

More compute can compensate for a delay, but not for permanent loss of the opportunity to recover. At $N = L$ in the trap-only model, there is no room for wasted expansions: success requires every demonstrated tactic on its first attempt, and both the deployed and surrogate weights equal one. Larger budgets create opportunities to recover, making the outcome depend on whether excursions eventually return. The following limits hold with the policy and excursion parameters fixed.

Theorem 3 (Large-Budget Limits and Recoverability)**.** *(i)* ***Pass@N.*** *Under Assumption 1,*

$$w_t^\star(N) \longrightarrow 0, \qquad \widehat{w}_t(N) \longrightarrow 0.$$

Hence the search-aware weight error tends to zero, while CE's weight error tends to one.

(ii) ***Shared-budget excursions.*** *Under Assumption 3, each miss at step s independently causes permanent absorption with probability $q_s \in [0, 1)$; otherwise it incurs a bounded excursion before returning to the same trace state. Only the demonstrated proof counts as success. Holding q_s and the finite-excursion laws fixed under differentiation,*

$$\widehat{w}_t(N) \longrightarrow 0, \qquad w_t^{\mathrm{trap}}(N) \longrightarrow \omega_t := \frac{q_t}{p_t + (1 - p_t)q_t}.$$

See Propositions 5 and 6 for the derivations.

When all excursions are recoverable ($q_s = 0$), both deployed and surrogate weights eventually vanish: avoiding a particular miss matters less when search has enough budget to try again. With permanent absorption, avoiding the miss retains value even with unlimited compute. The retry surrogate overlooks this risk, leaving limiting error ω_t; CE's limiting error is $1 - \omega_t$.

The analysis therefore identifies what trace-only training leaves unresolved: alternative routes to success, the compute needed to recover, and the possibility of never recovering. These quantities can be modeled, but are not determined by the demonstrated trace. Additional assumptions can specify them, and training-time search offers a way to collect evidence about them. Neither is replaced by evaluating the same trace-supported surrogate more exactly.

5 Experiments

We test whether accounting for budgeted search during SFT improves proof discovery, whether matching the training objective to the deployed strategy helps, and how the resulting gains vary with test-time budget. We compare search-aware CAT and search-agnostic UA with a CE control.

Setup. We use Lean 4 through LeanDojo (Yang et al., 2023), training on `leandojo-benchmark-4-random`, a random split of mathlib4 (The mathlib Community, 2020). The held-out pool contains 458 theorems whose reference proofs have 2–5 steps; any kernel-verified proof counts as success. We initialize from Qwen2.5-Math-7B-Instruct (Qwen Team, 2024) and fine-tune with LoRA (Hu et al., 2021). Within each experiment, models share a CE warm-up.[6] The control then receives one additional CE epoch, while the search-aware and UA models use one additional epoch with their respective weights. Data, ordering, and optimizer settings are otherwise identical. Evaluation budgets count node expansions, each comprising one tactic sample and its kernel verification.[7] Full details are in Section G.

5.1 Experiment 1: Training for Search Across Strategies

At $N = 256$ expansions, we evaluate Pass@N, BFS, DFS variants (Section D), and Monte Carlo Tree Search (MCTS) (Kocsis & Szepesvári, 2006).[8] For each deployed strategy, we compare CE, the corresponding search-aware weights, and the same single shared UA adapter. This study also applies wall-clock caps; capped runs without a proof count as failures (Section G.3).

[6] Two warm-up epochs for Section 5.1 and one for Section 5.2.

[7] The Pass@N objective in Section 3.1 counts rollouts. We use $N/4$ as its training parameter, a fixed approximation to the expansion budget rather than an exact conversion.

[8] Our MCTS implementation uses the policy's Levin score as its value estimate.

Table 1: **Proofs found (%) at $N = 256$ on 458 held-out theorems.** Columns are deployed strategies. The same UA adapter is used throughout. * marks $p < 0.05$ against CE under the same strategy; bold marks the highest reported rate.

Training objective	Pass@N	BFS	DFS	RDFS	VDFS	MCTS
CE	21.2	23.8	19.2	20.7	21.2	20.3
Search-aware CAT	**26.6***	25.1	20.7	23.6*	23.8	22.5
Shared UA	25.1*	**26.9**	**23.6***	**25.3***	**24.5***	**25.1***

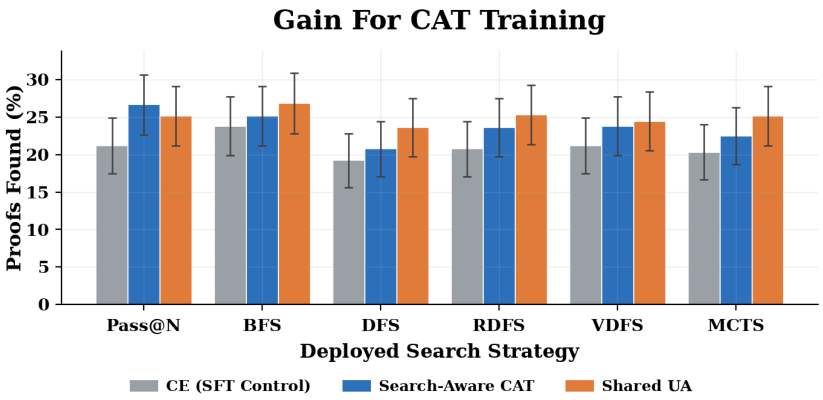

Figure 1: **Both training families improve accuracy over CE.** Proofs found (%) on 458 held-out theorems with a budget of $N = 256$ tactic expansions. Search-aware CAT uses strategy-matched training objectives. UA uses one adapter across all strategies. UA has the highest accuracy under branching strategies; search-aware CAT leads under Pass@N. Error bars show 95% intervals.

Results Both search-aware CAT and UA have higher observed success rates than CE under all six strategies (Table 1), showing that accounting for deployment-time compute provides substantial gains over standard SFT. The shared UA objective is a strong search-agnostic approach, attaining the highest point estimate under the five branching strategies. At the same time, search-aware CAT achieves the highest observed success under Pass@N (26.6% versus 25.1% for UA), showing modeling the deployed search strategy can yield additional improvement. Consistent with this, within the search-aware family, matching training and deployment gives higher point estimates for the Pass@N/BFS adapter pair: 26.6% versus 23.7% under Pass@N, and 25.1% versus 24.5% under BFS (Table 3). Together, these results support compute-aware training and suggest that modeling the deployed strategy can provide additional benefits.

5.2 Experiment 2: Alignment Gain versus Test-Time Budget

We train a separate search-aware CAT adapter for each strategy (Pass@N or BFS) and budget $N \in \{16, 64, 256\}$, and evaluate each adapter at its corresponding budget. This sweep removes the per-strategy and per-theorem wall-clock caps, but each model receives at most 80 hours of evaluation on four A100 GPUs. At each budget, we compare CAT and CE only on theorems for which both completed evaluation; Table 2 reports the paired subset sizes. $N = 1024$ runs completed only 56–57 paired theorems and are reported separately in Section H.

Gains persist across the evaluated budgets. Search-aware CAT has positive paired gains at every reported budget in Table 2. The observed gains at $N = 256$ exceed those at $N = 16$ for both Pass@N (+4.2 versus +2.0 points) and BFS (+6.9 versus +4.4). However, these are finite-budget performance comparisons, not direct measurements of the weight-error limits in Section 4.

6 Conclusion

This paper studies how to train an LLM from demonstrations when it will be deployed within policy-guided search under a finite compute budget. We develop two complementary approaches:

Table 2: **Paired search-aware CAT gain over CE.** n is the paired subset size; b and c count CAT-only and CE-only successes. Gains are in percentage points. The 95% intervals use the variance approximation $(b + c)/n^2$ on the probability scale; p is the exact two-sided McNemar test.

Strategy	Budget N	n	b/c	Gain (pp), 95% interval	p
Pass@N	16	458	10/1	$+2.0$ $[0.5, 3.4]$	0.012
	64	458	28/8	$+4.4$ $[1.8, 6.9]$	0.001
	256	408	29/12	$+4.2$ $[1.1, 7.2]$	0.012
BFS	16	458	28/8	$+4.4$ $[1.8, 6.9]$	0.001
	64	458	34/13	$+4.6$ $[1.7, 7.5]$	0.003
	256	407	37/9	$+6.9$ $[3.6, 10.1]$	< 0.001

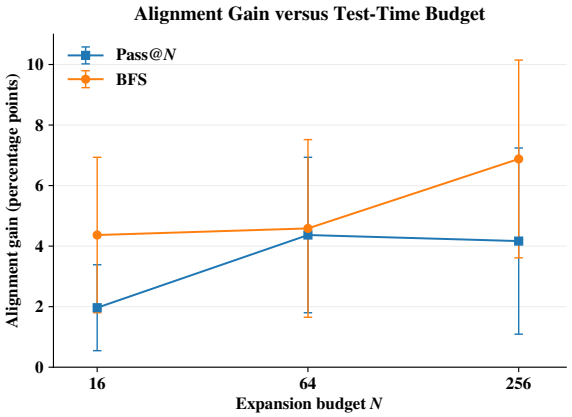

Figure 2: Paired search-aware CAT gain over CE with budget-matched adapters. Error bars are 95% intervals. Completed subsets differ across budgets, so connecting lines are descriptive rather than a common-cohort comparison.

search-aware CAT objectives that incorporate a model of the search strategy, and a search-agnostic uniform-allocation objective that accounts for compute without specifying the search rule. Both induce scalar weights on the usual per-tactic CE gradients. Although our experiments focus on ATP, the framework provides a starting point for studying other settings in which generative models guide tree search. Our analysis characterizes how alternative proofs and off-trace exploration affect search-aware gradient weights. It identifies conditions under which their approximation error vanishes as the test-time budget grows while CE remains misaligned. Empirically, all objectives attain higher observed proof-success rates than an epoch-matched CE control across all six evaluated strategies. The shared UA objective performs strongly across search algorithms, while search-aware CAT achieves the highest observed success under Pass@N, demonstrating that incorporating strategy-specific structure can, in some cases, provide an additional benefit beyond generic compute awareness. Budget sweeps additionally show larger gains over CE at 256 than at 16 expansions for both tested strategies. Together, these findings show that considering the downstream search during training can improve performance.

Limitations and Future Work. Our experiments use one model and short reference proofs, with training restricted to offline SFT. UA accounts for compute without modeling the search rule; search-aware CAT models selected search dynamics on the demonstrated trace. Both remain trace-only. Additional training-time compute could sample one-step deviations, explore local trees around demonstrations, and increasingly approximate deployment search. One-step samples reveal alternative transitions; deeper searches provide evidence about alternative proofs and recovery costs. Such observations could refine search-aware objectives beyond fixed allocation. Testing whether these refinements improve performance, and how to allocate training compute between supervision and search, remain future work.

REFERENCES

AlphaProof and AlphaGeometry teams. AI achieves silver-medal standard solving international mathematical olympiad problems. Google DeepMind Blog, 2024. URL https://deepmind.google/blog/ai-solves-imo-problems-at-silver-medal-level/.

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model for mathematics. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=4WnqRR915j.

Ananth Balashankar, Ziteng Sun, Jonathan Berant, Jacob Eisenstein, Michael Collins, Adrian Hutter, Jong Lee, Chirag Nagpal, Flavien Prost, Aradhana Sinha, Ananda Theertha Suresh, and Ahmad Beirami. InfAlign: Inference-aware language model alignment. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 2646–2672. PMLR, 2025. URL https://proceedings.mlr.press/v267/balashankar25a.html.

Yoshua Bengio and Nikolay Malkin. Machine learning and information theory concepts towards an AI mathematician, 2024. URL https://arxiv.org/abs/2403.04571.

Feng Chen, Allan Raventós, Nan Cheng, Surya Ganguli, and Shaul Druckmann. Rethinking fine-tuning when scaling test-time compute: Limiting confidence improves mathematical reasoning. In *Advances in Neural Information Processing Systems*, volume 38, 2025. URL https://proceedings.neurips.cc/paper_files/paper/2025/hash/e8f4eae0a41cab67fdead3aa6b77f083-Abstract-Conference.html.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Yinlam Chow, Guy Tennenholtz, Izzeddin Gur, Vincent Zhuang, Bo Dai, Sridhar Thiagarajan, Craig Boutilier, Rishabh Agarwal, Aviral Kumar, and Aleksandra Faust. Inference-aware fine-tuning for best-of-N sampling in large language models. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=77gQUdQhE7.

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

Rémi Coulom. Efficient selectivity and backup operators in Monte-Carlo tree search. In *Computers and Games*, volume 4630 of *Lecture Notes in Computer Science*, pp. 72–83. Springer, 2007a. doi: 10.1007/978-3-540-75538-8_7. URL https://doi.org/10.1007/978-3-540-75538-8_7.

Rémi Coulom. Computing Elo ratings of move patterns in the game of Go. In *Computer Games Workshop*, 2007b. URL https://www.remi-coulom.fr/Amsterdam2007/.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://openreview.net/forum?id=mZn2Xyh9Ec.

Leonardo de Moura and Sebastian Ullrich. The Lean 4 theorem prover and programming language. In *Automated Deduction – CADE 28*, pp. 625–635. Springer International Publishing, 2021. doi: 10.1007/978-3-030-79876-5_37. URL https://doi.org/10.1007/978-3-030-79876-5_37.

Jesse Michael Han, Jason Rute, Yuhuai Wu, Edward W. Ayers, and Stanislas Polu. Proof artifact co-training for theorem proving with language models. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2102.06203.

Peter E Hart, Nils J Nilsson, and Bertram Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions on Systems Science and Cybernetics*, 4(2):100–107, 1968. URL https://ieeexplore.ieee.org/document/4082128/.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models, 2021. URL https://arxiv.org/abs/2106.09685.

Albert Q. Jiang, Sean Welleck, Jin Peng Zhou, Wenda Li, Jiacheng Liu, Mateja Jamnik, Timothée Lacroix, Yuhuai Wu, and Guillaume Lample. Draft, sketch, and prove: Guiding formal theorem provers with informal proofs. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=SMa9EAovKMC.

Levente Kocsis and Csaba Szepesvári. Bandit based monte-carlo planning. In *European Conference on Machine Learning*, pp. 282–293. Springer, 2006. doi: 10.1007/11871842_29. URL https://doi.org/10.1007/11871842_29.

Guillaume Lample, Timothée Lacroix, Marie-Anne Lachaux, Aurélien Rodriguez, Amaury Hayat, Thibaut Lavril, Gabriel Ebner, and Xavier Martinet. HyperTree proof search for neural theorem proving. In *Advances in Neural Information Processing Systems*, volume 35, pp. 26337–26349, 2022. URL https://papers.nips.cc/paper_files/paper/2022/hash/a8901c5e85fb8e1823bbf0f755053672-Abstract-Conference.html.

Laurent Orseau and Levi H. S. Lelis. Policy-guided heuristic search with guarantees, 2021. URL https://arxiv.org/abs/2103.11505.

Laurent Orseau, Levi H. S. Lelis, Tor Lattimore, and Théophane Weber. Single-agent policy tree search with guarantees. In *Advances in Neural Information Processing Systems*, volume 31, 2018. URL https://arxiv.org/abs/1811.10928.

Adam Ousherovitch and Ambuj Tewari. Compute aligned training: Optimizing for test time inference. In *Advances in Neural Information Processing Systems*, 2026. URL https://arxiv.org/abs/2604.24957. Accepted; to appear.

Judea Pearl. *Heuristics: Intelligent Search Strategies for Computer Problem Solving*. Addison-Wesley, 1984.

Gabriel Poesia, David Broman, Nick Haber, and Noah D. Goodman. Learning formal mathematics from intrinsic motivation. In *Advances in Neural Information Processing Systems*, volume 37, 2024. URL https://arxiv.org/abs/2407.00695.

Stanislas Polu and Ilya Sutskever. Generative language modeling for automated theorem proving, 2020. URL https://arxiv.org/abs/2009.03393.

Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. Formal mathematics statement curriculum learning. *arXiv preprint arXiv:2202.01344*, 2022. doi: 10.48550/arxiv.2202.01344. URL https://arxiv.org/abs/2202.01344.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024. URL https://arxiv.org/abs/2408.03314.

Peiyang Song, Kaiyu Yang, and Anima Anandkumar. Lean Copilot: Large language models as copilots for theorem proving in Lean. In *Proceedings of the International Conference on Neuro-symbolic Systems*, volume 288 of *Proceedings of Machine Learning Research*, pp. 144–169. PMLR, 2025. URL https://proceedings.mlr.press/v288/song25a.html.

Yunhao Tang, Kunhao Zheng, Gabriel Synnaeve, and Rémi Munos. Optimizing language models for inference time objectives using reinforcement learning. In *Proceedings of the 42nd International Conference on Machine Learning*, 2025. URL https://arxiv.org/abs/2503.19595.

The mathlib Community. The Lean mathematical library. In *Proceedings of the 9th ACM SIGPLAN International Conference on Certified Programs and Proofs (CPP)*, pp. 367–381. ACM, 2020. doi: 10.1145/3372885.3373824. URL https://doi.org/10.1145/3372885.3373824.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023. URL https://arxiv.org/abs/2203.11171.

Yuhuai Wu, Albert Q Jiang, Wenda Li, Markus N Rabe, Charles Staats, Mateja Jamnik, and Christian Szegedy. Autoformalization with large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 32353–32368, 2022. URL https://arxiv.org/abs/2205.12615.

Ran Xin, Chenguang Xi, Jie Yang, Feng Chen, Hang Wu, Xia Xiao, Yifan Sun, Shen Zheng, and Kai Shen. BFS-Prover: Scalable best-first tree search for LLM-based automatic theorem proving, 2025. URL https://arxiv.org/abs/2502.03438.

Kaiyu Yang, Aidan Swope, Alex Gu, Rahul Chalamala, Peiyang Song, Shixing Yu, Saad Godil, Ryan J. Prenger, and Animashree Anandkumar. LeanDojo: Theorem proving with retrieval-augmented language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 36, pp. 21573–21612, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/4441469427094f8873d0fecb0c4e1cee-Abstract-Datasets_and_Benchmarks.html.

Kaiyu Yang, Gabriel Poesia, Jingxuan He, Wenda Li, Kristin Lauter, Swarat Chaudhuri, and Dawn Song. Formal mathematical reasoning: A new frontier in AI, 2024. URL https://arxiv.org/abs/2412.16075.

Kunhao Zheng, Jesse Michael Han, and Stanislas Polu. miniF2F: A cross-system benchmark for formal olympiad-level mathematics. In *International Conference on Learning Representations*, 2022. URL https://arxiv.org/abs/2109.00110.

A STRATEGY IMAGES

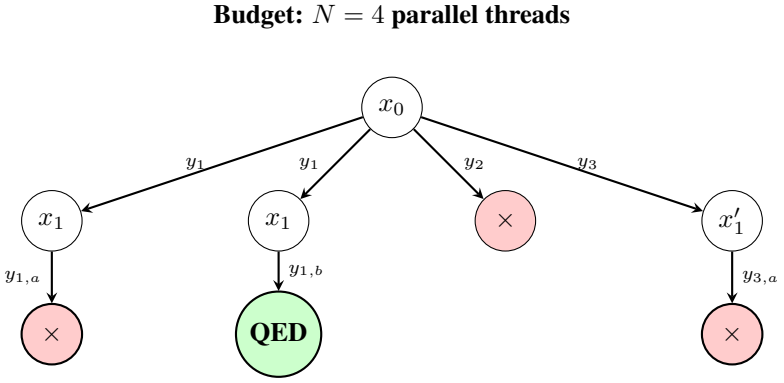

Figure 3: Pass@N Parallel Search. $N = 4$ independent paths explore the tree simultaneously. A single valid path reaching `QED` constitutes a global success.

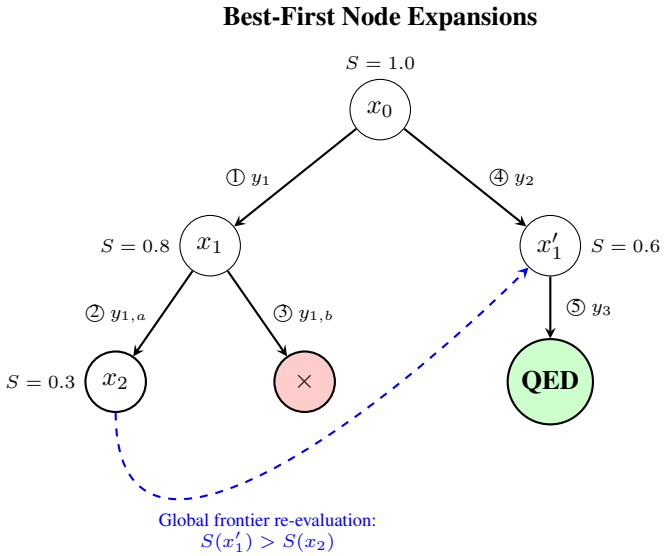

Figure 4: Policy-Guided Best-First Search. The numbered circles indicate chronological node expansions. Expanding x_0 yields two valid states on the frontier, x_1 ($S = 0.8$) and x_1' ($S = 0.6$). The algorithm greedily expands x_1 (step 2). However, generating the tactic to reach x_2 drops the path's geometric mean to $S = 0.3$. Because the algorithm evaluates the global frontier, the decision rule abandons the active path and dynamically jumps across the tree to expand x_1' (step 4), ultimately finding the proof.

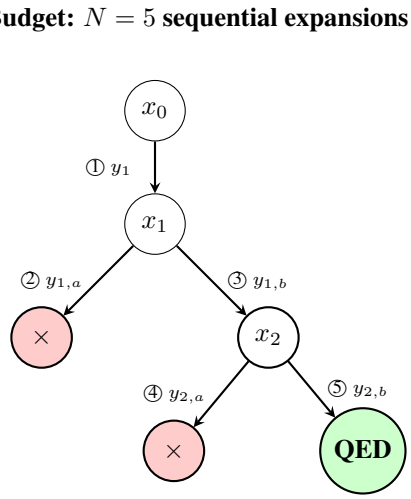

Figure 5: Depth-First Search. The numbered circles indicate the chronological order of node expansions. The algorithm pushes forward, backtracks locally upon failure (e.g., from step 2 back to x_1), and terminates the search immediately upon reaching the `QED` state at step 5.

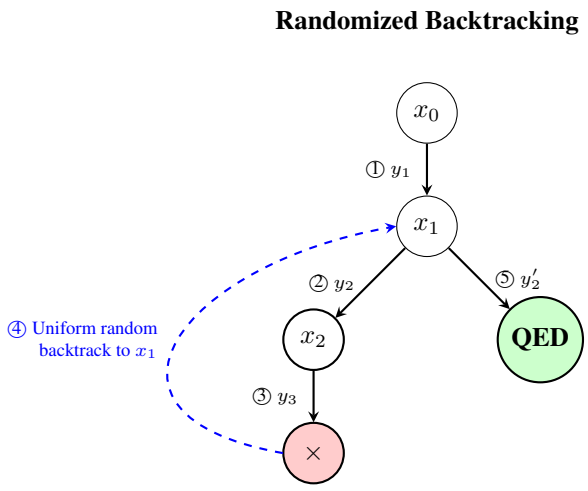

Figure 6: Randomized Depth-First Search. After expanding a deep, dead-end path (steps 1–3), the algorithm selects a node uniformly at random from the active path $\{x_0, x_1, x_2\}$ to backtrack to. In this case, it randomly selects x_1 (step 4), escaping the trap at x_2 and successfully finding the proof on an alternative branch (step 5).

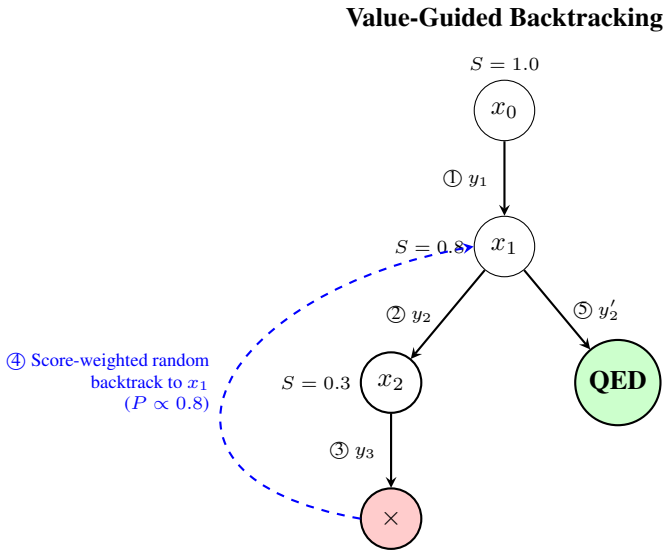

Figure 7: Value Guided Depth-First Search (VDFS). After expanding into a dead end (step 3), the algorithm evaluates the active path prefix $\{x_0, x_1, x_2\}$. Instead of backtracking uniformly, it samples an ancestor proportional to its Levin score S. In this case, it randomly jumps to x_1 (step 4) because its higher score ($S = 0.8$) makes it a heavily favored target over the deep trap at x_2 ($S = 0.3$). This allows it to successfully escape and find the proof on an alternative branch (step 5).

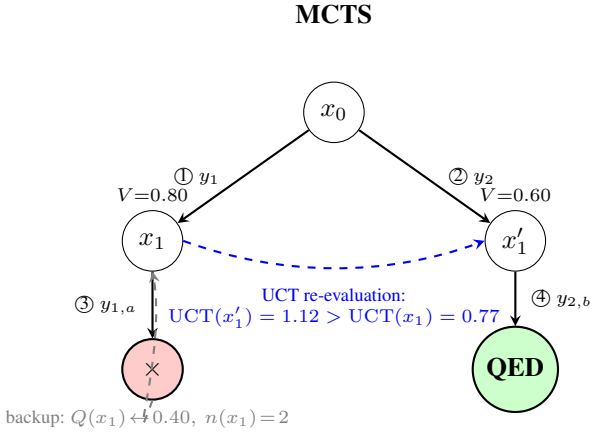

Figure 8: Deployed Monte Carlo Tree Search. Circled numbers indicate the chronological order of node expansions; a budget of $N = 4$ expansions is exhausted in this run. *Selection/Expansion:* the root is expanded twice, sampling y_1 into x_1 (step 1) and y_2 into x_1' (step 2); each is evaluated once via the policy's own Levin score, giving $V(x_1) = 0.80$ and $V(x_1') = 0.60$. With both children visited once, UCT (exploration constant $c = 0.5$) scores x_1 at 1.22 against x_1' at 1.02, so the search commits to the higher-value child and expands it (step 3), reaching a dead end. *Backup:* the failure backs up a value of 0 through x_1, dropping its running average to $Q(x_1) = 0.40$. *Selection reverses:* recomputing UCT with the updated statistics gives x_1' a score of 1.12 against x_1's now-diminished 0.77 – unlike Policy-Guided Best-First Search (Figure 4), where the frontier is re-ranked by a static score, here it is the shrinking exploration bonus *and* the value drop from a real backup that overturn the earlier choice. The search abandons x_1's subtree, expands x_1' (step 4), and reaches `QED`.

B Formal MDP Formulation of Automated Theorem Proving

We formalize the structure of LLM-guided theorem proving in interactive environments like Lean, where text generation is strictly mediated by a formal verifier (the Lean kernel), as a Markov Decision Process (MDP) (Polu & Sutskever, 2020; Lample et al., 2022; Song et al., 2025). We define this sequential decision-making process as a tuple $\mathcal{M} = \langle \mathcal{X}, \mathcal{Y}, \mathcal{T}, \mathcal{R}, x_0 \rangle$:

- **State Space ($\mathcal{X}$):** The discrete set of all valid Lean proof states. This space includes two terminal absorbing states: a universal success state x_{QED} and a failure state x_{err}.
- **Action Space ($\mathcal{Y}$):** The space of possible tactics the LLM can generate.
- **Transition Function ($\mathcal{T}$):** The deterministic update rules governed by the Lean kernel, defined as $\mathcal{T} : \mathcal{X} \times \mathcal{Y} \to \mathcal{X}$. Given a state x_{t-1} and tactic y_t, $\mathcal{T}(x_{t-1}, y_t) = x_t$ if the tactic is logically and syntactically valid. If y_t is invalid, the kernel immediately rejects it, and $\mathcal{T}(x_{t-1}, y_t) = x_{err}$. If y_t completes the proof, $\mathcal{T}(x_{t-1}, y_t) = x_{QED}$.
- **Reward Function ($\mathcal{R}$):** A sparse terminal reward function mapping to $\{0, 1\}$. $\mathcal{R}(x_{t-1}, y_t) = 1$ if $\mathcal{T}(x_{t-1}, y_t) = x_{QED}$, and 0 otherwise.
- **Initial State (x_0):** The underlying theorem declaration to be proven.

The probability of executing any valid trajectory of length K, denoted $\tau = (x_0, y_1, x_1, \ldots, y_K, x_K)$, is the product of the model's autoregressive action probabilities:

$$P_{\pi_\theta}(\tau \mid x_0) = \prod_{t=1}^{K} \pi_\theta(y_t \mid x_{t-1}) \tag{17}$$

Let $\Gamma_{succ}(x_0)$ denote the intractable set of all valid, successful trajectories originating from x_0 and terminating in x_{QED}. The true probability of proof discovery without search (Pass@1) is the marginal likelihood over this entire set:

$$P_{\text{any}} := \sum_{\tau \in \Gamma_{succ}(x_0)} P_{\pi_\theta}(\tau \mid x_0) \tag{18}$$

Because marginalizing over $\Gamma_{succ}(x_0)$ is computationally impossible, standard SFT sidesteps this global objective, reducing the objective to maximizing the likelihood of a single optimal trace τ^* via behavioral cloning, which yields the standard cross-entropy loss discussed in Section 2.

C Approximation Error and Scaling

This appendix develops the approximation analysis for the *search-aware* objectives introduced in Section 3. These objectives replace the deployed success probability $J^{\mathcal{A}}$ with a trace-supported proxy $\widehat{J}_{\mathcal{A}}$. We study the resulting approximation through the per-tactic gradient weights, asking two questions: (i) at a fixed budget, when is the search-aware weight closer to the deployed weight than CE's constant unit weight? (ii) how does this approximation error change as the test-time budget grows? The search-agnostic UA objective is a separate budget-based construction, discussed in Section J, and need not approximate the dynamics of a particular deployed search.

We study Pass@N (Section C.2) and the shared-budget retry model used for best-first search (Section C.3). For Pass@N, the budget counts independent rollouts, allowing us to isolate the effect of alternative successful trajectories omitted by the demonstrated trace. For the retry model, we examine how unsuccessful off-trace excursions consume a shared expansion budget. The latter results rely on explicit assumptions about excursion costs and recovery, rather than providing a complete characterization of deployed best-first search.

The analysis separates three effects:

- **Bypass channel.** For Pass@N, under the proportional-reallocation convention and Assumption 1, the deployed weight is no larger than the trace-supported weight, which is no larger than CE's unit weight (Theorem 4).
- **Trap channel.** In a trap-only retry model with a common deterministic miss cost, excursions that return to the same trace state reduce the effective budget. The resulting weight lies between the retry-surrogate weight and CE's unit weight (Proposition 3).
- **Budget scaling.** In the models studied here, large-budget weight error vanishes for Pass@N and for recoverable shared-budget excursions. Permanent absorption can instead leave a nonzero error floor (Section C.4).

These results identify off-trace quantities that demonstrations alone do not determine. Incorporating them requires additional modeling assumptions or observations of search behavior. Section I explores a simple use of such assumptions through priors on off-trace cost and recovery.

C.1 Well-posed deployed weights

The true objective depends on the entire policy, not only on the demonstrated probabilities $p = (p_1, \ldots, p_L)$, so "$\partial J/\partial p_t$" is meaningless until we fix how the remaining probability mass moves when p_t does. We adopt the assumption of *proportional reallocation*: at each demonstrated state, write

$$\pi_\theta(y \mid x_{t-1}) = p_t\, \mathbf{1}\{y = y_t^*\} + (1 - p_t)\, \rho_t(y), \qquad \rho_t \in \Delta\big(\mathcal{Y} \setminus \{y_t^*\}\big), \tag{19}$$

and define derivatives in p_t holding ρ_t, all other p_s, and all conditionals at non-demonstrated states fixed. This is not explicitly true, tactics similar to the demonstrated tactic will gain probability mass while distant ones will lose mass; however, this convention makes analysis feasible. This is strictly a convention and not a statement about how changing the gradient redistributes mass to unobserved tactics. All statements about the true weights below are relative to this convention. For any success function $J > 0$ define the *elasticity*

$$w_t(J) := \frac{\partial \log J}{\partial \log p_t} = \frac{p_t}{J}\frac{\partial J}{\partial p_t}, \tag{20}$$

so that $w_t^\star := w_t(J^{\mathcal{A}})$ is the (unknown) deployed weight, $\widehat{w}_t := w_t(\widehat{J}_{\mathcal{A}})$ is the CAT weight of Proposition 1, and CE corresponds to $w_t^{\mathrm{CE}} \equiv 1$. Comparing training rules then reduces to a scalar question at each step.

Proposition 2 (When is CAT closer than CE?)**.** *Suppose* $0 \le \widehat{w}_t < 1$. *Then* $\big|w_t^\star - \widehat{w}_t\big| \le \big|w_t^\star - 1\big|$ *if and only if* $w_t^\star \le \frac{1}{2}(1 + \widehat{w}_t)$. *If* $\widehat{w}_t = 1$, *CAT and CE coincide.*

Proof. Square both sides, cancel $(w_t^\star)^2$, and divide by $1 - \widehat{w}_t > 0$. □

Thus, CAT wins exactly when the deployed search makes the demonstrated tactic less critical than single-shot generation assumes; CE can win only when the tactic remains a near single-shot bottleneck under deployment. The next two subsections determine which side of the crossover each strategy occupies.

C.2 PASS@N: UNIT ELASTICITY AND A ONE-SIDED SQUEEZE

Let $a(p) = \prod_{i=1}^{L} p_i$ be the one-rollout probability of the demonstrated trace, $b(p)$ the one-rollout probability of proving the theorem by any *other* successful trajectory, and $P_{\text{any}} = a + b$.[9] With $F(x) = 1 - (1-x)^N$, the deployed objective is $J^{\text{true}} = F(P_{\text{any}})$ and the surrogate is $\widehat{J} = F(a)$. Define the elasticity of the outer function,

$$H_F(x) = \frac{xF'(x)}{F(x)}. \tag{21}$$

Lemma 1 (Properties of H_F)**.** *For $F(x) = 1 - (1-x)^N$ and $x \in (0,1)$,*

$$H_F(x) = \frac{N}{\sum_{j=0}^{N-1}(1-x)^{-j}}, \tag{22}$$

so H_F is nonincreasing, $H_F(x) \le 1$ with $H_F(0^+) = 1$, and for fixed x, $H_F(x) \to 0$ as $N \to \infty$.

Proof. $F(x) = x\sum_{k=0}^{N-1}(1-x)^k$ and $F'(x) = N(1-x)^{N-1}$; divide and shift the index by $N-1$. Each summand $(1-x)^{-j} \ge 1$ and is nondecreasing in x, giving the bound and monotonicity; the sum diverges geometrically in N for fixed x, giving the limit. □

Assumption 1 (At most one visit)**.** *Almost surely, every successful one-rollout trajectory visits each demonstrated proof state x_{t-1} at most once. This holds automatically when states are history-augmented (the search tree), which is the case in our setting.*

Lemma 2 (Unit elasticity)**.** *Under Equation* (19) *and Assumption 1, partition the non-demonstrated success mass at step t as $b = b_t^{\text{sh}} + b_t^{\text{sib}} + b_t^{\varnothing}$: trajectories that traverse the demonstrated transition (x_{t-1}, y_t^*) but are not τ^*; trajectories that visit x_{t-1} and deviate there; and trajectories that never visit x_{t-1}. Then*

$$p_t \frac{\partial P_{\text{any}}}{\partial p_t} = \left(a + b_t^{\text{sh}}\right) - \frac{p_t}{1-p_t}\, b_t^{\text{sib}} \;\le\; P_{\text{any}}, \qquad \textit{i.e.} \qquad \eta_t := \frac{p_t\, \partial_{p_t} P_{\text{any}}}{P_{\text{any}}} \le 1. \tag{23}$$

No demonstrated tactic can move the total success probability excessively. (η_t may be negative when sibling mass dominates.)

Proof. By Assumption 1 each successful trajectory contains the conditional at x_{t-1} at most once. Under Equation (19) its probability is therefore of the form $p_t C$, $(1-p_t)\rho_t(y)C$, or C with C free of p_t, according to which subset of b it belongs. Differentiate termwise and multiply by p_t. The inequality follows since the first term is at most P_{any} and the second is nonpositive. □

Theorem 4 (Weight squeeze for Pass@N)**.** *Under Assumption 1, for every step t and budget N,*

$$w_t^\star \;=\; \eta_t\, H_F(P_{\text{any}}) \;\le\; H_F(P_{\text{any}}) \;\le\; H_F(a) \;=\; \widehat{w}_t \;\le\; 1 \;=\; w_t^{\text{CE}}. \tag{24}$$

Consequently $|\widehat{w}_t - w_t^\star| \le |1 - w_t^\star|$: CAT weakly dominates CE at every step and every budget, with strict domination whenever any inequality is strict.

Proof. $w_t^\star = H_F(P_{\text{any}})\,\eta_t$ and $\widehat{w}_t = H_F(a)\cdot 1$ by the chain rule and $p_t\partial_{p_t} a = a$. Apply Lemma 2, then $a \le P_{\text{any}}$ with H_F nonincreasing, then $H_F \le 1$ (Lemma 1). Domination: $w_t^\star \le \widehat{w}_t \le 1$ gives $\widehat{w}_t - w_t^\star \le 1 - w_t^\star$. □

[9] Note that π_{proof} in Section 3.1 equals a.

Error decomposition. The gap localizes into the two ways the surrogate is blind:

$$\widehat{w}_t - w_t^\star = \underbrace{\big[H_F(a) - H_F(P_{\text{any}})\big]}_{\text{ignored bypass mass}} + \underbrace{(1-\eta_t)\, H_F(P_{\text{any}})}_{\text{sharing / reallocation structure}} . \tag{25}$$

The first term is the objective-level blindness to b; the second is present even at $N = 1$ and reflects that raising p_t proportionally starves sibling alternatives at x_{t-1} while riding along with shared-prefix alternatives.

Remark. A useful strengthening is assuming $b \le \varepsilon a$, the mean value theorem bounds the first term of Equation (25) by $\varepsilon a \sup_{x\in[a,a+b]} |H_F'(x)|$, giving a quantitative guarantee. This assumption is in a sense that the expert trace takes up a nontrivial chunk of the probability mass of correct proofs.

C.3 Best-first search: censoring and excursions

For BFS the surrogate is the retry objective of Equation (14),

$$\widehat{J}(N) = \mathbb{P}(S_L \le N), \qquad S_L = \sum_{t=1}^{L} T_t, \qquad T_t \sim \text{Geom}(p_t) \text{ independent.} \tag{26}$$

Unlike Pass@N, a deployed miss does not cost exactly one budget unit. The search may wander off-trace before the trace node is re-selected, or it may find a proof out there without returning. We separate these two events, then approximate each.

C.3.1 The CAT weight is a criticality

Lemma 3 (CAT weight as criticality)**.** *Let T_t' be an independent copy of T_t and $p_i \in (0,1)$. For Equation* (26)*,*

$$\widehat{w}_t(N) = 1 - \frac{\mathbb{P}(S_L + T_t' \le N)}{\mathbb{P}(S_L \le N)} = \mathbb{P}(S_L \le N < S_L + T_t' \,|\, S_L \le N)\,. \tag{27}$$

Hence $0 \le \widehat{w}_t(N) \le 1$, $\widehat{w}_t(L) = 1$, $\widehat{w}_t(N) \to 0$ as $N \to \infty$, and $\widehat{w}_t(N)$ is non-increasing in N.

Proof. For $T \sim \text{Geom}(p)$ and any bounded ψ, $p\,\partial_p \mathbb{E}[\psi(T)] = \mathbb{E}[\psi(T)] - \mathbb{E}[\psi(T+T')]$. Apply with $\psi(k) = \mathbb{P}(R_t + k \le N)$, $R_t = \sum_{i\neq t} T_i$, and divide by $\widehat{J}(N)$. For monotonicity: the pmf of S_L is a convolution of log-concave pmfs, hence log-concave, and so is its CDF F; thus $F(n-k)/F(n)$ is nondecreasing in n for fixed $k \ge 1$, and $1 - \widehat{w}_t(n) = \mathbb{E}_{T_t'}[F(n-T_t')/F(n)]$ is nondecreasing. □

C.3.2 Separating alternative proofs: the bypass censored coupling

To analyze the impact of off-trace proofs on BFS, consider an artificial "bypass censored process". This process behaves exactly like standard, deployed BFS. Except, if it ever samples a tactic which would yield a proof other than the expert proof, we replace the observation of x_{QED} by x_{err}. So it only succeeds if it returns the demonstrated trace but it is allowed to be off path during the search.

Lemma 4 (Bypass Censoring)**.** *Let N be the initial budget, $J^{\text{trap}}(N)$ be the censored process' success probability and let $\beta(N)$ be the probability that the deployed run samples a transition which would have been censored if we had instead run the censored process. Then*

$$J^{\text{trap}}(N) \;\le\; J^{\text{true}}(N) \;\le\; J^{\text{trap}}(N) + \beta(N). \tag{28}$$

Proof. Couple the two runs on the same tactic draws and selector randomness. They are identical until the deployed run first samples a transition which would have been censored: at that moment, the deployed run has succeeded. If no censored transition is ever sampled, outcomes agree. So censored success implies deployed success, and deployed success implies censored success or divergence. □

This is just to say that alternate proof paths can only *add* success. To use this to analyze the weights, split the true success probability by which event occurs first, $J^{\text{true}} = J^\dagger + B$ ($J^\dagger$: completion of the demonstrated trace; B: completion of an alternate trace). From looking at the sum of elasticities,

$$\frac{\partial J^{\text{true}}}{\partial p_t} = \frac{\partial J^\dagger}{\partial p_t} + \frac{\partial B}{\partial p_t}$$

$$w_t^\star = \frac{p_t}{J^{\text{true}}}\frac{\partial J^\dagger}{\partial p_t} + \frac{p_t}{J^{\text{true}}}\frac{\partial B}{\partial p_t}$$

$$w_t^\star = \left(\frac{J^\dagger}{J^{\text{true}}}\right)\left(\frac{p_t}{J^\dagger}\frac{\partial J^\dagger}{\partial p_t}\right) + \left(\frac{B}{J^{\text{true}}}\right)\left(\frac{p_t}{B}\frac{\partial B}{\partial p_t}\right)$$

$$w_t^\star = \lambda\, w_t\big(J^\dagger\big) + (1-\lambda)\, w_t(B), \qquad \lambda = \frac{J^\dagger}{J^{\text{true}}}, \tag{29}$$

holds *exactly*.

This mixture identity reveals that the true weight is subject to a tug-of-war between two competing "channels" of success.

First, the **trap channel** ($J^\dagger$) captures trajectories that wander into dead ends but eventually return to complete the demonstrated proof. Because these off-trace excursions consume a shared compute budget, they increase the bottleneck on the remaining path. This increased criticality pushes the weight $w_t(J^\dagger)$ *up* relative to the trace-supported surrogate $\widehat{w}_t$ (formalized in Proposition 3).

Conversely, the **bypass channel** (B) captures trajectories that successfully discover alternative, unobserved proofs. The existence of these alternative solutions makes the exact demonstrated tactic less critical for global success, which pushes the weight *down*, mirroring the elasticity squeeze observed in the Pass@N analysis. Thus, we will focus the remaining analysis on the effect of the trap channel.

One structural caveat remains: the true BFS selector's geometric-mean scores dynamically depend on p, a dependence the trace-supported surrogate erases to maintain independence between steps. Because Levin scores are cumulative, increasing an early on-trace probability p_s creates a cascading advantage for all subsequent trace nodes. This allows the trace to win frontier comparisons sooner, actively suppressing the length of future off-trace excursions. This positive feedback loop means the true search penalizes on-trace mistakes even more heavily than our surrogate models. Consequently, the trap channel's upward pressure on the deployed weight $w_t^\star$ is strictly stronger in reality. We will, however, treat the mixture identity (Equation (29)) as exact.

C.3.3 The Trap Channel: Exploration Waste and Shared-Budget Bracketing

In the trap channel, when BFS samples an off-trace tactic at state x_{t-1}, the algorithm embarks on an off-trace excursion until it returns back to the trace. To quantify the computational toll of deviations, let $E_{t,j} \in \{0, 1, \ldots\} \cup \{\infty\}$ denote the number of wasted node expansions spent off-trace following the j-th miss at state x_{t-1} before the search backtracks and re-selects the trace node ($E_{t,j} = \infty$ if the algorithm is permanently absorbed). The trap-channel success probability is then precisely represented by:

$$J^{\text{trap}}(N) = \mathbb{P}\left(\sum_{t=1}^{L} T_t \;+\; \sum_{t=1}^{L}\sum_{j=1}^{T_t-1} E_{t,j} \;\le\; N\right), \tag{30}$$

where $T_t \sim \text{Geom}(p_t)$ tracks on-trace trials. Different choices of weight during training can be viewed as distinct assumptions on E:

- **Minimal waste ($E \equiv 0$):** Every off-trace tactic is instantly rejected by the environment (x_{err}). A miss costs exactly 1 expansion, recovering our CAT retry surrogate exactly ($J^{\text{trap}}(N) = \widehat{J}(N)$).
- **Permanent absorption ($E \equiv \infty$):** Any off-trace mistake traps the search indefinitely. Here, $J^{\text{trap}}(N) = \prod_{t=1}^{L} p_t$ for any budget $N \ge L$, yielding unit elasticity ($w_t^\star = 1$), recovering the exact regime where standard SFT weighting is optimal.

Between these two extremes lies the realistic setting where off-trace excursions consume finite, nonzero compute before backtracking. To isolate how this exploration waste alters the true weight, we first examine the case of a fixed return cost.

Assumption 2 (Deterministic excursion cost)**.** *In the trap-channel process, every miss incurs a total expenditure of $\kappa \ge 1$ expansions (the initial trial plus an excursion of length $E \equiv \kappa - 1$) before returning the search to x_{t-1}; successful trials cost 1 expansion; κ is independent of the policy probabilities p.*

Proposition 3 (Shared-budget bracketing)**.** *Under Assumption 2, let $N \geq L$ be an integer budget and $p_s \in (0,1)$ for all s. Completing the demonstrated path in the bypass-censored process within budget N is equivalent to evaluating the retry surrogate at a reduced effective budget:*

$$J^{\mathrm{trap}}(N) = \widehat{J}(N_{\mathrm{eff}}), \qquad N_{\mathrm{eff}} = L + \frac{N-L}{\kappa} \leq N. \tag{31}$$

For noninteger budget arguments B, interpret $\widehat{J}(B) = \widehat{J}(\lfloor B \rfloor)$ and $\widehat{w}_t(B) = \widehat{w}_t(\lfloor B \rfloor)$.

Define the trap-channel weight by $w_t^{\mathrm{trap}}(N) := w_t(J^{\mathrm{trap}}(N))$. Because κ is independent of p, the effective-budget identity and Lemma 3 give

$$\widehat{w}_t(N) \leq w_t^{\mathrm{trap}}(N) = \widehat{w}_t(N_{\mathrm{eff}}) \leq 1 = w_t^{\mathrm{CE}}. \tag{32}$$

Proof. Completing step t requires $G_t \sim \mathrm{Geom}(p_t)$ trials costing $\kappa(G_t - 1) + 1$ total expansions. Summing over the trace yields a total cost $\kappa S_L - (\kappa - 1)L \leq N \iff S_L \leq N_{\mathrm{eff}}$. The bracket directly follows since $N_{\mathrm{eff}} \leq N$ for $\kappa \geq 1$ and $\widehat{w}_t(\cdot)$ is non-increasing. □

In real policy-guided Best-First Search, excursions do not incur deterministic costs; rather, they exhibit stochastic, bounded trajectories ($E \leq \bar{\kappa} - 1$ a.s.). Crucially, we can actively engineer this bound during deployment to prevent permanent absorption.

Enforcing Bounded Excursions via Global Search Restarts. In practical ATP deployments, we actively prevent policies from being trapped in cyclical or deep, unguided proof attempts ($P(E = \infty) = 0$) by deploying a restart-based Best-First Search heuristic. Specifically, if the global BFS does not discover a complete proof within a step threshold of $M < N$ total node expansions, the active search tree is aborted and the algorithm resets entirely to the initial theorem state x_0 to begin a fresh run with the remaining compute budget. However, the cost to setting such an M too low is aborting a possibly prominent path so in practice this M is fairly high. This restart mechanism guarantees that no single exploratory trajectory can waste more than M expansions off-trace. Consequently, the exploration waste incurred before re-attempting the demonstrated trajectory is strictly bounded ($E \leq M$ a.s., enforcing a maximum excursion severity $\bar{\kappa} \leq M + 1$). While enforcing this test-time constraint eliminates the SFT ceiling ($w_t^\star < 1$), combining it with Proposition 3 reveals an irreducible tension between trace-only supervision and search optimization.

Proposition 4 (Irreducible Ambiguity under Bounded Excursions)**.** *Call a per-step weighting rule* trace-supported *if it depends solely on training-time observables along τ^* (specifically, the demonstrated probabilities $\{p_s\}_{s=1}^{L}$, the search strategy $\mathcal{A}$, and the initial budget N). Suppose test-time search engineering enforces an upper bound on excursion waste such that $\kappa \in [1, \bar{\kappa}]$ for a finite limit $\bar{\kappa} > 1$. Then:*

(i) The true deployed weight $w_t^\star$ is strictly confined to the compressed bracket:

$$w_t^\star \in \left[\widehat{w}_t(N),\ \widehat{w}_t\left(L + \frac{N-L}{\bar{\kappa}} \right) \right]. \tag{33}$$

(ii) Because the realized excursion waste κ depends on unobserved off-trace dynamics, no trace-supported objective can uniquely identify $w_t^\star$. Any trace-supported rule w_t incurs a worst-case approximation error bounded below by half the bracket width:

$$\sup_{\kappa \in [1,\bar{\kappa}]} \left| w_t - w_t^\star(\kappa; N) \right| \ \geq\ \frac{1}{2} \left(\widehat{w}_t\left(L + \frac{N-L}{\bar{\kappa}} \right) - \widehat{w}_t(N) \right) \ >\ 0. \tag{34}$$

Proof. By Proposition 3, an environment with effective excursion cost κ realizes the deployed weight $w_t^\star = \widehat{w}_t(L + \frac{N-L}{\kappa})$. Because environments with $\kappa = 1$ (instant syntax error) and $\kappa = \bar{\kappa}$ (maximum restart cap M) yield identical training-time observables along τ^*, any trace-supported rule must output the same scalar w_t across both environments. The minimax lower bound follows directly from applying the triangle inequality to the interval endpoints. □

C.4 BUDGET SCALING OF THE WEIGHT ERROR: HUMP OR FLOOR

Using the understanding we've built, we can understand how the error in our approximation scales. A key intuition we will need is that the surrogate's weight error is generally not monotone in N. When N is minimal ($N = 1$ in our setting), we are just performing a single rollout. We aren't really performing search, and both the surrogate and real weights understand this, so the error of the surrogate is 0. Conversely, as $N \to \infty$, our probability of success (with some caveats) becomes 1. Thus, both the surrogate and deployed weights tend to 0. The error only really occurs in the middle between these two extremes.

The minimal-budget endpoint is universal. At the smallest budget that can possibly succeed, any strategy must execute τ^* with no misses: both $J^{\mathcal{A}}$ (trap channel) and $\widehat{J}_{\mathcal{A}}$ equal $\prod_t p_t$, whose elasticity is 1 at every step. So $\widehat{w}_t = w_t^\star = 1 = w_t^{\mathrm{CE}}$ and all three rules coincide (alternate proofs shorter than L are the only exception, entering through B in Equation (29)).

Proposition 5 (Pass@N: vanishing error)**.** *Fix the policy with $a \in (0,1)$. Then $\widehat{w}_t(1) = 1$ and $w_t^\star(1) = \eta_t$; both $\widehat{w}_t(N)$ and $w_t^\star(N)$ tend to 0 as $N \to \infty$; the bypass term of Equation* (25) *is zero at $N = 1$ and vanishes as $N \to \infty$ (a hump), while the reallocation term decays monotonically from $1 - \eta_t$. In particular, the CAT error tends to 0 while the CE error $|1 - w_t^\star|$ tends to 1.*

Proof. At $N = 1$, $H_F \equiv 1$. As $N \to \infty$, $H_F(x) \to 0$ for fixed $x > 0$ (Lemma 1); apply to $x = a$ and $x = P_{\mathrm{any}}$ in Theorem 4 and Equation (25). □

For BFS, the outcome depends on the excursions. We isolate it to a specific case.

Assumption 3 (Absorbing-excursion model)**.** *In the trap-only censored process, each miss at step t, independently of everything else, launches a non-returning excursion ($E = \infty$) with probability $q_t \in [0,1]$, and otherwise an excursion of bounded length $E \le \bar{\kappa} - 1$.*

Proposition 6 (Hump or floor)**.** *Under Assumption 3:*

(a) ***(Recurrence ⇒ hump.)*** *If $q_t = 0$ for all t and excursions are deterministic ($E \equiv \kappa - 1$), the trap-channel error is $w_t^\star(N) - \widehat{w}_t(N) = \widehat{w}_t(N_{\mathrm{eff}}) - \widehat{w}_t(N) \ge 0$, which is 0 at $N = L$, tends to 0 as $N \to \infty$ (since $N_{\mathrm{eff}} \to \infty$), and is positive in between whenever $\kappa > 1$: the error is humped. For bounded random excursions with no absorption, the error is likewise zero at $N = L$ and vanishes as $N \to \infty$; its finite-budget behavior is governed by Equation* (30)*.*

(b) ***(Absorption ⇒ floor.)*** *If $q_t > 0$, then as $N \to \infty$,*

$$J^{\mathrm{trap}}(N) \longrightarrow \prod_{t=1}^{L} \frac{p_t}{p_t + (1-p_t)q_t}, \qquad w_t^\star(N) \longrightarrow \frac{q_t}{p_t + (1-p_t)q_t} > 0, \tag{35}$$

while $\widehat{w}_t(N) \to 0$: the CAT error converges to the strictly positive floor (35)*. As $q_t \to 1$, the floor tends to 1 and CE becomes asymptotically exact at step t, recovering the permanent-absorption limit; as $q_t \to 0$, the limiting error at that step vanishes.*

Proof. (a) is Proposition 3 plus Lemma 3. (b) With unlimited budget, finite excursions are free, so step t succeeds iff y_t^* is sampled before an absorbing excursion: per trial, advance w.p. p_t, absorb w.p. $(1-p_t)q_t$, retry otherwise, giving per-step success $f(p_t) = p_t/(p_t + (1-p_t)q_t)$, independent across steps. Direct computation gives $\partial \log f / \partial \log p = q/(p + (1-p)q)$. The interchange of limit and derivative is justified within the model by monotone convergence of $J^{\mathrm{trap}}(N) \uparrow \prod_t f(p_t)$ together with monotonicity of $w_t^\star(N)$ inherited from Lemma 3 applied at the extended-cost representation. □

Approximation is Incorrect Asymptotically, but Weights Converge. The pieces we don't account for in our surrogate (the alternate proof mass $\beta(N)$ and total excursion expansions) do grow with N. But their effect on the weights is self-limiting. Both objectives saturate near 1, and every elasticity is near 0, so no weighting rule can be far from the truth. This is a key advantage of test-time aligned training.

D Depth First Search Variants

Depth First Search (DFS) is not a standard algorithm used in policy guided search. We introduce it in this context because analyzing it and its variants serve as excellent examples about the mechanics of constructing search aligned loss functions.

D.1 Standard DFS

Standard DFS utilizes its computational budget to sequentially construct a single path. When the algorithm expands a node x_{t-1} and the tactic is rejected ($\mathcal{T}(x_{t-1}, y_t) = x_{err}$), the search does not abandon the entire trajectory. Instead, the decision rule dictates a retreat of exactly one step to the immediate parent state x_{t-1}, expending another unit of compute to sample an alternative tactic. This localized trial-and-error continues until the algorithm either successfully reaches x_{QED} or exhausts the global node expansion budget N.

Finally, we note that because standard DFS commits rigidly to a valid path, its chronologically local backtracking is a structural weakness. If the search ventures into a deep, valid, but useless branch, the localized backtracking cannot escape the broader strategic error. The LLM will simply exhaust its entire computational budget trapped in the depths of a spurious path.

Instantiation. Define $\widehat{c} = \langle x, b \rangle$, where x is the active trace state and b is the remaining node expansion budget. The trace-supported frontier is always a singleton, $\widehat{\mathcal{F}} = \{x\}$.

The positive update advances the path and consumes one expansion. For the negative update, an off-trace tactic immediately collapses the search back to the parent x_{t-1} at the cost of one expansion. Thus, the updates are:

$$\widehat{U}^{+}_{\mathrm{DFS}}(x_{t-1}, b) = (x_t, b-1), \qquad \widehat{U}^{-}_{\mathrm{DFS}}(x_{t-1}, b) = (x_{t-1}, b-1)$$

Because the frontier is always a single node, the selector is trivial: $S_{\mathrm{DFS}}(x \mid \widehat{c}) = 1$. Consequently, the trace-supported recurrence for DFS perfectly mirrors the recurrence for BFS (Equation (13)):

$$\widehat{J}_{\mathrm{DFS}}(b, t) = p_t \widehat{J}_{\mathrm{DFS}}(b-1, t+1) + (1-p_t)\widehat{J}_{\mathrm{DFS}}(b-1, t).$$

Thus, the closed form loss will collapse to that which we derived for BFS.

D.2 Randomized Depth-First Search (RDFS)

To mitigate the vulnerability of DFS to spurious paths, we introduce a randomized variant. RDFS disrupts this failure mode by altering the backtracking target. When the policy π_θ reaches a dead end, the algorithm does not simply return to x_{t-1}. Instead, it selects a state *uniformly at random* from the entire active path prefix $(x_0, x_1, \ldots, x_{t-1})$ and resumes expansion from there. This allows the search to stochastically jump out of deep traps, efficiently reallocating compute to higher-level branching points while preserving the depth-seeking advantages of the standard algorithm. A visualization is given in Figure 6.

Instantiation. Let $\widehat{c} = \langle t, b \rangle$, with the frontier tracking the full active path $\widehat{\mathcal{F}} = \{1, 2, \ldots, t\}$. To incorporate randomized backtracking, the randomness is injected directly into the selector. When a dead end is reached, rather than the update rule deterministically collapsing the path, the negative update triggers the selector to sample a state uniformly at random from the entire available frontier $\widehat{\mathcal{F}}$.

Trace-supported recurrence. Because of this randomized selector, the compute consumed at depth t cannot be modeled as a simple sequence of independent geometric random variables. Instead, the trace-supported objective forms a dynamic program.

Let $\widehat{J}_{\mathrm{RDFS}}(b, t)$ denote the exact probability of discovering the complete proof given that the search is currently evaluating the trace state at index t with a remaining budget b. The boundary conditions

are:

$$\widehat{J}_{\mathrm{RDFS}}(b, L+1) = 1 \quad \text{for all } b \geq 0 \quad \text{(Proof successfully completed)} \tag{36}$$

$$\widehat{J}_{\mathrm{RDFS}}(0, t) = 0 \quad \text{for all } t \leq L \quad \text{(Compute budget exhausted)} \tag{37}$$

For $b > 0$ and $t \leq L$, the recurrence relation resolving Equation (5) branches into two outcomes. With probability p_t, the model generates the correct tactic and advances. With probability $(1 - p_t)$, the model fails, and the selector backtracks uniformly to any valid ancestor index $k \in \{1, \ldots, t\}$. Thus, the recurrence becomes:

$$\widehat{J}_{\mathrm{RDFS}}(b, t) = p_t \widehat{J}_{\mathrm{RDFS}}(b-1, t+1) + (1 - p_t) \frac{1}{t} \sum_{k=1}^{t} \widehat{J}_{\mathrm{RDFS}}(b-1, k) \tag{38}$$

Unlike Pass@N or BFS, this objective does not reduce to a standard closed-form probability distribution.

D.3 Value Guided Depth-First Search (VDFS)

To mitigate the vulnerability of DFS while leveraging the policy's own internal confidence, we introduce Value Guided DFS (VDFS). Unlike RDFS, which backtracks uniformly, VDFS makes an informed retreat. When the search reaches a dead end, it backtracks to an ancestor state in the active path prefix with probability proportional to that state's Levin score (from Equation (12)). By using the geometric mean of the policy's probabilities as a confidence metric, the search preferentially returns to the most promising points on the current path. This concentrates the remaining computational budget on states the policy deems most reliable, while maintaining enough stochasticity to escape deep, spurious branches. A visualizaiton can be seen in Figure 7.

Instantiation. Let $\widehat{c} = \langle t, b \rangle$, where t is the active trace index (corresponding to the state x_{t-1}), and b is the remaining node expansion budget. The trace-supported frontier tracks the full active path prefix of states: $\widehat{\mathcal{F}} = \{x_0, x_1, \ldots, x_{t-1}\}$.

To compute the backtrack distribution, we evaluate the Levin score for each state on the frontier. Let s_k denote the Levin score of state x_k. For the root, $s_0 = 1$. For $k \geq 1$, the score is the geometric mean of the on-trace probabilities leading to that state:

$$s_k = \left(\prod_{j=1}^{k} p_j \right)^{\frac{1}{k}} \tag{39}$$

The positive update advances the proof by one step: $\widehat{U}^{+}_{\mathrm{VDFS}}(t, b) = (t+1, b-1)$. For the negative update, an off-trace tactic consumes a unit of budget and triggers a backtrack. The selector samples a backtrack target state $x_k \in \widehat{\mathcal{F}}$ with probability proportional to its score s_k. Transitioning to x_k sets the new active trace index to $k+1$. Thus, the negative update maps stochastically:

$$\widehat{U}^{-}_{\mathrm{VDFS}}(t, b) = (k+1, b-1), \quad \text{where } k \sim P(k) = \frac{s_k}{\sum_{i=0}^{t-1} s_i} \tag{40}$$

Trace-supported recurrence. Let $\widehat{J}_{\mathrm{VDFS}}(b, t)$ denote the exact probability of discovering the complete proof from the trace state at index t with remaining budget b. The boundary conditions are identical to RDFS:

$$\widehat{J}_{\mathrm{VDFS}}(b, L+1) = 1 \quad \text{for all } b \geq 0 \quad \text{(Proof successfully completed)} \tag{41}$$

$$\widehat{J}_{\mathrm{VDFS}}(0, t) = 0 \quad \text{for all } t \leq L \quad \text{(Compute budget exhausted)} \tag{42}$$

For $b > 0$ and $t \leq L$, the recurrence resolving Equation (5) weights the backtrack sum by the Levin scores. With probability p_t, the model generates the correct tactic and advances. With probability

$(1 - p_t)$, the model fails, and the selector backtracks to state x_k ($k \in \{0, \ldots, t-1\}$) according to the score distribution. The recurrence becomes:

$$\widehat{J}_{\mathrm{VDFS}}(b, t) = p_t \widehat{J}_{\mathrm{VDFS}}(b-1, t+1) + (1 - p_t) \sum_{k=0}^{t-1} \frac{s_k}{\sum_{i=0}^{t-1} s_i} \widehat{J}_{\mathrm{VDFS}}(b-1, k+1) \tag{43}$$

Like RDFS, this objective doesn't have a closed form. Notably, the gradients flow not only through the direct local transitions but also through the Levin scores s_k, which are themselves functions of the policy probabilities. This couples the training objective, forcing the model to calibrate its on-path probabilities to optimize the resulting backtracking distribution.

E MONTE CARLO TREE SEARCH (MCTS)

The strategies considered so far commit their budget by a rule fixed in advance: DFS always retries the deepest node, RDFS backtracks uniformly, and VDFS backtracks in proportion to a static per-node score. Monte Carlo Tree Search (Kocsis & Szepesvári, 2006; Coulom, 2007a) instead allocates compute according to statistics it accumulates as the search runs. We include it because it is the only strategy in our taxonomy whose selector depends on the *history* of the search rather than on the current configuration alone, and it therefore tests our construction at its least convenient.

MCTS is a 4-phase process at deployment. At each node, *Selection* either commits to its most promising child or stops to gain more information about another tactic at the current node. The choice to widen is governed by a progressive-widening schedule: a node that has been visited n times is permitted $\lceil \sqrt{n+1} \rceil$ distinct tactic samples (Coulom, 2007b). *Expansion* samples one tactic from π_θ at the selected node and submits it to the kernel, consuming one node expansion. *Evaluation* assigns the resulting state a value; in the variant, we choose to use the policy's own Levin score Equation (12) rather than a learned critic, so that MCTS requires no component the other strategies don't utilize. *Backup* propagates that value to every ancestor on the selected path, updating the statistics that drive future selections. A visualization is given in Figure 8.

Instantiation. With abuse of notation, let a configuration be

$$\widehat{c} \;=\; \big\langle\, (n_k, m_k, W_k)_{k=0}^{L-1},\; b \,\big\rangle,$$

where n_k is the visit count of the state x_k, m_k the number of tactics already sampled there, W_k the accumulated backed-up value, and b the remaining expansion budget. The trace-supported frontier is the prefix $\{x_0, \ldots, x_{L-1}\}$. The selector is not stochastic: the descent stops at the shallowest node that is either allowed a new tactic by the widening schedule or has no child yet,

$$\begin{aligned} d(\widehat{c}) \;&=\; \min\Big\{\, k \;:\; m_k < \big\lceil \sqrt{n_k + 1} \big\rceil \;\text{ or }\; x_{k+1} \text{ not yet reached} \,\Big\}, \\ &\widehat{S}_{\mathrm{MCTS}}(x_k \mid \widehat{c}) = \mathbf{1}\{k = d(\widehat{c})\}. \end{aligned} \tag{44}$$

The updates apply the expansion and the backup together. Writing $d = d(\widehat{c})$, a demonstrated tactic advances the trace and backs up the value $V(x_{d+1})$ of the state it reaches, while a miss backs up zero:

$$\widehat{U}^{+}_{\mathrm{MCTS}}(\widehat{c}): \quad m_d \mathrel{+}= 1, \quad n_k \mathrel{+}= 1,\; W_k \mathrel{+}= V(x_{d+1})\;\; \forall k \le d, \quad b \mathrel{-}= 1, \tag{45}$$

$$\widehat{U}^{-}_{\mathrm{MCTS}}(\widehat{c}): \quad m_d \mathrel{+}= 1, \quad n_k \mathrel{+}= 1,\; W_k \mathrel{+}= 0\;\; \forall k \le d, \quad b \mathrel{-}= 1, \tag{46}$$

with solved configurations (for the trace supported version) those in which depth L has been reached and dead configurations those with $b = 0$. Both updates decrement the budget, so the recurrence terminates.

For BFS, the naive selector is degenerate on a demonstrated trace because a chain never offers two frontier candidates at once. The same erasure happens here, in a slightly different place. The UCT rule ranks the children of a node by an exploration-adjusted value estimate, but under the trace transition Equation (4) every node has at most one child, so there is never a comparison for it to make. What survives is only the widening schedule in Equation (44): the rule that decides how many tactic samples a node receives before the search commits to descending. Thus in practice, this strategy trains like a local Pass@N at each node.

F Strategy Matching Within the Search-Aware Family

The main comparison evaluates search-aware CAT and search-agnostic UA across deployed strategies. Here, we isolate a different question: within the search-aware family, does matching the training objective to the deployed strategy improve proof success? Using the fixed-budget setup of Section 5.1, we evaluate the Pass@N and BFS search-aware adapters under both deployment strategies, with CE as a reference.

Table 3: **Strategy matching within the search-aware family.** Proof-success rates (%) at $N = 256$ tactic expansions. Columns are deployed strategies; rows are training objectives. Bold marks the higher point estimate among the two search-aware objectives in each column.

Training objective	Pass@N	BFS
CE	21.2	23.8
Search-aware CAT (Pass@N)	**26.6**	24.5
Search-aware CAT (BFS)	23.7	**25.1**

The matched objective has the higher observed success rate under each deployment strategy. This supports strategy matching within the tested search-aware pair, complementing the comparison with shared UA in Table 1.

G Experimental Details

G.1 Common setup

Environment. All experiments run against Lean 4 through LeanDojo $4.20.0$ (Yang et al., 2023). The policy is prompted with the pretty-printed proof state and emits a single tactic, which is submitted to the Lean kernel; the kernel verdict determines the transition. One *node expansion* denotes one policy sample together with its kernel verification, and is the unit in which all budgets are expressed. Kernel interaction is serial within a theorem, so wall-clock cost per unsolved theorem grows linearly in the budget.

Data. Training and evaluation both draw on `cat-searcher/leandojo-benchmark-4-random`, a random split of mathlib4 (The mathlib Community, 2020). The alignment stage consumes $40{,}660$ traced proofs comprising $76{,}003$ (state, tactic) training examples. The evaluation pool is the 458 held-out theorems whose reference proofs contain between 2 and 5 tactic steps; there is no overlap between training and evaluation theorems. Reference proofs are used only to compute training weights and to define the pool. At evaluation, a theorem counts as solved if the search returns any kernel-verified proof, whether or not it matches the reference.

Model and adapters. The policy is Qwen2.5-Math-7B-Instruct (Qwen Team, 2024) in bfloat16. All fine-tuning uses LoRA (Hu et al., 2021) with rank $r = 16$, $\alpha = 32$, dropout 0.05, applied to the query, key, value, output, gate, up, and down projections. This trains $40{,}370{,}176$ of $7{,}655{,}986{,}688$ parameters (0.53%). Gradient checkpointing is enabled throughout. All runs use a single NVIDIA A100-40GB. Training runs used FlashAttention-2 (Dao, 2024).

G.2 Training procedure and hyperparameters

Stage 1: Warm-up. A single supervised checkpoint is the common ancestor of every model we report. The first set of experiments branch off of two warm up epochs of SFT; while the second set branch off of only one. This discrepancy is a result of human error; however, given that runs are not compared cross-experiment, we find it reasonable to keep.

Stage 2: Alignment. From the warm-up checkpoint the runs branch. The control receives one further epoch of standard cross-entropy SFT. Each aligned model receives one further epoch over the same examples in the same order under the same optimizer settings, with the only difference being the scalar weight $\widehat{w}_t^{\mathcal{A}}$ applied to each tactic's cross-entropy gradient. An alignment epoch is $19{,}001$ optimizer steps and takes approximately 2.4 h.

Table 4: Optimization hyperparameters, identical for the control and every aligned run.

Optimizer	AdamW
Learning rate	1×10^{-4}
Schedule	linear decay, warm-up over 5% of steps
Micro-batch size	4
Gradient accumulation	16
Effective batch size	64
Gradient clipping	1.0 (global ℓ_2 norm)
Maximum sequence length	512 tokens
Precision	bfloat16 with autocast
Alignment epochs	1 (on top of the shared warm-up)
Random seed	42

Non-finite losses are skipped without an optimizer step; this occurs rarely and is logged. Examples exceeding the maximum sequence length are dropped at dataset construction rather than truncated, so no partially-supervised tactic enters training. The closed-form weights are evaluated directly in log space for numerical stability. The weights defined by a dynamic program are obtained by autodiff.

G.3 Inference and evaluation

Serving. Generation uses vLLM $0.6.3$ with the LoRA adapter supplied per request, GPU memory fraction 0.85, and maximum model length 4096 tokens. Sampling temperature is 0.8. The evaluation seed (42) fixes both the shuffle of the theorem pool and the sampling stream, so every configuration sees the identical theorems in the identical order; this is what licenses the paired analysis below and what makes a truncated cell a well-defined prefix of the full pool.

Table 5: Search parameters. Budgets are node expansions.

Parameter	Experiment 1	Experiment 2
Budget N	256	16, 64, 256, 1024
Best-first beam width	2	2
Expansions per pop	4	4
Best-first score exponent α	0.5	0.5
Pass@N generation batch	16	64
Maximum proof depth	5	5
Per-strategy wall-clock cap	90 s	none
Per-theorem wall-clock cap	500 s	none

Scoring and truncation. A theorem is scored as solved only if a kernel-verified proof is returned within the budget; unsolved-within-budget counts as a failure, with no exclusion. The two experiments differ in one respect that we record explicitly. Experiment 1 imposes the wall-clock caps in Table 5, so a theorem may in principle be cut off before its expansion budget is exhausted; Experiment 2 disables them, guaranteeing that every theorem spends its full budget. The caps are the reason the two experiments' absolute rates are not directly comparable, and they were removed for Experiment 2 because the effect grows with the budget: a wall-clock cap binds far more often at $N = 1024$ than at $N = 64$ and would otherwise have depressed exactly the high-budget cells whose behavior the experiment is about.

Statistics. Because all configurations are evaluated on the same theorems in the same order, every comparison is paired. For two configurations we report b, the number of theorems solved by the second and not the first, and c, the number solved by the first and not the second, on the subset n that both evaluated. The reported difference is $(b - c)/n$; its 95% interval uses the McNemar variance $\mathrm{Var}[(b-c)/n] = (b+c)/n^2$; and the reported p-value is the exact two-sided McNemar (sign) test on the $b + c$ discordant pairs. We use the paired difference rather than the difference of marginal rates throughout, because the two diverge whenever the cells being compared cover different numbers of theorems, and only the paired quantity is consistent with the interval and the p-value beside it. Where a figure shows unpaired intervals on individual rates (Figure 1) this is stated in the caption; those intervals are wider than the paired ones and overlapping bars do not indicate a non-significant difference.

G.4 Experiment 1: fixed-budget comparison

Configurations. Six deployed strategies at $N = 256$: Pass@N and best-first search from Section 3, and DFS, RDFS, VDFS and MCTS from Section D. Each is paired with the CAT weight derived for it (same for BFS/DFS), and we additionally deploy the single shared UA adapter under every strategy.

Control. The control for this experiment is the epoch-matched model: warm-up plus one additional epoch of standard SFT. Control and aligned models therefore receive identical total optimization, and the comparison isolates the loss weighting.

G.5 Experiment 2: budget sweep

Budget-matched adapters. We train a separate adapter per evaluation budget with the weight's budget parameter set to the deployment value, and evaluate each at its own budget only.

Control. The control for this experiment is the shared warm-up checkpoint, with an additional SFT epoch.

Pools. The $N \in \{16, 64, 256, 1024\}$ are evaluated on the full 458-theorem pool. However, we only allowed 80 GPU hours so not all evaluations were fully completed.

Reproduction. Given the trained adapters, each cell is one invocation of the evaluation harness with the strategy, budget, and adapter path. Cells write one JSON log per strategy keyed by theorem name, and all tables and figures in this paper are generated directly from those logs.

H HIGH-BUDGET EVALUATION

To complement Section 5.2, we also evaluated budget-matched CAT adapters at $N = 1024$ under Pass@N and BFS, using the same training and evaluation protocol. The evaluation-time limit restricted the paired comparisons to 57 theorems for Pass@N and 56 for BFS. Table 6 reports these results, and Figure 9 shows the complete budget sweep.

Table 6: **Compute-limited evaluation at $N = 1024$ node expansions.** n is the paired subset size, b counts CAT-only successes, and c counts control-only successes. Gains and 95% intervals are in percentage points. Intervals and exact two-sided McNemar tests follow Table 2.

Strategy	Budget N	n	b/c	Gain (pp), 95% interval	p
Pass@N	1024	57	6/1	+8.8 [−0.3, 17.9]	0.125
BFS	1024	56	5/2	+5.4 [−3.9, 14.6]	0.453

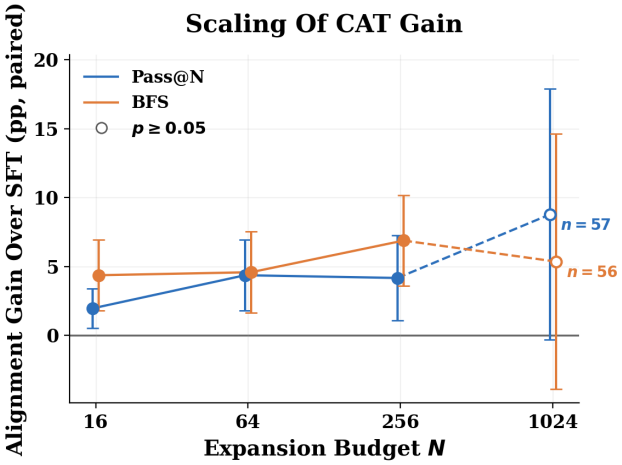

Figure 9: Complete budget sweep, including the compute-limited $N = 1024$ evaluation. Error bars show 95% intervals; filled markers denote $p < 0.05$ and hollow markers denote $p \geq 0.05$. Dashed segments connect to the $N = 1024$ estimates, annotated with their paired subset sizes. The completed subsets differ across budgets, so connecting lines are descriptive rather than a common-cohort comparison.

Interpretation. Both high-budget point estimates favor CAT, but each comparison is based on only seven discordant theorem pairs, and both reported 95% intervals include zero. These results are therefore inconclusive about the magnitude of the alignment benefit at $N = 1024$ and do not establish a trend from $N = 256$ to $N = 1024$. We report them for completeness rather than using them as evidence of increasing alignment gains at large budgets.

I Priors on Off-Trace Behaviour

In Section C, we characterized the gap between our trace-supported surrogate and the deployed search. However, we haven't deeply explored if we can use this to meaningfully improve training. We implied that placing priors on off-trace seems like a reasonable way to utilize this observation. We use this section to take a brief step in that direction and see if a simple prior on loss can be beneficial.

I.1 A two-parameter prior

Proposition 3 and Proposition 6 show that off-trace behaviour enters the objective through two scalars:

- $\kappa \geq 1$, the expected number of expansions a miss consumes, counting the off-trace excursion before the search returns to the trace;
- $q \in [0, 1)$, the probability that a miss launches an excursion that never returns.

Both enter mechanically and in a strategy-independent way. Proposition 3 states that a deployment in which a miss costs κ expansions realises the surrogate at the deflated budget

$$N_{\text{eff}} \;=\; L + \frac{N - L}{\kappa} \;\leq\; N, \tag{47}$$

so κ is applied by evaluating the strategy's own objective at, N_{eff} in place of N. Absorption enters the trace-supported recurrence Equation (5) by splitting the miss branch: with probability $(1 - p_t)q$ the search is absorbed and contributes nothing, and with probability $(1 - p_t)(1 - q)$ it continues as the uncorrected recurrence prescribes. Every occurrence of $(1 - p_t)$ on a continuation branch is therefore replaced by $(1 - p_t)(1 - q)$, with the missing mass leaking to a terminal failure state.

Note that a better prior can make assumptions on κ and q as functions of p or prior probabilities, but we are merely going to test the simple case.

I.2 Applying the prior to each strategy

The prior is applied to the objective the surrogate implies for each strategy.

Best-first search. The trace-supported objective for BFS is the retry recurrence Equation (13). The prior is applied directly to that recurrence,

$$\widehat{J}^{\kappa,q}(b, t) = p_t\, \widehat{J}^{\kappa,q}(b - 1,\, t + 1) + (1 - p_t)(1 - q)\, \widehat{J}^{\kappa,q}(b - 1,\, t), \tag{48}$$

with $\widehat{J}^{\kappa,q}(b, L) = 1$, $\widehat{J}^{\kappa,q}(0, t) = 0$, run to $b = N_{\text{eff}} = L + (N - L)/\kappa$.

RDFS and VDFS. The same substitution applies to the dynamic programs of Section D: the budget axis runs to N_{eff} and the backtrack branch carries the factor $(1 - q)$, leaving the uniform and value-weighted backtrack kernels otherwise unchanged.

MCTS. Similarly, we just readjust the dynamic program in essentially the same fashion as the other search strategies.

I.3 Setup

The prior is set by hand. The primary setting is $\kappa = 5$, $q = 0.02$; under BFS we additionally train $\kappa = 3$, $q = 0$ to probe sensitivity. Everything else matches Section 5.1: the same warm-up checkpoint, one alignment epoch over the same examples in the same order under the same optimiser settings, a budget of $N = 256$ node expansions, and the full 458-theorem pool. The only difference between a prior-corrected adapter and its uncorrected counterpart is the scalar weight applied to each tactic's cross-entropy gradient. Each adapter is deployed under the strategy whose objective it corrects.

Table 7: The off-trace prior applied to each strategy's own CAT objective (proofs found, %, N=256 node expansions, 458 theorems). "CAT" is the uncorrected weight of Sections 3 and D, i.e. the $(\kappa, q) = (1, 0)$ member of the family.

Strategy	Prior	SFT	CAT	+prior
BFS	κ=5, q=0.02	23.8	25.1	**25.8**
	κ=3, q=0	23.8	**25.1**	**25.1**
RDFS	κ=5, q=0.02	20.7	**23.6**	22.7
VDFS	κ=5, q=0.02	21.2	23.8	**24.2**
MCTS	κ=5, q=0.02	20.3	22.5	**22.9**

I.4 RESULTS

Table 7 reports the strategies for which the prior was applied. Generally, the priors edged out the basic CAT weights but not by a significant amount in any case. The conclusion we take from this is that perhaps with optimal tuning of the prior, we can get increased gains, but the value is marginal.

J Search-Agnostic Uniform-Allocation Objective: Additional Details

This appendix supplements the search-agnostic UA objective introduced in Section 2.5. UA accounts for the compute budget through a uniform local allocation without specifying the deployed search rule. We relate its weighting function to the shared-budget retry objective, providing an additional interpretation of how the two objectives account for compute.

From shared-budget to local criticality. Consider the retry objective in Equation (14), with independent $T_t \sim \mathrm{Geom}(p_t)$, $p_t \in (0,1)$, and integer $N \geq L$. Write

$$S_L = \sum_{s=1}^{L} T_s, \qquad R_t = \sum_{s \neq t} T_s, \qquad B_t = N - R_t.$$

Here, B_t is the budget left for step t after accounting for the waiting times of all other steps. For integer $k \geq 1$, define the success probability and gradient weight of a single-tactic retry problem by

$$\begin{aligned} F(p,k) &= 1-(1-p)^k, \\ h(p,k) &= \frac{\partial \log F(p,k)}{\partial \log p} = \frac{kp(1-p)^{k-1}}{1-(1-p)^k}. \end{aligned} \tag{49}$$

The shared-budget retry weight then has the exact representation

$$\widehat{w}_t^{\mathrm{retry}}(N) = \mathbb{E}[h(p_t, B_t) \,|\, S_L \leq N]\,. \tag{50}$$

To see this, extend $F(p,k) = 0$ for $k \leq 0$ and write $\widehat{\mathcal{J}}^{\mathrm{retry}}(N) = \mathbb{E}[F(p_t, B_t)]$. The distribution of B_t does not depend on p_t, so differentiating and dividing by the success probability weights each residual budget by $\mathbb{P}(B_t = k)F(p_t,k)/\widehat{\mathcal{J}}^{\mathrm{retry}}(N) = \mathbb{P}(B_t = k \mid S_L \leq N)$, giving Equation (50). Only $k \geq 1$ occurs under this conditional distribution.

Thus, the shared-budget weight combines two ingredients: the local criticality $h(p_t, k)$ and a distribution over the budget available to that step. The other tactic probabilities enter through this budget distribution.

Uniform allocation and objective. The UA objective instead assigns each demonstrated step a uniform local allocation,

$$\bar{K} = \frac{N}{L}. \tag{51}$$

This is a modeling choice, not an estimate of $\mathbb{E}[B_t \mid S_L \leq N]$ or an average over deployed search strategies. It yields the factorized objective

$$\widehat{\mathcal{J}}^{\,\mathrm{UA}}(\theta) = \prod_{t=1}^{L} \left[1-(1-p_t)^{\bar{K}}\right]. \tag{52}$$

For integer $\bar{K}$, this is the probability that every demonstrated step succeeds within its own quota of $\bar{K}$ independent attempts, without transferring unused quota between steps. Noninteger N/L uses the continuous extension of the same formula. Applying the gradient identity in Equation (7) gives

$$\widehat{w}_t^{\mathrm{UA}} = \frac{\bar{K}p_t(1-p_t)^{\bar{K}-1}}{1-(1-p_t)^{\bar{K}}} = h(p_t, \bar{K}). \tag{53}$$

Comparing Equations (50) and (53), UA evaluates the local criticality at a fixed allocation, whereas the shared-budget retry objective averages it over the conditional distribution of the residual budget. The UA weight is the exact gradient weight of its own objective in Equation (52). This connection does not require UA to approximate the dynamics of a particular deployed search algorithm.

Properties of the uniform-allocation weights. UA is search-agnostic, but not compute-agnostic. At $\bar{K} = 1$, Equation (52) reduces to the demonstrated trace likelihood and all weights equal one, recovering CE. For fixed $\bar{K} \geq 1$, the weight approaches one as $p_t \to 0$; for fixed $p_t \in (0,1)$, it approaches zero as $\bar{K} \to \infty$. Thus, tactics that are unlikely to be found within the local budget retain high weight, whereas tactics that can already be recovered through repeated sampling receive less emphasis. Unlike the shared-budget retry weight, the UA weight does not depend on the other steps' consumption of a shared budget. At fixed N and L, each weight depends only on p_t.

Interpretation. The shared-budget and UA objectives retain the same local retry criticality but differ in their treatment of compute allocation. As discussed in Section 4, demonstrations alone do not generally determine deployed gradient weights because they omit alternative proofs and off-trace recovery behavior. The empirical comparison in Table 1 evaluates the usefulness of search-aware CAT and search-agnostic UA for training; it does not establish which objective more accurately approximates the deployed gradient weights.

K LLM DISCLOSURE

LLMs were used in 3 primary ways. The first is essentially all experiments were set up and instantiated with LLM coding. We had a separate LLM check for correctness of the code. The second is when the paper was drafted, we repeatedly asked LLMs to critique and we incorporated its feedback if we deemed it to be appropriate. The third was finding related papers. We read all of these before ever citing them to confirm the relation to our work.