Agentic Systems as *Boosting* Weak Reasoning Models

Varun Sunkaraneni[*] Texas A&M University

Pierfrancesco Beneventano[*] MIT

Riccardo Neumarker MIT

Tomaso Poggio MIT

Tomer Galanti[†] Texas A&M University

Abstract

Can a committee of weak reasoning-model calls reach the performance of much stronger models? We study verifier-backed committee search as inference-time boosting for reasoning language models. The mechanism is not simply that "more agents help": samples expose latent correct solutions, while critics and comparators must recover them without access to the hidden verifier. We formalize this view by separating proposal coverage, local identifiability, progress, and diversity. We prove that coverage can be amplified by repeated sampling, but cannot by itself create useful critics or comparators; reliable amplification requires an additional local soundness signal, such as execution, proof checking, type checking, tests, or constraint solving. We give rank-based bounds showing when local selection errors compose into reliable trajectories, and characterize the proposer-side ceiling: oracle best-of-k converges only to the mass of task slices on which the proposal system assigns nonzero useful probability. Empirically, on SWE-bench Verified, a single `GPT-5.4 nano` proposal solves 67.0% of tasks. Using the same nano model, our critic–comparator orchestration reaches 76.4% with $k = 8$ proposals, matching the standalone performance of `Gemini 3 Pro` and `Claude Opus 4.5` Thinking and approaching the 79.0% oracle best-of-8 upper bound. Thus, many correct patches are already present in weak-model proposal pools; the main challenge is selecting them. The remaining failures are mostly proposal-coverage failures, indicating shared blind spots that stronger selection alone cannot close.

1 Introduction

Boosting turns weak predictors into strong predictors by repeatedly combining imperfect but useful signals [1–3]. Modern language-model systems use a related idea at inference time: they sample several candidates, check or compare them, search over partial states, and select a final output [4–9]. However, reasoning is not ordinary supervised boosting. In supervised prediction, each weak learner returns a label that can be evaluated against training examples. In reasoning, the system must instead generate an intermediate move, decide whether that move is useful, and avoid letting small local errors accumulate into a wrong final answer.

We study this mechanism for verifier-backed reasoning tasks such as code repair, theorem proving, and program synthesis. These domains provide tests, proof checkers, type checkers, execution, or constraint solvers that can supply local soundness signals [10–15]. We model agentic systems as *inference-time boosting* for reasoning language models: repeated weak proposals increase the chance of producing a useful next move, critics or comparators help identify that move, and verifier-backed progress allows useful moves to be chained into a terminal solution.

[*]Equal contribution. Random coin flip determined author order between the first two authors.

[†]Corresponding author: galanti@tamu.edu.

Preprint.

The analysis separates four quantities: proposal coverage, local selection signal or identifiability, progress, and diversity. Coverage asks whether a good move appears; identifiability asks whether the system can recognize it; progress makes local choices compose; diversity determines whether more calls escape different failure modes. This separation is essential. Sampling more candidates can increase the chance that a useful move appears, but sampling alone does not explain how the system recognizes that move. Final-answer verification is also not enough: for a multi-step task, the system needs intermediate states where progress can be generated, checked, and safely composed. On the proposer side, more calls reduce ordinary sampling noise, but they cannot fix shared blind spots. If all proposers assign near-zero probability to the useful moves needed for a particular type of instance, then even an ideal critic cannot recover those moves from the sample pool. Thus best-of-k with an oracle critic measures an upper bound on what inference-time selection can recover from the proposed candidates, not the full capability of an ideal reasoner.

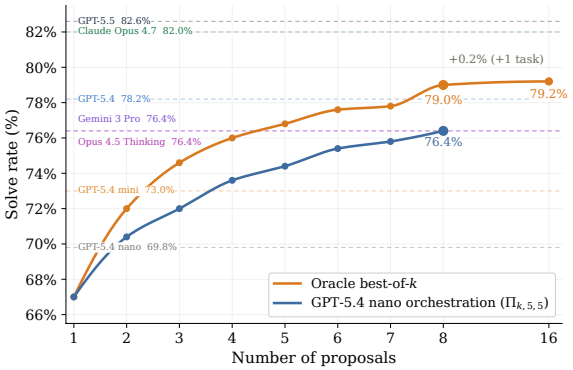

Figure 1: **A committee of `GPT-5.4 nano` calls reaches much stronger models.** Increasing proposer diversity lifts nano orchestration far above the nano baseline and up to `Gemini 3 Pro` and `Claude Opus 4.5` Thinking. The oracle best-of-n curve shows that correct solutions are often already in the proposal pool; the remaining gap is selection. Dashed lines denote single-model resolve rates.

This viewpoint also changes how such systems should be evaluated. Pass@1 measures one-shot generation. Oracle best-of-k measures whether a correct solution appears anywhere in the sampled candidate pool. Implemented system success measures how much of this oracle gap is recovered by a finite critic, comparator, verifier, or search harness. Budgeted success curves measure how quickly the harness approaches its best achievable performance as the number of calls, checks, or search steps increases.

Running example. Consider SWE-bench Verified [12]. The system receives a repository, an issue description, and visible tests, but success is measured by hidden tests. A single patch may choose the wrong design, miss a cross-file dependency, or overfit visible tests. A diverse nano pool may already contain a hidden-test-passing patch, but this latent capability matters only if the harness can recover it. Figure 1 shows this effect: `GPT-5.4 nano` starts at 67.0%, while our reviewer-comparator harness reaches 76.4%, matching `Gemini 3 Pro` and `Claude Opus 4.5` Thinking and exceeding `GPT-5.4 mini`. The oracle best-of-k curve reaches 79.0%, showing that correct patches are often already in the nano proposal pool. Thus, the harness-oracle gap diagnoses selection, while the oracle-stronger-model gap reflects remaining generation and shared-blind-spot limitations.

Contributions. This paper makes five contributions.

(i) **Inference-time boosting for reasoning language models.** We formalize verifier-backed sample–identify–advance systems over partial reasoning states as inference-time boosting of LLMs. We isolate four amplification quantities: proposal coverage, local identifiability, progress depth, and diversity.

(ii) **Coverage does not imply identifiability.** We prove a black-box separation: nontrivial probability of generating progressing-sound moves does not by itself yield a useful critic or comparator. Reliable amplification requires an additional local identifiability signal, such as execution, proof checking, type checking, constraints, tests, or a learned reviewer.

(iii) **Local-to-global oracle–identifiability bounds.** We decompose failure into an oracle miss term, asking whether any of the k proposals contains a progressing-sound move, and an identifiability miss term, asking whether finite critics/comparators recover one. Along rank-bounded trajectories these errors add: $\Pr(\text{failure}) \le L(\varepsilon_{\text{orc}}(k) + \varepsilon_{\text{id}}(k, m, r))$, with $\varepsilon_{\text{id}}(k, m, r) \lesssim k^2 e^{-\beta m - 2r\sigma^2}$.

(iv) **Blind-spot ceilings and boostable capability.** We characterize the oracle miss term under a latent subpopulation model as $\varepsilon_{\text{orc}}(k) = B + o(k)$, where B is blind-spot mass and $o(k)$ is

finite-sampling residual. Hence oracle best-of-k converges to $1 - B$, giving a formal boostable-capability ceiling for a proposal system.

(v) **Weak-to-frontier empirical amplification.** On SWE-bench Verified, critic–comparator orchestration lifts `GPT-5.4 nano` from weak one-shot performance to the level of substantially stronger standalone models. Ablations show the mechanism: diversity exposes latent correct patches, critics filter flawed candidates, comparators rank plausible alternatives, and remaining failures are mostly proposal-coverage failures.

2 Related Work

Boosting and inference-time amplification. Classical boosting converts weak predictive edges into strong predictors under supervised feedback [1–3]. The analogy is useful but incomplete for verifier-backed reasoning: the system must generate a useful local move, identify it, and repeat this without losing progress. Prior work studies LLMs as weak learners, weak-to-strong generalization, and boosting-style uses of language models [16–18]. We instead study the black-box inference-time regime, where model weights are fixed and the question is when repeated calls plus local selection amplify reasoning.

Sampling and test-time scaling. Many inference-time methods improve performance by sampling more candidates. Self-consistency, rationale ensembles, universal self-consistency, multi-agent voting, and large-scale repeated sampling all exploit the fact that useful answers may appear away from the greedy path [4–6, 19, 20]. Recent work studies scaling laws for more LM calls, best-of-N selection, and broader test-time scaling [7, 8, 21]. Our contribution is structural: sampling helps only when it exposes useful candidates, and its ceiling is set by shared blind spots.

Selection, verifiers, and judges. A second line studies how to choose among generated candidates. Learned verifiers, process reward models, pairwise rankers, multi-verifier systems, reward-model benchmarks, and tool-backed checking show that selection can be as important as generation [10, 22–30]. These works motivate our coverage–identifiability separation. A correct candidate in the sample pool is useful only if the harness has a critic, comparator, verifier, or test signal strong enough to recover it.

Search, agents, and compound systems. Inference-time reasoning often proceeds over partial states rather than flat answers. Least-to-most prompting, Tree of Thoughts, RAP, LATS, ReAct, Reflexion, and Self-Refine use decomposition, search, feedback, planning, or tool interaction [9, 31–36]. Multi-agent and compound-system frameworks such as CAMEL, AutoGen, MetaGPT, Mixture-of-Agents, Archon, and Smoothie explore larger orchestration spaces [37–42]. We isolate one mechanism inside this design space: verifier-backed committee search over bounded-depth trajectories.

Verifier-backed benchmarks and oversight. Code and formal-reasoning benchmarks make this mechanism concrete because candidates can often be checked by tests, execution, types, proof checkers, or other local signals. AlphaCode, Codex, SWE-bench, SWE-agent, AutoCodeRover, and Agentless all use some combination of sampling, localization, repair, validation, and tool-backed selection [12–15, 43, 44]. Debate, prover-verifier games, and scalable oversight also decompose hard judgments into simpler checks or comparisons, but usually study argument evaluation, supervision, or strategic interaction rather than verifier-backed local actions [45–52]. Our setting is narrower and more mechanistic: local moves, local signals, bounded progress, and measurable blind spots.

3 Verifier-Backed Committee Search

We model verifier-backed agent systems as bounded-depth search over partial objects with local progress. Let $\mathcal{X}$ denote a family of tasks, such as SWE-bench, and fix a task instance $x \in \mathcal{X}$, for example a particular SWE-bench problem.

Our setting is inspired by reinforcement learning and consists of states and actions. We view an agentic workflow through the sequence of states it induces, and the system stops when it reaches a terminal state. A state represents a partial reasoning object, such as an intermediate proof state or a partially written program together with its current specification.

Let $\mathcal{S}_x$ be a countable state space. Let $\mathrm{Valid}_x \subseteq \mathcal{S}_x$ be the set of valid states, where validity means that some correct completion remains possible. Let $s_0(x) \in \mathrm{Valid}_x$ be the initial state. Let $R_x : \mathcal{S}_x \to \{0,1\}$ be a verifier over states, and let $\mathrm{Term}_x \subseteq \mathrm{Valid}_x$ be the set of terminal states. Terminal states are accepted by the verifier, meaning that $R_x(s) = 1$ for all $s \in \mathrm{Term}_x$.

Definition 1 (Setting: valid state system with progress)**.** *A* valid state system *is a tuple containing* $\mathcal{S}_x, s_0(x), \mathrm{Valid}_x, \mathrm{Term}_x$*, and action sets* $\mathcal{A}(s)$ *where an action* $a \in \mathcal{A}(s)$ *leads to the next state, and a rank function* $d_x : \mathrm{Valid}_x \to \{0, 1, \ldots, L_x\}$ *describing the "distance from solution" such that*

$$d_x(s) = 0 \text{ for } s \in \mathrm{Term}_x, \quad \text{and} \quad d_x(s) \geq 1 \text{ for } s \in \mathrm{Valid}_x \setminus \mathrm{Term}_x.$$

A state is reachable if it is obtained from $s_0(x)$ *by finitely many actions.*

The rank certifies progress: if every chosen action decreases the rank while preserving validity, the process terminates within L_x steps.

Definition 2 (Progressing-sound actions)**.** *At a reachable valid nonterminal state* s*, an action* $a \in \mathcal{A}(s)$ *is* progressing-sound *if* s_a^* *obtained by using* a *from* s *satisfies*

$$s_a^* \in \mathrm{Valid}_x \qquad \text{and} \qquad d_x(s_a^*) < d_x(s).$$

Write the set of sound actions $\mathrm{Sound}_x(s) := \{a \in \mathcal{A}(s) : s_a^* \in \mathrm{Valid}_x \text{ and } d_x(s_a^*) < d_x(s)\}$.

For example, in SWE-bench, the state contains the current repository worktree, the issue, and the visible tests. A progressing-sound action is a code edit that preserves some hidden-test-passing patch while reducing the remaining work, such as by fixing failing visible tests. Visible tests, types, and linters reject many unsound edits but do not certify correctness, since hidden tests may still fail.

Committee rotocol $\Pi_{k,m,r}$**.** At each reachable nonterminal state s, the protocol:

(1) samples k candidate actions from the proposer harness;
(2) applies m independent critic calls per candidate and discards candidates rejected at least once;
(3) declares local failure if no candidate survives;
(4) otherwise selects a Copeland winner among survivors using r comparator votes per pair, applies it, and repeats.

The protocol separates generation from identification. Proposers create breadth, critics remove locally refutable errors, and comparators select among surviving candidates. The theory below shows that this architecture amplifies weak local competence only when two distinct resources are present: proposal coverage and local identifiability.

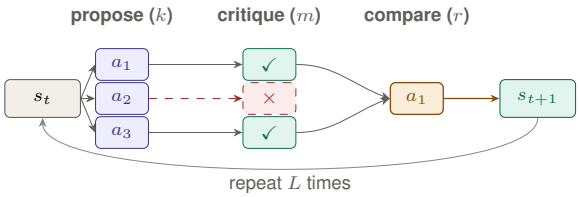

Figure 2: One step of the committee protocol.

Assumption 1 (Per-state local coverage)**.** *For each problem size* N*, there exists a proposer portfolio* P_N *with* $|P_N| = \mathrm{poly}(N)$ *such that for every reachable valid nonterminal state* s *and input* x *of size* N*, some proposer policy or prompt* $p_s \in P_N$ *satisfies*

$$\alpha(s, p_s) := \mathbb{P}[\mathrm{LLM}(p_s) \text{ outputs an action in } \mathrm{Sound}_x(s)] \geq \alpha_0 > 0.$$

Assumption 2 (Efficient local identifiability)**.** *At every valid nonterminal state* s*, there exist polynomial-time randomized procedures* Crit_s *and* Comp_s *and constants* $\beta_0, \sigma_0 > 0$ *such that:*

(i) $a \in \mathrm{Sound}_x(s) \Rightarrow \mathsf{Crit}_s(a)$ *never emits a verified rejection;*
(ii) $a \notin \mathrm{Sound}_x(s) \Rightarrow \mathbb{P}[\mathsf{Crit}_s(a) = \textsc{reject}] \geq \beta_0$*;*
(iii) if $a \in \mathrm{Sound}_x(s)$ *and* $b \notin \mathrm{Sound}_x(s)$*, then* $\mathbb{P}[\mathsf{Comp}_s(a, b) = a] \geq 1/2 + \sigma_0$.

Assumption 1 says that good moves can be generated. Assumption 2 says that bad moves can be identified or ranked below good ones. These are different capabilities.

4 From Coverage to Reliable Local Steps

Assumption 1 is a generation condition, not a verification condition. It says that some portfolio policy can sample a progressing-sound action, but not how to recognize one. This section shows that

sampling can amplify coverage but cannot manufacture critics or comparators; local identifiability must come from an additional soundness signal.

Proposition 1 (Coverage does not imply local identifiability)**.** *For every $M \geq 2$, there exists a one-step task family with action set $\mathcal{A} = \{1, \ldots, M\}$ and hidden world parameter $\theta \in \{1, \ldots, M\}$ such that the proposer distribution is $\mathrm{Unif}(\mathcal{A})$ in every world, the progressing-sound set is $\mathrm{Sound}_\theta = \mathcal{A} \setminus \{\theta\}$, and no procedure observing only candidate actions and polynomially many samples from the proposer distribution has a uniform critic or comparator edge over all worlds.*

Thus Assumption 1$\Rightarrow (\beta, \sigma)$ fails in general. Local identifiability needs an accessible signal—proof checking, execution, type checking, constraint solving, or another certificate mechanism—that can reject or rank bad moves.

Theorem 1 (Bridge theorem: coverage plus identifiability)**.** *Under Assumption 1 with proposer portfolio P_N and Assumption 2 with edges (β_0, σ_0). At every reachable valid nonterminal state s, round-robin assignment over the proposer portfolio with $k \geq |P_N| \lceil \ln(1/\delta_{\mathrm{prop}})/\alpha_0 \rceil$ proposer calls gives*

$$\alpha_{\mathrm{committee}}(s) \geq 1 - (1 - \alpha_0)^{\lfloor k/|P_N| \rfloor} \geq 1 - \delta_{\mathrm{prop}}.$$

The critic and comparator edges remain $\beta(s) \geq \beta_0$ and $\sigma(s) \geq \sigma_0$.

The theorem separates two resources: portfolio calls amplify the chance that a progressing-sound action appears, while critic and comparator edges come from Assumption 2. The proof is in Appendix D.

Verifier-backed instantiation. If a one-sided local verifier can reject unsound moves, it supplies Assumption 2. This is stronger than final-answer verification: the decomposition must expose useful local checks.

Assumption 3 (One-sided local verifier)**.** *At every reachable valid nonterminal state s, there exists a poly-time randomized verifier $V_x(s, a) \in \{\textsc{accept}, \textsc{reject}\}$ such that $a \in \mathrm{Sound}_x(s)$ implies $V_x(s, a) = \textsc{accept}$ almost surely, while $a \notin \mathrm{Sound}_x(s)$ implies $\mathbb{P}[V_x(s, a) = \textsc{reject}] \geq 1 - \nu$.*

Corollary 1 (Verifier-backed bridge)**.** *Under Assumptions 1, 2, and 3 holds with $\beta_0 = 1 - \nu$ and $\sigma_0 = (1 - \nu)/2$.*

The corollary uses the verifier as the critic. For comparison, we verify both candidates, choose the unrejected candidate when exactly one is rejected, and break ties uniformly. The full argument is in Appendix D.

5 Amplification Along a Trajectory

The previous section bounds the local committee error. We now show how to reduce global failure to local committee error. For an input x, define

$$\mathrm{err}_x(k, m, r) := \mathbb{P}(R_x(\Pi_{k,m,r}(x)) = 0)\,,$$

the probability that the full protocol fails.

At a reachable valid nonterminal state s, let

$$\varepsilon_{\mathrm{loc}}(s) := \mathbb{P}(A_t \notin \mathrm{Sound}_x(s) \mid S_t = s)\,,$$

where local failure is counted as selecting an unsound action. This is the one-step error of the committee. It has two sources:

$$\varepsilon_{\mathrm{loc}}(s) \leq \underbrace{\varepsilon_{\mathrm{prop}}(k; s)}_{\text{no good proposal}} + \underbrace{k^2 e^{-\beta m - 2r\sigma^2}}_{\text{bad proposal survives and wins}}\,.$$

The first term is proposal failure; the second is identification failure. If each reachable local step fails with probability at most ε and the trajectory depth is at most L_x, then $\mathrm{err}_x(k, m, r) \leq L_x \varepsilon$.

Lemma 1 (Adaptive cumulative-error bound)**.** *Let x have rank bound L_x. Suppose that along the protocol trajectory, $\mathbb{P}(A_t \notin \mathrm{Sound}_x(S_t) \mid \mathcal{F}_t) \leq \varepsilon_t$ whenever $S_t \in \mathrm{Valid}_x \setminus \mathrm{Term}_x$, with local failure counted as an invalid action. Then $\mathrm{err}_x(k, m, r) \leq \sum_{t=0}^{L_x - 1} \mathbb{E}[\varepsilon_t]$. In particular, if $\varepsilon_t \leq \varepsilon$ for all t, then $\mathrm{err}_x(k, m, r) \leq L_x \varepsilon$.*

At a fixed reachable valid nonterminal state s, let $\varepsilon_{\mathrm{prop}}(k;s)$ be the probability, conditional on reaching s, that none of the k proposers outputs an action in $\mathrm{Sound}_x(s)$. Let $\varepsilon_{\mathrm{loc}}(s)$ be the conditional probability that the local committee step either declares local failure or selects an action outside $\mathrm{Sound}_x(s)$.

Theorem 2 (Local error decomposition)**.** *At any reachable valid nonterminal state s, under the local role model with critic edge β and comparator edge σ, and under the conditional independence assumptions stated in Appendix E,*

$$\varepsilon_{\mathrm{loc}}(s) \le \varepsilon_{\mathrm{prop}}(k;s) + k^2(1-\beta)^m e^{-2r\sigma^2} \le \varepsilon_{\mathrm{prop}}(k;s) + k^2 e^{-\beta m - 2r\sigma^2}.$$

If the proposer calls are conditionally independent and each succeeds with probability at least α, then $\varepsilon_{\mathrm{prop}}(k;s) \le (1-\alpha)^k \le e^{-\alpha k}$.

Sizing rule for global success. Let $\varepsilon_{\mathrm{prop}}(k) := \sup_s \varepsilon_{\mathrm{prop}}(k;s)$, where the supremum is over reachable valid nonterminal states. Combining the cumulative and local bounds gives

$$\mathrm{err}_x(k,m,r) \le L_x\big(\varepsilon_{\mathrm{prop}}(k) + k^2 e^{-\beta m - 2r\sigma^2}\big).$$

Thus, if $\varepsilon_{\mathrm{prop}}(k) + k^2 e^{-\beta m - 2r\sigma^2} \le \delta/L_x$, then the full depth-$L_x$ protocol fails with probability at most δ.

The bound also yields a standard polynomial-resource corollary when the rank, proposal coverage, critic/comparator edges, and per-call runtimes are polynomially bounded; we state this consequence in Appendix B.

6 Oracle Error and Blind-Spot Limits

Section 5 reduced global failure to proposal failure plus identifiability failure. This section studies the proposal term. If no progressing-sound action appears among the k proposals, then even a perfect local identifier cannot help. The main point is that this oracle miss probability splits into an irreducible blind-spot floor and a finite-sampling residual. For extensions and additional details, see Appendix F.

Lemma 2 (Local oracle miss and blind-spot floor)**.** *Fix a reachable valid nonterminal state s. Suppose that, conditional on a latent variable Z, the k proposal calls are independent and each lies in $\mathrm{Sound}_x(s)$ with probability $q_s(Z)$. Then*

$$\varepsilon_{\mathrm{prop}}(k;s) = \mathbb{E}[(1-q_s(Z))^k].$$

Equivalently,

$$\varepsilon_{\mathrm{prop}}(k;s) = B_s + R_k(s),$$

where

$$B_s := \mathbb{P}(q_s(Z)=0), \qquad R_k(s) := \mathbb{E}[(1-q_s(Z))^k \mathbf{1}\{q_s(Z)>0\}].$$

In particular, $\varepsilon_{\mathrm{prop}}(k;s) \to B_s$ as $k \to \infty$.

The term B_s is the local blind-spot floor: on these latent subpopulations, the proposal system assigns zero probability to any progressing-sound action. The term $R_k(s)$ is finite-sampling error on non-blind subpopulations. Thus more proposals can reduce $R_k(s)$, but they cannot reduce B_s. Reducing B_s requires changing the proposal system itself: the model, prompts, tools, retrieval, decomposition, or proposer diversity. An average coverage condition can still hide blind spots: it is possible that $\mathbb{E}[q_s(Z)] > 0$ while $q_s(Z) = 0$ on a nontrivial subpopulation. The floor vanishes only under a stronger conditional coverage condition, for example $q_s(Z) \ge \alpha_0$ almost surely. Lemma 2 is useful precisely because it separates average proposal success from latent subpopulations with zero proposal mass.

Let

$$B := \sup_s B_s, \qquad R_k := \sup_s R_k(s),$$

where the suprema range over reachable valid nonterminal states. Combining Lemma 2 with Theorem 2 and the cumulative bound gives

$$\mathrm{err}_x(k,m,r) \le L_x\Big[\underbrace{B}_{\text{blind spots}} + \underbrace{R_k}_{\text{finite proposal sampling}} + \underbrace{k^2 e^{-\beta m - 2r\sigma^2}}_{\text{identifiability error}}\Big].$$

This is the main error decomposition. The first term is irreducible for a fixed proposal system, the second decreases with proposal width, and the third decreases with critic and comparator resources. The appendix gives the heterogeneous-portfolio version of Lemma 2, finite-k convergence rates for R_k, and the common-shock specialization.

The same calculation also gives the task-level oracle quantities used in evaluation. The theorem above is local: $q_s(Z)$ is the probability of proposing a progressing-sound next action at state s. For complete outputs, define

$$p_1(P) := \mathbb{P}_{x,Y\sim P(\cdot|x)}(R_x(Y)=1), \qquad p_{\mathrm{oracle}}(k;P) := \mathbb{P}(\exists i \le k:\ R_x(Y_i)=1).$$

If $q_P(Z) := \mathbb{P}(R_x(Y)=1 \mid Z)$, then

$$p_{\mathrm{oracle}}(k;P) = 1 - \mathbb{E}[(1-q_P(Z))^k], \qquad \lim_{k\to\infty} p_{\mathrm{oracle}}(k;P) = 1 - \mathbb{P}(q_P(Z)=0).$$

Precisely, pass@1 measures one-shot success, oracle best-of-k measures terminal boostable capability, and the limiting oracle curve measures the boostable ceiling of the proposal system.

For an implemented system evaluated on the same candidate pool, let $p_{\mathrm{system}}(k,m,r;P)$ denote its success probability and define oracle-gap recovery by

$$\mathrm{Rec}(k,m,r;P) := \frac{p_{\mathrm{system}}(k,m,r;P) - p_1(P)}{p_{\mathrm{oracle}}(k;P) - p_1(P)},$$

when the denominator is positive. This measures how much of the oracle-exposed capability is recovered by the implemented selector, rather than lost to imperfect critics, verifiers, or comparators.

This gives a role-wise benchmark decomposition. Pass@1 measures the *one-shot capability* of the proposal system. Oracle best-of-k measures *the capability that is latent in the proposal distribution* under perfect identification. The gap between oracle best-of-k and pass@1 measures *how much there is to recover by boosting*. The gap between oracle best-of-k and the implemented system measures *the remaining identifiability bottleneck*. Thus these quantities tell us whether failures come from weak proposal coverage, shared blind spots, or insufficient local identifiability.

7 Experiments

We use SWE-bench Verified [12] as a diagnostic testbed for inference-time orchestration. Beyond testing whether additional calls improve solve rate, we ask where the gains come from. In the theory's terminology, the experiment separates proposal coverage, critic filtering, comparator ranking, and residual shared blind spots.

The design estimates three role-wise quantities. First, the oracle best-of-k curve measures proposal coverage: whether a correct patch appears in the generated pool. Second, the gap between deployed orchestration and the oracle measures harness recovery: how much latent pool capability the selector recovers. Third, tasks unreachable by any generated proposal estimate the remaining coverage failure, or shared-blind-spot mass, of the current proposer family.

7.1 Setup

SWE-bench Verified contains 500 software-engineering tasks. Each task provides a repository, an issue description, and visible tests, while success is measured by held-out hidden tests. For each task, we generate a fixed pool of $k = 8$ candidate patches using independent `GPT-5.4 nano` proposer runs. All selector ablations reuse this same cached proposal pool. Thus, differences in solve rate reflect differences in selection rather than differences in generation.

The harness uses two local selection signals. First, binary critics evaluate individual patches using local evidence such as the issue, the patch, visible tests, and execution traces. A patch survives the

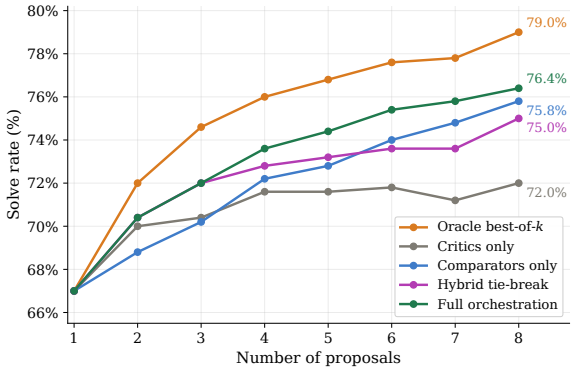

Figure 3: **Scaling with the number of proposals and orchestration components.** Solve rate as the proposal budget k increases. The oracle best-of-k curve succeeds whenever any generated proposal solves the task. Full orchestration reaches 76.4% at $k = 8$, close to the 79.0% oracle upper bound, while component-only variants remain below the full system.

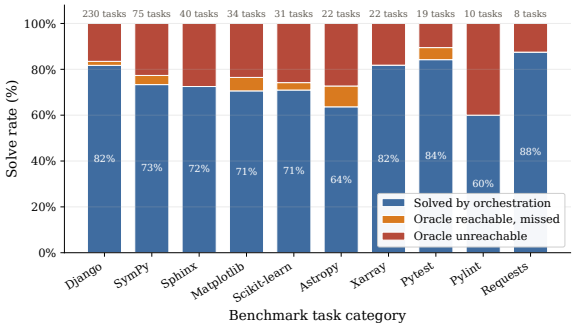

Figure 4: **Failure decomposition by benchmark category.** For each category, we decompose tasks into those solved by orchestration, those that were oracle reachable but missed by selection, and those that were oracle unreachable under the proposal budget. The small oracle-reachable-but-missed segments indicate that most remaining failures are coverage failures rather than selection failures.

critic gate if at least τ out of five critic votes judge it plausible. Second, pairwise comparators rank candidate patches against each other. Comparator outcomes are aggregated with a tournament rule to select a final patch.

We evaluate these signals separately and jointly. Critics-only selection tests how far coarse patch filtering can go. Comparator-only selection tests whether pairwise preferences can recover correct patches without a critic gate. The full harness tests whether filtering and ranking are complementary. To reduce presentation-order bias, each pair is compared in both patch orders. A pairwise win is counted only if both orders select the same patch after mapping the swapped response back to the original indices; disagreements and ties are treated as ties.

We also report oracle-gap recovery. Let p_1 denote the single-proposal solve rate, $p_{\text{oracle}}(k)$ the oracle best-of-k solve rate, and $p_{\text{system}}(k)$ the solve rate of the deployed harness using the same k-proposal pool. We define $\text{Rec}(k) := \frac{p_{\text{system}}(k) - p_1}{p_{\text{oracle}}(k) - p_1}$, whenever the denominator is positive. This measures how much of the latent proposal-pool capability exposed by oracle best-of-k is recovered by the actual selector.

We report hidden-test solve rate over all 500 tasks. We also report an oracle best-of-k upper bound, which counts a task as solved if any of the k generated patches passes the hidden tests. This oracle is not deployable, since it uses hidden-test outcomes; it serves only as a diagnostic of latent proposal coverage. Diagnostics requiring complete critic or comparator vote logs are computed on the subset of tasks with complete traces, with the denominator reported where applicable.

Full implementation details, including candidate generation, critic prompting, comparator aggregation, tie handling, and evaluation protocol, appear in Appendix C.

7.2 Results

Proposal diversity exposes latent correct patches, but does not by itself solve selection. Figure 3 shows the central scaling behavior. With a single nano proposal, solve rate is substantially below the oracle best-of-k curve. As the proposal budget increases, the oracle rises to 79.0%, showing that many tasks already contain a hidden-test-passing patch somewhere in the nano-generated

pool. This is the proposal-coverage effect: additional calls increase the probability that a correct patch is present.

However, oracle best-of-k is not a deployable system. The deployed harness must identify the correct patch without access to hidden tests. Full orchestration reaches 76.4% at $k = 8$, recovering most of the gap between one-shot nano performance and the oracle. Thus the main empirical effect is not "more samples" alone. The additional samples expose latent capability, while the critic–comparator harness converts a large fraction of that latent capability into realized solve rate.

Critics and comparators are complementary. The component ablations show that neither selection signal is sufficient on its own. Critics-only selection improves over the single-proposal baseline but plateaus well below the full system. This suggests that binary critics are useful for rejecting clearly flawed patches, but are too coarse to reliably choose among multiple plausible candidates. Comparator-only selection is stronger, indicating that direct pairwise ranking carries substantial information about patch quality. Still, it remains below full orchestration, showing that pairwise comparison also benefits from first removing implausible candidates.

This supports the role-wise picture from the theory. Critics provide a local filtering signal: they reduce the number of obviously bad actions that can win. Comparators provide a local ranking signal: they decide among candidates that survive this coarse filter. Full orchestration works best because it composes these two forms of identifiability.

Conservative comparator aggregation matters. The pairwise comparator is noisy and can be sensitive to presentation order. We therefore query each unordered pair in both orders, once with p_i shown as Patch A and once with p_j shown as Patch A, and map the second judgment back to the original patch indices. A pairwise comparison is counted as a win only when both orders agree on the same patch. If the two orders disagree, or if either order returns TIE, the pairwise outcome is treated as a tie.

This conservative rule is important conceptually. It treats comparator decisions as weak local evidence rather than as ground truth. A patch receives a pairwise win only when the preference is robust to a superficial change in presentation. Thus the comparator stage is not merely a majority vote over judge outputs; it is a debiased local-identification mechanism. The strong performance of the comparator and full-orchestration curves suggests that much of the oracle gap can be recovered by stable pairwise preferences, while the remaining gap reflects cases where either no correct proposal was generated or the available local signals were insufficient to identify it.

Remaining failures are mostly coverage failures, not selection failures. Figure 4 decomposes full-budget failures by benchmark category. For each category, we separate tasks solved by orchestration, tasks that were oracle reachable but missed by the selector, and tasks that were oracle unreachable because none of the generated proposals passed the hidden tests. The oracle-reachable-but-missed region is relatively small compared with the oracle-unreachable region in most categories. This indicates that, after critic–comparator orchestration recovers most of the best-of-k gain, the dominant remaining bottleneck is proposal coverage.

This is exactly the blind-spot diagnosis predicted by the theory. If no proposal in the pool is correct, no selector can recover a correct patch. Further gains therefore require either more diverse proposers, stronger proposal mechanisms, or new ways of targeting slices on which the current nano proposer family has low correct-patch probability. Stronger selectors may still help on the oracle-reachable-but-missed tasks, but they cannot close the oracle-unreachable gap.

8 Discussion and Limitations

The main lesson is that weak-model failure is often not a lack of information, but a failure to select it. On SWE-bench Verified, hidden-test-passing patches often appear in a pool of `GPT-5.4 nano` proposals even when a single nano call fails. Thus the central question is not only whether weak models can generate correct solutions, but whether an inference-time harness can identify them among several imperfect candidates.

Our results show that this selection problem is substantially solvable, but not for free. Critics remove candidates with clear local defects, while comparators rank the remaining plausible patches. Together, they recover most of the oracle best-of-k gap, turning latent proposal-pool capability into

actual solve rate. This explains how a committee of weak nano calls can approach much stronger standalone models: sampling exposes correct patches, and local selection makes them usable.

The same decomposition also identifies the ceiling. When a correct patch is in the pool but the harness chooses another one, the bottleneck is identifiability: better critics, comparators, tests, or aggregation can help. When no correct patch appears, the bottleneck is proposal coverage: no selector can recover an absent solution. Our remaining failures are mostly of this second kind, so further gains likely require more diverse proposers, stronger proposal mechanisms, or targeted methods for hard slices where the current proposer family has shared blind spots. More broadly, verifier-backed orchestration is most natural when tasks provide useful local evidence, and its gains must be weighed against added model calls, verification cost, latency, and system complexity.

References

[1] Robert E. Schapire. The strength of weak learnability. *Machine Learning*, 5(2):197–227, 1990.

[2] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory: Second European Conference, EuroCOLT 1995*, volume 904 of *Lecture Notes in Computer Science*, pages 23–37. Springer, 1995.

[3] Yoav Freund and Robert E. Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139, 1997.

[4] 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.

[5] Junyou Li, Qin Zhang, Yangbin Yu, Qiang Fu, and Deheng Ye. More agents is all you need. *Transactions on Machine Learning Research*, 2024.

[6] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling, 2024.

[7] Lingjiao Chen, Jared Quincy Davis, Boris Hanin, Peter Bailis, Ion Stoica, Matei A. Zaharia, and James Y. Zou. Are more LM calls all you need? towards the scaling properties of compound AI systems. In *Advances in Neural Information Processing Systems*, volume 37, pages 45767–45790, 2024.

[8] Audrey Huang, Adam Block, Dhruv Rohatgi, Cyril Zhang, Max Simchowitz, and Akshay Krishnamurthy. Is best-of-N the best of them? coverage, scaling, and optimality in inference-time alignment, 2025.

[9] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[10] 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, 2021.

[11] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step, 2023.

[12] Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R. Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations*, 2024.

[13] John Yang, Carlos E. Jimenez, Alexander Wettig, Kilian Lieret, Shunyu Yao, Karthik Narasimhan, and Ofir Press. SWE-agent: Agent-computer interfaces enable automated software engineering. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

[14] Yuntong Zhang, Haifeng Ruan, Zhiyu Fan, and Abhik Roychoudhury. AutoCodeRover: Autonomous program improvement, 2024. Published version appears in the ACM SIGSOFT/ISSTA proceedings.

[15] Chunqiu Steven Xia, Yinlin Deng, Soren Dunn, and Lingming Zhang. Demystifying LLM-based software engineering agents. *Proceedings of the ACM on Software Engineering*, 2(FSE):801–824, 2025.

[16] Hariharan Manikandan, Yiding Jiang, and J. Zico Kolter. Language models are weak learners. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[17] Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, Ilya Sutskever, and Jeffrey Wu. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 4971–5012. PMLR, 2024.

[18] Zehao Chen, Tianxiang Ai, Yifei Li, Gongxun Li, Yuyang Wei, Wang Zhou, Guanghui Li, Bin Yu, Zhijun Chen, Hailong Sun, Fuzhen Zhuang, Jianxin Li, Deqing Wang, and Yikun Ban. LLMBoost: Make large language models stronger with boosting, 2025. Preprint; also submitted to ICLR 2026 on OpenReview.

[19] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, and Denny Zhou. Rationale-augmented ensembles in language models, 2022.

[20] Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language model generation, 2023.

[21] Qiyuan Zhang, Fuyuan Lyu, Zexu Sun, Lei Wang, Weixu Zhang, Wenyue Hua, Haolun Wu, Zhihan Guo, Yufei Wang, Niklas Muennighoff, Irwin King, Xue Liu, and Chen Ma. A survey on test-time scaling in large language models: What, how, where, and how well?, 2025.

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

[23] Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. Math-shepherd: Verify and reinforce LLMs step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[24] Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. LLM-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14165–14178, Toronto, Canada, July 2023. Association for Computational Linguistics.

[25] Shalev Lifshitz, Sheila A. McIlraith, and Yilun Du. Multi-agent verification: Scaling test-time compute with multiple verifiers, 2025.

[26] Jon Saad-Falcon, E. Kelly Buchanan, Mayee F. Chen, Tzu-Heng Huang, Brendan McLaughlin, Tanvir Bhathal, Shang Zhu, Ben Athiwaratkun, Frederic Sala, Scott Linderman, Azalia Mirhoseini, and Christopher Ré. Shrinking the generation-verification gap with weak verifiers, 2025.

[27] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[28] Nathan Lambert, Valentina Pyatkin, Jacob Morrison, Lester James V. Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hanna Hajishirzi. Rewardbench: Evaluating reward models for language modeling, 2024.

[29] Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen. CRITIC: Large language models can self-correct with tool-interactive critiquing. In *International Conference on Learning Representations*, 2024.

[30] Shehzaad Dhuliawala, Mojtaba Komeili, Jing Xu, Roberta Raileanu, Xian Li, Asli Celikyilmaz, and Jason Weston. Chain-of-verification reduces hallucination in large language models. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 3563–3578, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[31] Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V. Le, and Ed H. Chi. Least-to-most prompting enables complex reasoning in large language models. In *International Conference on Learning Representations*, 2023.

[32] Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahong Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. Reasoning with language model is planning with world model. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8154–8173, Singapore, 2023. Association for Computational Linguistics.

[33] Andy Zhou, Kai Yan, Michal Shlapentokh-Rothman, Haohan Wang, and Yu-Xiong Wang. Language agent tree search unifies reasoning, acting, and planning in language models, 2023.

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

[35] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[36] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[37] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. CAMEL: Communicative agents for "mind" exploration of large scale language model society, 2023.

[38] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W. White, Doug Burger, and Chi Wang. AutoGen: Enabling next-gen LLM applications via multi-agent conversation, 2023.

[39] Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and J"urgen Schmidhuber. MetaGPT: Meta programming for a multi-agent collaborative framework. In *International Conference on Learning Representations*, 2024.

[40] Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities, 2024.

[41] Jon Saad-Falcon, Adrian Gamarra Lafuente, Shlok Natarajan, Nahum Maru, Hristo Todorov, Etash Kumar Guha, Estefany Kelly Buchanan, Mayee F. Chen, Neel Guha, Christopher Ré, and Azalia Mirhoseini. Archon: An architecture search framework for inference-time techniques, 2024.

[42] Neel Guha, Mayee F. Chen, Trevor Chow, Ishan S. Khare, and Christopher Ré. Smoothie: Label free language model routing, 2024.

[43] Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with AlphaCode. *Science*, 378(6624):1092–1097, 2022.

[44] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke

Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

[45] Geoffrey Irving, Paul Christiano, and Dario Amodei. AI safety via debate, 2018.

[46] Paul Christiano, Buck Shlegeris, and Dario Amodei. Supervising strong learners by amplifying weak experts, 2018.

[47] Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 11733–11763. PMLR, 2024.

[48] Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Shuming Shi, and Zhaopeng Tu. Encouraging divergent thinking in large language models through multi-agent debate. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17889–17904, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

[49] Jonah Brown-Cohen, Geoffrey Irving, and Georgios Piliouras. Scalable AI safety via doubly-efficient debate. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 4585–4602. PMLR, 2024.

[50] Cem Anil, Guodong Zhang, Yuhuai Wu, and Roger Grosse. Learning to give checkable answers with prover-verifier games, 2021.

[51] Jan Hendrik Kirchner, Yining Chen, Harri Edwards, Jan Leike, Nat McAleese, and Yuri Burda. Prover-verifier games improve legibility of LLM outputs, 2024.

[52] Zachary Kenton, Noah Y. Siegel, János Kramár, Jonah Brown-Cohen, Samuel Albanie, Jannis Bulian, Rishabh Agarwal, David Lindner, Yunhao Tang, Noah D. Goodman, and Rohin Shah. On scalable oversight with weak LLMs judging strong LLMs. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

[53] Leo Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[54] David H. Wolpert. Stacked generalization. *Neural Networks*, 5(2):241–259, 1992.

[55] Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer, 2000.

[56] Audrey Huang, Adam Block, Dylan J. Foster, Dhruv Rohatgi, Cyril Zhang, Max Simchowitz, and Akshay Krishnamurthy. Self-improvement in language models: The sharpening mechanism, 2024. ICLR 2025.

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

[58] Zhijun Chen, Xiaodong Lu, Jingzheng Li, Pengpeng Chen, Zhuoran Li, Kai Sun, Yuankai Luo, Qianren Mao, Ming Li, Likang Xiao, Dingqi Yang, Xiao Huang, Yikun Ban, Hailong Sun, and Philip S. Yu. Harnessing multiple large language models: A survey on LLM ensemble, 2025.

[59] V. Venktesh, Mandeep Rathee, and Avishek Anand. Trust but verify! a survey on verification design for test-time scaling, 2025.

[60] Joonhyuk Lee, Virginia Ma, Sarah Zhao, Yash Nair, Asher Spector, Regev Cohen, and Emmanuel J. Candès. FUSE: Ensembling verifiers with zero labeled data, 2026.

[61] Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel M. Ni, and Jian Guo. A survey on LLM-as-a-judge, 2024.

[62] Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. Toolformer: Language models can teach themselves to use tools. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[63] Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. HuggingGPT: Solving AI tasks with ChatGPT and its friends in Hugging Face. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[64] Yuxi Xie, Kenji Kawaguchi, Yiran Zhao, James Xu Zhao, Min-Yen Kan, Junxian He, and Michael Xie. Self-evaluation guided beam search for reasoning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.

[65] Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, Juyuan Xu, Dahai Li, Zhiyuan Liu, and Maosong Sun. ChatDev: Communicative agents for software development. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 15174–15186, Bangkok, Thailand, August 2024. Association for Computational Linguistics.

[66] Weize Chen, Yusheng Su, Jingwei Zuo, Cheng Yang, Chenfei Yuan, Chi-Min Chan, Heyang Yu, Yaxi Lu, Yi-Hsin Hung, Chen Qian, Yujia Qin, Xin Cong, Ruobing Xie, Zhiyuan Liu, Maosong Sun, and Jie Zhou. AgentVerse: Facilitating multi-agent collaboration and exploring emergent behaviors, 2023.

A Broader Impacts

This work studies inference-time amplification for verifier-backed reasoning tasks, including software engineering, theorem proving, and program synthesis. A positive implication is that stronger performance may be obtained from weaker model calls by sampling multiple candidates and selecting among them with local critics, comparators, tests, proof checkers, or constraint solvers. This could make capable reasoning systems more accessible without requiring the training or deployment of substantially larger models.

The main risk is that the same selection mechanism can amplify harmful uses of reasoning models. If a weak model can occasionally generate a useful candidate, a committee system may make that capability more reliable, including in dual-use settings such as code generation, exploit repair, or automated search over formal constraints. Our analysis also cautions against overtrusting such systems: when all proposals share a blind spot, critics and comparators can only select among flawed candidates. Deployment should therefore use sandboxing, access controls, provenance logs, held-out verification, and human review when outputs affect security, privacy, or critical infrastructure.

B Polynomial-Resource Consequence of the Main Amplification Bound

Section 5 shows that the failure probability of $\Pi_{k,m,r}$ is controlled by two local quantities: proposer failure and identification failure. In particular, the main text gives the bound

$$\mathrm{err}_x(k,m,r) \le L_x\big(\varepsilon_{\mathrm{prop}}(k) + k^2 e^{-\beta m - 2r\sigma^2}\big).$$

This appendix records the corresponding polynomial-resource regime. The result is only a sufficient condition: it identifies when the committee protocol gives polynomial reliable amplification, but does not imply that all verifier-backed tasks satisfy these assumptions.

Definition 3 (Committee-decomposable verifier family)**.** *A family of search relations $\mathcal{F} = \{R_x : x \in \mathcal{X}_N,\ N \in \mathbb{N}\}$ is* committee-decomposable *if every instance $x \in \mathcal{X}_N$ admits the ingredients used in Sections 3–5:*

(i) a valid state system with rank bound $L_x \le \mathrm{poly}(N)$;
(ii) a proposer portfolio P_N of size $\mathrm{poly}(N)$ satisfying Assumption 1;
(iii) efficient local identifiability satisfying Assumption 2;
(iv) proposer, critic, comparator, and verifier calls with runtime polynomial in N.

Definition 4 (Polynomial reliable amplification)**.** *Fix a class $\mathfrak{F}$ of search-relation families. A committee protocol achieves* polynomial reliable amplification *over $\mathfrak{F}$ if, for every $\mathcal{F} \in \mathfrak{F}$, every instance $x \in \mathcal{X}_N$, and every $\delta \in (0,1)$, it outputs a witness y satisfying $R_x(y) = 1$ with probability at least $1-\delta$, using at most $\mathrm{poly}(N, \log(1/\delta))$ resources whenever such a witness exists.*

Theorem 3 (Polynomial amplification for committee-decomposable families)**.** *Let $\mathfrak{F}$ be a class of committee-decomposable verifier families, and let $x \in \mathcal{X}_N$ be an instance with rank bound L_x. Suppose the proposer harness has a uniform proposer-failure bound $\varepsilon_{\mathrm{prop}}(k)$ over all reachable valid nonterminal states, and suppose local identifiability holds with edges (β_0, σ_0). If*

$$\varepsilon_{\mathrm{prop}}(k) + k^2 \exp(-\beta_0 m - 2\sigma_0^2 r) \le \frac{\delta}{L_x}, \tag{1}$$

then $\Pi_{k,m,r}$ outputs a witness y satisfying $R_x(y) = 1$ with probability at least $1-\delta$. The all-pairs implementation uses

$$O\big(L_x(k + mk + rk^2)\big) \tag{2}$$

role calls.

In particular, if L_x, $|P_N|$, per-call runtime, $1/\alpha_0$, $1/\beta_0$, and $1/\sigma_0$ are polynomially bounded, and if $\varepsilon_{\mathrm{prop}}(k)$ can be driven below the target accuracy with polynomially many proposer calls, then the family is solvable with $\mathrm{poly}(N, \log(1/\delta))$ resources.

Proof. Fix an instance $x \in \mathcal{X}_N$ with rank bound L_x, and suppose Eq. (1) holds:

$$\varepsilon_{\mathrm{prop}}(k) + k^2 e^{-\beta_0 m - 2\sigma_0^2 r} \le \frac{\delta}{L_x}.$$

By Theorem 2, every reachable valid nonterminal state s has local error

$$\varepsilon_{\mathrm{loc}}(s) \le \varepsilon_{\mathrm{prop}}(k) + k^2 e^{-\beta_0 m - 2\sigma_0^2 r} \le \frac{\delta}{L_x}.$$

The bound is uniform over reachable states, so Lemma 1 gives

$$\mathbb{P}(R_x(\Pi_{k,m,r}(x)) = 0) \le L_x \cdot \frac{\delta}{L_x} = \delta.$$

Thus $\Pi_{k,m,r}$ outputs a witness y satisfying $R_x(y) = 1$ with probability at least $1 - \delta$.

For the role-call bound, each step uses k proposer calls, mk critic calls, and at most $r\binom{k}{2} = O(rk^2)$ comparator calls in the all-pairs implementation. Since every successful trajectory has length at most L_x, the total number of role calls is

$$O\big(L_x(k + mk + rk^2)\big).$$

□

Proof. Fix an instance $x \in \mathcal{X}_N$ with rank bound L_x, and suppose Eq. (1) holds:

$$\varepsilon_{\mathrm{prop}}(k) + k^2 e^{-\beta_0 m - 2\sigma_0^2 r} \le \frac{\delta}{L_x}.$$

By Theorem 2, every reachable valid nonterminal state s has local error

$$\varepsilon_{\mathrm{loc}}(s) \le \varepsilon_{\mathrm{prop}}(k) + k^2 e^{-\beta_0 m - 2\sigma_0^2 r} \le \frac{\delta}{L_x}.$$

The bound is uniform over reachable states, so Lemma 1 gives

$$\mathbb{P}(R_x(\Pi_{k,m,r}(x)) = 0) \le L_x \cdot \frac{\delta}{L_x} = \delta.$$

Thus $\Pi_{k,m,r}$ outputs a witness y satisfying $R_x(y) = 1$ with probability at least $1 - \delta$.

For the role-call bound, each step uses k proposer calls, mk critic calls, and at most $r\binom{k}{2} = O(rk^2)$ comparator calls in the all-pairs implementation. Running the bounded-depth protocol for at most L_x steps therefore uses

$$O\big(L_x(k + mk + rk^2)\big)$$

role calls. □

Explicit parameter choices for the polynomial-resource claim. Under the round-robin portfolio assignment from Theorem 1, the good prompt $p_s \in P_N$ is queried at least $\lfloor k/|P_N| \rfloor$ times at every reachable state. Hence

$$\varepsilon_{\mathrm{prop}}(k) \le (1 - \alpha_0)^{\lfloor k/|P_N| \rfloor} \le \exp\left(-\alpha_0 \left\lfloor \frac{k}{|P_N|} \right\rfloor\right).$$

Thus it is enough to choose

$$k \ge |P_N| \left\lceil \frac{\log(2L_x/\delta)}{\alpha_0} \right\rceil.$$

For the identification term, it suffices that

$$k^2 e^{-\beta_0 m - 2\sigma_0^2 r} \le \frac{\delta}{2L_x},$$

equivalently,

$$\beta_0 m + 2\sigma_0^2 r \ge \log \frac{2k^2 L_x}{\delta}.$$

One separated choice is

$$m \ge \left\lceil \frac{1}{2\beta_0} \log \frac{2k^2 L_x}{\delta} \right\rceil, \qquad r \ge \left\lceil \frac{1}{4\sigma_0^2} \log \frac{2k^2 L_x}{\delta} \right\rceil.$$

Therefore, if L_x, $|P_N|$, per-call runtime, $1/\alpha_0$, $1/\beta_0$, and $1/\sigma_0$ are polynomially bounded, the required parameters and total runtime are polynomial in N and $\log(1/\delta)$.

The theorem makes explicit the efficient regime implicit in the main amplification bound. Inference-time boosting can fail to be polynomial for several different reasons: exponentially small proposal mass, weak critic or comparator edges, long progress depth, or expensive verification. Thus the result applies to the locally identifiable core of a task, not automatically to every task with a final verifier.

C Implementation Details

This appendix describes the implementation used for the SWE-bench Verified experiments in Section 7. The experiments are designed to separate proposal coverage from selection quality. We therefore first generate a fixed pool of candidate patches for each task and then reuse this same pool across all selector ablations.

C.1 Benchmark and Evaluation

We evaluate on SWE-bench Verified [12], which contains 500 real software-engineering tasks. Each task provides a repository state and an issue description. A submitted patch is counted as correct only if it passes the benchmark's held-out hidden tests. All headline solve rates are computed over the full set of 500 tasks.

Candidate patches are evaluated with the SWE-bench evaluation harness. Hidden test outcomes are used only for scoring and for oracle diagnostics. They are never exposed to the deployed selector.

C.2 Proposal Generation

For each task x, we generate a pool of $k = 8$ candidate patches

$$\mathcal{P}(x) = \{p_1(x), \ldots, p_k(x)\}$$

using independent `GPT-5.4 nano` proposer calls. Each proposer is implemented as a mini-swe-agent trajectory over the repository. The agent receives the issue and interacts with the repository environment to produce a patch. The proposer runs are independent, and their outputs are cached before any selector ablation is performed.

This fixed-pool design is important. All selector variants, including binary-judge-only selection, comparator-only selection, hybrid tie-breaking, and full orchestration, select from the same candidate patches. Therefore differences in solve rate come from the selection rule rather than from changes in proposal generation.

C.3 Oracle Best-of-k

The oracle best-of-k score measures latent proposal coverage. For each task, we evaluate every cached proposal against the hidden tests and count the task as oracle solved if at least one proposal succeeds:

$$\mathbf{1}_{\text{oracle}}(x) = \mathbf{1}\{\exists j \leq k : \ p_j(x) \text{ passes the hidden tests}\}.$$

This oracle is not deployable because it uses hidden-test outcomes. It is used only as a diagnostic: it measures whether the proposal pool contains a correct patch somewhere, independently of whether the selector can identify that patch.

C.4 Binary Patch Judges

The single-patch selection signal is implemented as a binary patch judge. For each candidate patch, the judge receives the issue text and the proposed patch and answers whether the patch resolves the issue without breaking previously working behavior. Each judgment returns a binary decision, a short rationale, and a confidence score. The prompt asks the judge to trace the failure mode from the issue to the code path and decide whether the patch changes the relevant behavior.

For each task and patch, multiple independent binary judgments are cached. A binary-judge-only selector ranks patches by the number of positive judgments, with ties broken by the lowest proposal index. We also evaluate a confidence-weighted variant, where positive judgments are weighted by the reported confidence. These cached judgments are reused for voter-count and proposal-budget ablations.

C.5 Pairwise Comparators

The pairwise comparator evaluates two candidate patches for the same issue. Given the issue and two patches, the comparator first states a hypothesis for the failure mode, lists the relevant changes

made by each patch, and then decides whether each patch is consistent with the hypothesized failure and whether it introduces collateral behavioral changes. It then outputs a winner in $\{A, B, \text{TIE}\}$ together with a confidence score.

Pairwise comparator outcomes are aggregated into a tournament. A Copeland-style rule scores each candidate by its pairwise wins and selects the highest-scoring candidate. Ties in the final Copeland score are broken deterministically by the lowest proposal index in the offline ablations. This gives a comparator-only selector when applied to the full proposal pool.

C.6 Hybrid and Full Orchestration Selectors

The hybrid selector combines the binary patch judge with pairwise comparators. First, the binary judge scores each patch. Patches that pass a fixed yes-rate threshold are retained as candidates for pairwise comparison. If no patch passes the gate, the selector falls back to the highest-scoring patch under the binary judge. Pairwise comparator aggregation is then applied only to the retained candidate set.

The full orchestration system uses the same cached proposal pool and combines the binary-judge signal with pairwise comparator aggregation. Thus, the comparison between binary-judge-only, comparator-only, hybrid tie-breaking, and full orchestration isolates the value of different selection signals while holding proposal generation fixed.

C.7 Proposal-Budget Ablations

To study the effect of proposal diversity, we compute solve rate as a function of the proposal budget k. For each k, the selector is restricted to a subset of k proposals from the cached $k = 8$ pool. The oracle best-of-k curve is computed using hidden-test outcomes for those k proposals. The implemented selector curves are computed using only cached binary-judge and comparator signals.

For subset-based ablations, results are averaged over proposal subsets of the same size when applicable. This avoids attributing performance changes to an arbitrary ordering of proposal indices.

C.8 Vote-Count Ablations

The binary-judge and comparator vote caches also allow offline vote-count ablations. For binary judges, we recompute the selector using subsets of the available judge votes and report the resulting solve rate. For comparators, we similarly recompute the Copeland tournament using subsets of the available pairwise votes. These ablations test how much selection accuracy depends on the number of local judgments rather than on additional proposal generation.

C.9 Failure Decomposition

For the category-level decomposition in Figure 4, each task is assigned to one of three mutually exclusive groups:

- **Solved by orchestration:** the patch selected by the deployed orchestration system passes the hidden tests.
- **Oracle reachable, missed:** at least one generated proposal passes the hidden tests, but the deployed selector does not select a passing patch.
- **Oracle unreachable:** none of the generated proposals passes the hidden tests.

This decomposition separates selection failures from proposal-coverage failures. An oracle-reachable-but-missed task is a selection failure: the correct patch was present in the proposal pool, but the selector failed to recover it. An oracle-unreachable task is a coverage failure: no selector could solve the task using the available proposal pool.

C.10 Reported Quantities

The main metric is hidden-test solve rate. For a deployed selector A, the solve rate is

$$\frac{1}{500}\sum_{x}\mathbf{1}\{A(x)\text{ passes the hidden tests}\}.$$

We also report the oracle best-of-k solve rate and the oracle-gap recovery ratio

$$\frac{p_{\text{system}}(k)-p_1}{p_{\text{oracle}}(k)-p_1},$$

where p_1 is the single-proposal baseline, $p_{\text{oracle}}(k)$ is the oracle best-of-k rate, and $p_{\text{system}}(k)$ is the implemented orchestration rate. This ratio measures how much of the latent best-of-k improvement is recovered by the deployed selector.

C.11 Prompts and Parsing

This subsection gives the selector prompts and parsing rules used in the offline SWE-bench Verified selector experiments. The proposer patches are generated once and cached; the prompts below are used only for selecting among the cached proposals.

Binary patch-judge prompt. The binary patch judge evaluates one patch at a time. The prompt asks whether the patch resolves the issue and preserves previously passing behavior.

Binary patch-judge prompt

```
You are evaluating a single patch against the GitHub issue it claims to fix.
Output yes if and only if running the patched code passes the failing test
described in the issue and does not break any test that previously passed.

Trace the failure path from the issue to the originating code. Check whether
the patch modifies that code path. Then state whether the patch produces the
expected behavior on the smallest input that exhibits the failure.

ISSUE:
{problem_statement}

PROPOSED PATCH:
{patch}

Respond with EXACTLY this JSON, no markdown fences, no extra text:
{
  "resolves": <true|false>,
  "reasoning": "<one sentence: smallest input under original code produces X; under patch produces Y
     >",
  "confidence": <integer 1-5>
}
```

The issue text is truncated to 2000 characters and the proposed patch to 8000 characters. Each patch receives multiple independent binary judgments. Malformed responses, refusals, or parse failures are treated as abstentions and do not contribute positive votes.

Binary-judge aggregation. For each patch, we compute the number of positive votes,

$$s_{\text{yes}}(p)=\#\{v:\ v.\text{resolves}=\text{true}\}.$$

The majority binary selector chooses the patch with the largest s_{yes}. Ties are broken by the lowest proposal index. We also compute a confidence-weighted score

$$s_{\text{conf}}(p)=\sum_{v:\ v.\text{resolves}=\text{true}} v.\text{confidence},$$

again breaking ties by the lowest proposal index. Unless otherwise stated, the main binary selector uses the yes-count rule.

Pairwise comparator prompt. The comparator evaluates two patches directly. It is instructed to prefer the patch that minimally resolves the issue, rather than the patch that is longer, more ambitious, or more visually complex.

Pairwise comparator prompt

```
You are deciding which of two patches is more likely to make the failing
tests pass without breaking the passing tests. That is the only question
you are answering. Aesthetics, ambition, and visible effort are not relevant.

Do NOT select a patch because it makes more changes, addresses more cases,
or appears more thorough. The patch that minimally resolves the stated
issue is preferred. A 4kb patch that fixes the bug beats an 11kb patch
that doesn't.

ISSUE:
{problem_statement}

PATCH A:
{patch_a}

PATCH B:
{patch_b}

REQUIRED STRUCTURAL COMPARISON (fill these in BEFORE deciding):

  1. Failing test hypothesis: state the smallest hypothesis about what is
     wrong, derived from the issue text alone. One sentence. This is the
     ground truth you compare both patches against.

  2. A_changes: list the specific lines/functions Patch A modifies.
  3. B_changes: list the specific lines/functions Patch B modifies.

  4. A_consistent_with_hypothesis: do A's changes plausibly cause the
     failing test in the issue to start passing? (true/false + one-line
     justification)
  5. B_consistent_with_hypothesis: same question for B.

  6. A_collateral: does A change behavior on inputs unrelated to the
     failure mode? (true/false + one-line justification)
  7. B_collateral: same question for B.

The decision falls out of this comparison; do not pull a winner from prior.
If exactly one patch is consistent with the hypothesis and the other is
not, that one wins. If both are consistent, prefer the one with less
collateral. If both fail the hypothesis, output TIE. If they are
functionally equivalent (same lines changed differently, or different
lines that produce the same behavior), output TIE.

Respond in this EXACT JSON format -- JSON only, no markdown fences, no extra text:
{
  "hypothesis": "<one sentence>",
  "a_changes": "<files/functions modified by A>",
  "b_changes": "<files/functions modified by B>",
  "a_consistent": <true|false>,
  "b_consistent": <true|false>,
  "a_collateral": <true|false>,
  "b_collateral": <true|false>,
  "winner": "A" | "B" | "TIE",
  "confidence": <integer 1-5>,
  "reasoning": "<one sentence: why the winner falls out of the structural comparison>"
}
```

As in the binary patch-judge prompt, the issue text is truncated to 2000 characters and each patch to 8000 characters. Responses are parsed as JSON. If the response is malformed after the allowed retries, the comparator judgment is treated as missing and handled as a tie in the offline aggregation.

C.12 Selection, Aggregation, and Tie-Breaking

Position-swap debiasing. For each unordered pair of patches (p_i, p_j), the comparator is queried in both orders: first with p_i as Patch A and p_j as Patch B, and then with the positions swapped. The swapped response is mapped back to the original patch indices before aggregation. This position-swap protocol reduces lead-position bias and prevents the selector from favoring a patch because it was shown first rather than because it better addresses the issue.

In the conservative offline rule, a pairwise comparison is counted as a win for p_i only if both position orders select p_i, and as a win for p_j only if both position orders select p_j. If the two orders disagree, or if either response is missing or tied, the aggregated pairwise outcome is treated as a tie.

Comparator aggregation. Pairwise comparator outcomes are aggregated with a Copeland-style tournament. For each unordered pair (p_i, p_j), the aggregated pairwise outcome is one of p_i wins, p_j wins, or tie. A win contributes one point to the winning patch; a tie contributes no win to either patch in the offline aggregation. The selected patch is the one with the largest Copeland score. Ties in the final Copeland score are broken deterministically by the lowest proposal index.

Hybrid gated comparator. The hybrid selector first applies the binary patch judge and retains patches whose yes-rate exceeds a threshold τ. If no patch passes the gate, the selector falls back to the highest-scoring patch under the binary judge. Pairwise comparator aggregation is then applied only to the retained patches. Unless otherwise specified, the main gated-comparator ablations use $\tau = 0.5$. Threshold-sweep experiments are treated as offline diagnostics.

Parse failures and abstentions. All selector prompts require strict JSON outputs. If a response cannot be parsed after the allowed retries, the corresponding judgment is treated as an abstention. For the binary patch judge, abstentions are not counted as positive votes. For pairwise comparison, missing or unparseable comparator outputs are treated as ties in the offline aggregation. This prevents malformed outputs from creating artificial wins.

Oracle diagnostics. The oracle best-of-k diagnostic is computed only after hidden-test evaluation. It counts a task as solved if any cached proposal passes the hidden tests. The deployed selector never observes these hidden-test outcomes. Therefore, at a fixed proposal budget k, the gap between oracle best-of-k and implemented orchestration measures missed selection opportunities: cases where a correct patch was present but not selected. The remaining gap between oracle best-of-k and perfect performance reflects finite-budget proposal-coverage failures, including possible shared blind spots of the proposer pool.

C.13 Selector Ablations

Table 1 gives additional selector ablations for the SWE-bench Verified experiment. These ablations use the same cached $k = 8$ `GPT-5.4 nano` proposal pool as the main experiment, so changes in solve rate reflect the selector rather than proposal generation.

The first block varies the comparator aggregation rule using `GPT-5.4 nano` comparators with five comparator calls per pairwise matchup. Copeland round-robin and strict dominance both reach 75.8%, while the single-elimination bracket drops to 74.6%. This supports the main-text claim that all-pairs aggregation preserves useful pairwise evidence better than cheaper tournament structures.

The second block varies the critic-gate threshold. With no critic gate, pure Copeland over the eight proposals reaches 75.8%. Dropping only patches with zero critic support raises performance to 76.4%, matching the best full-harness result. Increasing the threshold beyond $\tau \geq 2$ slightly reduces performance. Thus the critic stage is most useful as a permissive coarse filter, not as a strict correctness certificate.

D Proofs for Section 4

This section proves the bridge results. The central point is that proposer coverage and local identifiability are logically distinct. Coverage says that good actions can be sampled with nontrivial probability. Identifiability says that bad actions can be rejected or ranked below good ones. The latter cannot be obtained from black-box sampling alone.

D.1 Formal information model for Proposition 1

A local identification procedure is allowed to know the family construction, the action set $\mathcal{A}$, and the proposer distribution μ. It may observe any finite list of candidate actions and polynomially many i.i.d. samples from μ. It does not observe the hidden world parameter θ.

A critic edge is required to hold uniformly over worlds: for every θ, the critic must never reject actions in Sound_θ and must reject actions outside Sound_θ with probability bounded away from

Table 1: **Selector ablations on SWE-bench Verified.** All rows use the same $k = 8$ `GPT-5.4 nano` proposal pool. Tournament rows vary the `GPT-5.4 nano` comparator aggregation rule with 5 comparator calls per pairwise matchup. Threshold rows vary the five-critic gate with 5 critics and 5 comparators. Oracle-gap recovery is computed using the metric defined in Section 7.1.

Ablation	Setting	Resolve rate	Oracle-gap recovery
Tournament rule	Copeland round-robin	75.8%	73.3%
Tournament rule	Sequential king	75.6%	71.7%
Tournament rule	Strict dominance	75.8%	73.3%
Tournament rule	Single-elim bracket	74.6%	63.3%
Critic threshold	$\tau \geq 0$ no gate	75.8%	73.3%
Critic threshold	$\tau \geq 1$, drop 0-yes patches	**76.4%**	**78.3%**
Critic threshold	$\tau \geq 2$, tied with $\tau \geq 1$	**76.4%**	**78.3%**
Critic threshold	$\tau \geq 3$, starts losing 1 instance	76.2%	76.7%
Critic threshold	$\tau \geq 4$	76.2%	76.7%
Critic threshold	$\tau \geq 5$, unanimous gate	76.0%	75.0%

zero. A comparator edge is also required to hold uniformly: for every θ, whenever one candidate is in Sound_θ and the other is not, the comparator must prefer the sound candidate with probability at least $1/2 + \sigma$ for some $\sigma > 0$.

D.2 Proof of Proposition 1

Proposition 1 (Coverage does not imply local identifiability)**.** *For every $M \geq 2$, there exists a one-step task family with action set $\mathcal{A} = \{1, \ldots, M\}$ and hidden world parameter $\theta \in \{1, \ldots, M\}$ such that the proposer distribution is $\mathrm{Unif}(\mathcal{A})$ in every world, the progressing-sound set is $\mathrm{Sound}_\theta = \mathcal{A} \setminus \{\theta\}$, and no procedure observing only candidate actions and polynomially many samples from the proposer distribution has a uniform critic or comparator edge over all worlds.*

Proof. Fix $M \geq 2$. Let the action set be $\mathcal{A} = \{1, \ldots, M\}$ and let the hidden world parameter be $\theta \in \{1, \ldots, M\}$. In world θ, define

$$\mathrm{Sound}_\theta = \mathcal{A} \setminus \{\theta\}.$$

Let the proposer distribution be

$$\mu = \mathrm{Unif}(\mathcal{A})$$

in every world. Then, for every θ,

$$\mathbb{P}_{a \sim \mu}(a \in \mathrm{Sound}_\theta) = \frac{M-1}{M} = 1 - \frac{1}{M}.$$

Thus the proposer succeeds with probability $\alpha_0 = 1 - 1/M$ in every world.

The observable transcript of any procedure that sees only candidate actions and samples from μ is independent of θ, because the proposer distribution is identical in every world. We use this to rule out uniform critic and comparator edges.

First consider critics. Fix an action $i \in \mathcal{A}$. In world $\theta = i$, the action i is unsound. In any world $\theta = j \neq i$, the same action i is sound. Since the transcript distribution is identical in these two worlds, the critic has the same probability of rejecting i in both worlds. Uniform one-sided soundness requires that this rejection probability be zero in every world where i is sound. Hence the rejection probability for i must be zero in every world $\theta \neq i$. But the transcript distribution is the same in world $\theta = i$, so the rejection probability is also zero in the world where i is unsound. Therefore no critic can reject the unsound action with probability at least $\beta > 0$ uniformly over worlds.

Now consider comparators. Fix two distinct actions $i, j \in \mathcal{A}$. In world $\theta = i$, action i is unsound and action j is sound, so a comparator with edge $\sigma > 0$ must prefer j to i with probability at least $1/2 + \sigma$. In world $\theta = j$, action j is unsound and action i is sound, so the comparator must prefer i to j with probability at least $1/2 + \sigma$. But the observable transcript distribution is identical in these

two worlds. Therefore the probability of preferring i over j must be the same in both worlds. It cannot be both at least $1/2 + \sigma$ and at most $1/2 - \sigma$. Hence no uniform comparator edge exists.

Thus proposer coverage can be arbitrarily strong while local identifiability is impossible in this black-box information model. □

D.3 Proof of Theorem 1

Theorem 1 (Bridge theorem: coverage plus identifiability)**.** *Under Assumption 1 with proposer portfolio P_N and Assumption 2 with edges (β_0, σ_0). At every reachable valid nonterminal state s, round-robin assignment over the proposer portfolio with $k \geq |P_N| \lceil \ln(1/\delta_{\text{prop}})/\alpha_0 \rceil$ proposer calls gives*

$$\alpha_{\text{committee}}(s) \geq 1 - (1 - \alpha_0)^{\lfloor k/|P_N| \rfloor} \geq 1 - \delta_{\text{prop}}.$$

The critic and comparator edges remain $\beta(s) \geq \beta_0$ and $\sigma(s) \geq \sigma_0$.

Proof. Fix a reachable valid nonterminal state s. By Assumption 1, there exists a policy or prompt $p_s \in P_N$ such that $\mathbb{P}(\text{LLM}(p_s) \in \text{Sound}_x(s)) \geq \alpha_0$. Under round-robin assignment, if the committee makes k proposer calls, then p_s is used at least $q = \lfloor k/|P_N| \rfloor$ times. Since these calls use fresh independent randomness, the probability that none of them returns an action in $\text{Sound}_x(s)$ is at most $(1 - \alpha_0)^q$. Hence

$$\alpha_{\text{committee}}(s) \geq 1 - (1 - \alpha_0)^{\lfloor k/|P_N| \rfloor}.$$

If $k \geq |P_N| \left\lceil \alpha_0^{-1} \ln(1/\delta_{\text{prop}}) \right\rceil$, then $\lfloor k/|P_N| \rfloor \geq \alpha_0^{-1} \ln(1/\delta_{\text{prop}})$. Using $1 - u \leq e^{-u}$ for $u \in [0, 1]$,

$$(1 - \alpha_0)^{\lfloor k/|P_N| \rfloor} \leq \exp(-\alpha_0 \lfloor k/|P_N| \rfloor) \leq \delta_{\text{prop}}.$$

Therefore $\alpha_{\text{committee}}(s) \geq 1 - \delta_{\text{prop}}$.

The critic and comparator guarantees are not derived from Assumption 1; they are exactly the local-identifiability guarantees supplied by Assumption 2. Thus $\beta(s) \geq \beta_0$ and $\sigma(s) \geq \sigma_0$. This proves the theorem. □

D.4 Proof of Corollary 1

Corollary 1 (Verifier-backed bridge)**.** *Under Assumptions 1, 2, and 3 holds with $\beta_0 = 1 - \nu$ and $\sigma_0 = (1 - \nu)/2$.*

Proof. Use the verifier $V_x(s, a)$ as the critic. If $a \in \text{Sound}_x(s)$, Assumption 3 gives $V_x(s, a) = \text{ACCEPT}$ almost surely, so the critic never rejects a progressing-sound action. If $a \notin \text{Sound}_x(s)$, Assumption 3 gives $\mathbb{P}(V_x(s, a) = \text{REJECT}) \geq 1 - \nu$. Hence the critic edge is $\beta_0 = 1 - \nu$.

For comparison, suppose $a \in \text{Sound}_x(s)$ and $b \notin \text{Sound}_x(s)$. Verify both candidates. The sound candidate a is never rejected, while b is rejected with probability at least $1 - \nu$. If b is rejected, choose a; if both candidates are accepted, break the tie uniformly at random. Therefore the probability of choosing the sound candidate is at least

$$(1 - \nu) + \frac{\nu}{2} = \frac{1}{2} + \frac{1 - \nu}{2}.$$

Thus the comparator edge is $\sigma_0 = (1 - \nu)/2$. This proves the corollary. □

D.5 Why disagreement-based identification is insufficient

One might try to identify bad actions by comparing them to a sampled reference action. This is not valid without a much stronger canonical-witness assumption. In many verifier-backed domains there are many distinct progressing-sound actions at the same state. Two proof steps may differ syntactically while both preserve solvability and make progress. Two program continuations may use different implementations while both satisfy the local specification. Therefore token agreement, syntactic proximity, or agreement with one sampled reference is not a reliable proxy for soundness.

A theorem based on sampled-reference agreement would require an additional assumption, such as the existence of a canonical progressing-sound witness at each state and a known representation in which soundness is equivalent, or nearly equivalent, to agreement with that witness. That is much stronger than Assumption 1. Without such an assumption, Proposition 1 rules out deriving local identification from coverage alone.

E Proofs for Section 5

This section proves the cumulative and local error bounds. We explicitly include the local-failure/no-survivor case in the bad event.

E.1 Proof of Lemma 1

Lemma 1 (Adaptive cumulative-error bound)**.** *Let x have rank bound L_x. Suppose that along the protocol trajectory, $\mathbb{P}(A_t \notin \mathrm{Sound}_x(S_t) \mid \mathcal{F}_t) \le \varepsilon_t$ whenever $S_t \in \mathrm{Valid}_x \setminus \mathrm{Term}_x$, with local failure counted as an invalid action. Then $\mathrm{err}_x(k, m, r) \le \sum_{t=0}^{L_x-1} \mathbb{E}[\varepsilon_t]$. In particular, if $\varepsilon_t \le \varepsilon$ for all t, then $\mathrm{err}_x(k, m, r) \le L_x\varepsilon$.*

Proof. For each $t < L_x$, let B_t be the event that, when $S_t \in \mathrm{Valid}_x \setminus \mathrm{Term}_x$, the protocol either declares local failure or selects an action $A_t \notin \mathrm{Sound}_x(S_t)$. If $S_t \notin \mathrm{Valid}_x \setminus \mathrm{Term}_x$, set $B_t = \emptyset$. Equivalently, local failure may be represented as selecting a formal invalid absorbing action. By assumption, $\mathbb{P}(B_t \mid \mathcal{F}_t) \le \varepsilon_t$.

Let $G = \bigcap_{t=0}^{L_x-1} B_t^c$. On G, whenever the protocol is at a valid nonterminal state, it selects an action in $\mathrm{Sound}_x(S_t)$. By Definition 2, the next state remains valid and the rank strictly decreases:

$$S_{t+1} = \Phi_x(S_t, A_t) \in \mathrm{Valid}_x, \qquad d_x(S_{t+1}) < d_x(S_t).$$

Since d_x is a nonnegative integer with rank bound L_x, the protocol reaches a terminal state within at most L_x progressing steps. By Definition 1, every terminal state $s \in \mathrm{Term}_x$ satisfies $R_x(\mathrm{Out}_x(s)) = 1$. Therefore,

$$\{R_x(\Pi_{k,m,r}(x)) = 0\} \subseteq \bigcup_{t=0}^{L_x-1} B_t.$$

The union bound and tower property give

$$\mathbb{P}(R_x(\Pi_{k,m,r}(x)) = 0) \le \sum_{t=0}^{L_x-1} \mathbb{P}(B_t) = \sum_{t=0}^{L_x-1} \mathbb{E}[\mathbb{P}(B_t \mid \mathcal{F}_t)] \le \mathbb{E}\left[\sum_{t=0}^{L_x-1} \varepsilon_t\right].$$

If $\varepsilon_t \le \varepsilon$ almost surely for every t, this becomes

$$\mathbb{P}(R_x(\Pi_{k,m,r}(x)) = 0) \le L_x\varepsilon.$$

This proves the lemma. □

E.2 Proof of Theorem 2

Theorem 2 (Local error decomposition)**.** *At any reachable valid nonterminal state s, under the local role model with critic edge β and comparator edge σ, and under the conditional independence assumptions stated in Appendix E,*

$$\varepsilon_{\mathrm{loc}}(s) \le \varepsilon_{\mathrm{prop}}(k; s) + k^2(1-\beta)^m e^{-2r\sigma^2} \le \varepsilon_{\mathrm{prop}}(k; s) + k^2 e^{-\beta m - 2r\sigma^2}.$$

If the proposer calls are conditionally independent and each succeeds with probability at least α, then $\varepsilon_{\mathrm{prop}}(k; s) \le (1-\alpha)^k \le e^{-\alpha k}$.

Proof. Fix a reachable valid nonterminal state s, and let $C_1, \ldots, C_k$ be the proposed candidates. Define

$$E_{\mathrm{prop}} := \{C_i \notin \mathrm{Sound}_x(s) \text{ for all } i = 1, \ldots, k\}.$$

By definition, $\mathbb{P}(E_{\text{prop}}) = \varepsilon_{\text{prop}}(k; s)$. If E_{prop} occurs, then no progressing-sound candidate is available, so the protocol may either select an unsound action or declare local failure; both are catastrophic local errors and are charged to $\varepsilon_{\text{prop}}(k; s)$.

Now condition on E^c_{prop}. At least one progressing-sound candidate has been proposed. Since critics are one-sided, no progressing-sound candidate is ever rejected, so at least one progressing-sound candidate survives criticism and the no-survivor case cannot occur.

Fix an ordered pair (a, b) of proposed candidates with $a \in \text{Sound}_x(s)$ and $b \notin \text{Sound}_x(s)$. The m critics assigned to b reject it independently with probability at least β, so

$$\mathbb{P}(b \text{ survives all } m \text{ critics}) \leq (1 - \beta)^m.$$

Conditioned on b surviving, each of the r fresh comparator votes prefers a to b with probability at least $1/2 + \sigma$, independently across votes. Let $Y_j \in \{0, 1\}$ indicate whether vote j prefers a to b. Since $\mathbb{E}[Y_j] \geq 1/2 + \sigma$, the unsound candidate b wins the majority comparison only if $\frac{1}{r}\sum_{j=1}^{r} Y_j \leq 1/2$, where ties may be broken adversarially against a. By Hoeffding's inequality,

$$\mathbb{P}\left(\frac{1}{r}\sum_{j=1}^{r} Y_j \leq \frac{1}{2}\right) \leq e^{-2r\sigma^2}.$$

Since the comparison randomness is fresh relative to the critic randomness, for this fixed ordered pair,

$$\mathbb{P}(b \text{ survives criticism and defeats } a) \leq (1 - \beta)^m e^{-2r\sigma^2}.$$

There are at most k^2 ordered sound/unsound pairs, so by a union bound the probability that some unsound candidate survives criticism and defeats some sound candidate is at most

$$k^2(1 - \beta)^m e^{-2r\sigma^2}.$$

Call this event E_{id}.

If neither E_{prop} nor E_{id} occurs, then at least one sound candidate survives, and every surviving unsound candidate loses its pairwise comparison to every surviving sound candidate. We claim that every Copeland winner is then sound. Let $S \geq 1$ be the number of surviving sound candidates and $U \geq 0$ the number of surviving unsound candidates. Every sound candidate beats all U unsound candidates, so its Copeland score is at least U. Every unsound candidate loses to all S sound candidates, so it can only score wins against other unsound candidates and has Copeland score at most $U - 1$. Thus no unsound candidate can be a Copeland winner.

Therefore catastrophic local error can occur only if E_{prop} or E_{id} occurs. Hence

$$\varepsilon_{\text{loc}}(s) \leq \varepsilon_{\text{prop}}(k; s) + k^2(1 - \beta)^m e^{-2r\sigma^2}.$$

Using $1 - \beta \leq e^{-\beta}$,

$$\varepsilon_{\text{loc}}(s) \leq \varepsilon_{\text{prop}}(k; s) + k^2 e^{-\beta m - 2r\sigma^2}.$$

Finally, if the k proposer calls are conditionally independent and each succeeds with probability at least α, then

$$\varepsilon_{\text{prop}}(k; s) \leq (1 - \alpha)^k \leq e^{-\alpha k}.$$

This proves the theorem. □

F Proofs and Extensions for Section 6

This appendix contains the derivations behind the blind-spot discussion in Section 6. The main text states the exchangeable local version because it is the clearest way to see the message: proposal failure equals a blind-spot floor plus a finite-sampling residual. We record the proof, the heterogeneous-portfolio extension, finite-k rates, and the terminal-output analogue used by oracle best-of-k benchmarks.

F.1 Proof of Lemma 2

Lemma 2 (Local oracle miss and blind-spot floor)**.** *Fix a reachable valid nonterminal state s. Suppose that, conditional on a latent variable Z, the k proposal calls are independent and each lies in $\mathrm{Sound}_x(s)$ with probability $q_s(Z)$. Then*

$$\varepsilon_{\mathrm{prop}}(k; s) = \mathbb{E}[(1 - q_s(Z))^k].$$

Equivalently,

$$\varepsilon_{\mathrm{prop}}(k; s) = B_s + R_k(s),$$

where

$$B_s := \mathbb{P}(q_s(Z) = 0), \qquad R_k(s) := \mathbb{E}[(1 - q_s(Z))^k \mathbf{1}\{q_s(Z) > 0\}].$$

In particular, $\varepsilon_{\mathrm{prop}}(k; s) \to B_s$ as $k \to \infty$.

Proof. Fix a reachable valid nonterminal state s. Conditional on Z, the k proposal-success indicators are independent Bernoulli variables with common success probability $q_s(Z)$. Therefore the conditional probability that all k proposals fail is

$$\mathbb{P}(\text{all proposals fail} \mid Z) = (1 - q_s(Z))^k.$$

Taking expectation over Z gives

$$\varepsilon_{\mathrm{prop}}(k; s) = \mathbb{E}[(1 - q_s(Z))^k].$$

Split the expectation according to whether $q_s(Z) = 0$:

$$\mathbb{E}[(1 - q_s(Z))^k] = \mathbb{P}(q_s(Z) = 0) + \mathbb{E}[(1 - q_s(Z))^k \mathbf{1}\{q_s(Z) > 0\}].$$

This is exactly $B_s + R_k(s)$. Since $(1 - q_s(Z))^k \mathbf{1}\{q_s(Z) > 0\} \to 0$ pointwise and is bounded by 1, dominated convergence gives $R_k(s) \to 0$. Hence $\varepsilon_{\mathrm{prop}}(k; s) \to B_s$. □

F.2 Heterogeneous proposer portfolios

The exchangeable formula in the main text can be replaced by a heterogeneous portfolio calculation. This version is useful when prompts, tools, retrieval contexts, decoding policies, or base models define different proposer families.

Proposition 2 (Heterogeneous portfolio oracle miss)**.** *Fix a reachable valid nonterminal state s. Suppose the proposal system contains families $g = 1, \ldots, G$, with n_g calls from family g. Conditional on a latent variable Z, calls are independent and each call from family g proposes an action in $\mathrm{Sound}_x(s)$ with probability $q_g(Z)$. Then*

$$\varepsilon_{\mathrm{prop}}(\{n_g\}; s) = \mathbb{E}\left[\prod_{g=1}^{G} (1 - q_g(Z))^{n_g}\right].$$

If $Q(Z) := \sum_{g=1}^{G} n_g q_g(Z)$, then

$$\varepsilon_{\mathrm{prop}}(\{n_g\}; s) \le \mathbb{E}[e^{-Q(Z)}].$$

Consequently, if $Q(Z) \ge \lambda$ except on an event of probability at most η, then

$$\varepsilon_{\mathrm{prop}}(\{n_g\}; s) \le \eta + e^{-\lambda}.$$

If $n_g \to \infty$ for every family g, then

$$\varepsilon_{\mathrm{prop}}(\{n_g\}; s) \to \mathbb{P}(\forall g,\ q_g(Z) = 0).$$

Proof. Conditional on Z, all calls are independent. The probability that all calls from family g fail is $(1 - q_g(Z))^{n_g}$, and multiplying over families gives the product formula. The upper bound follows from $1 - u \le e^{-u}$:

$$\prod_{g=1}^{G} (1 - q_g(Z))^{n_g} \le \exp\left(-\sum_{g=1}^{G} n_g q_g(Z)\right) = e^{-Q(Z)}.$$

If $Q(Z) \ge \lambda$ on an event A with $\mathbb{P}(A) \ge 1 - \eta$, then the integrand is at most $e^{-\lambda}$ on A and at most 1 on A^c, giving $\eta + e^{-\lambda}$. Finally, for fixed Z, the product converges to 1 exactly when all $q_g(Z) = 0$, and to 0 otherwise. Dominated convergence proves the limiting floor. □

This proposition is the formal version of the diversity claim in the main text. Adding more calls from the same family only helps where that family has positive conditional proposal mass. Diversifying the portfolio can reduce the blind-spot floor only when different families cover different latent subpopulations.

F.3 Finite-sampling residual and lower-tail rates

The main text writes the local oracle miss as

$$\varepsilon_{\mathrm{prop}}(k; s) = B_s + R_k(s), \qquad R_k(s) = \mathbb{E}[(1 - q_s(Z))^k \mathbf{1}\{q_s(Z) > 0\}].$$

The rate at which $R_k(s)$ decays is determined by the lower tail of $q_s(Z)$ near zero.

Proposition 3 (Lower-tail bound)**.** *Suppose that for some $a, C > 0$ and all sufficiently small $t > 0$,*

$$\mathbb{P}(0 < q_s(Z) \le t) \le Ct^a.$$

Then for all sufficiently large k,

$$R_k(s) \le C \left(\frac{a \log k}{k} \right)^a + k^{-a}.$$

In particular, if $q_s(Z) \ge \tau > 0$ whenever $q_s(Z) > 0$, then

$$R_k(s) \le e^{-\tau k}.$$

Proof. For the uniform lower bound, simply use $(1 - q_s(Z))^k \le e^{-\tau k}$ on $\{q_s(Z) > 0\}$. For the lower-tail bound, set $t_k = a \log(k)/k$. Split

$$R_k(s) = \mathbb{E}[(1 - q_s(Z))^k \mathbf{1}\{0 < q_s(Z) \le t_k\}] + \mathbb{E}[(1 - q_s(Z))^k \mathbf{1}\{q_s(Z) > t_k\}].$$

The first term is at most $\mathbb{P}(0 < q_s(Z) \le t_k) \le Ct_k^a$. For the second term, $(1 - q)^k \le e^{-kq} \le e^{-kt_k} = k^{-a}$. Substituting $t_k = a \log(k)/k$ proves the claim. □

Thus oracle boosting is complete up to blind spots, but the finite-k cost depends on the lower tail of proposal mass. Uniform positive mass gives logarithmic sample complexity in $1/\varepsilon$, polynomial lower tails give polynomial convergence, and exponentially small proposal mass gives exponentially expensive brute-force boosting.

F.4 Task-level oracle curves

The theorem in the main text is local: $q_s(Z)$ is the probability of proposing a progressing-sound next action at state s. Benchmarking often uses the terminal-output analogue. For a proposer harness P that samples complete outputs $Y \sim P(\cdot \mid x)$, define

$$p_{\mathrm{oracle}}(k; P) := \mathbb{P}(\exists i \le k : \ R_x(Y_i) = 1).$$

If $q_P(Z) := \mathbb{P}(R_x(Y) = 1 \mid Z)$, the same calculation gives

$$p_{\mathrm{oracle}}(k; P) = 1 - \mathbb{E}[(1 - q_P(Z))^k], \qquad \lim_{k \to \infty} p_{\mathrm{oracle}}(k; P) = 1 - \mathbb{P}(q_P(Z) = 0).$$

This is the terminal-output version of the blind-spot floor. It is the quantity estimated by oracle best-of-k curves in verifier-backed benchmarks. If an implemented system is evaluated on the same candidate pool, the oracle-gap recovery ratio compares its success to this oracle curve.

F.5 Common-shock special case

A useful closed-form dependence model is the following. With probability ρ, all proposers share a common fate. In this branch, the whole committee fails with probability $1 - \alpha$. With probability $1 - \rho$, the proposers act independently with marginal success probability α, so the whole committee fails only if all k proposers fail, with probability $(1 - \alpha)^k$. Therefore

$$\varepsilon_{\mathrm{prop}}(k) = \rho(1 - \alpha) + (1 - \rho)(1 - \alpha)^k.$$

Equivalently, the proposer success probability is

$$\alpha_{\mathrm{committee}} = 1 - \rho(1 - \alpha) - (1 - \rho)(1 - \alpha)^k.$$

As $k \to \infty$, the failure probability converges to $\rho(1 - \alpha)$, the common-shock floor.

G Extended Literature Review

This appendix expands the main related-work discussion. We organize prior work by *mechanism* rather than by chronology. The unifying question is how existing methods instantiate, implicitly or explicitly, the four ingredients in our framework:

$$\text{coverage}, \qquad \text{identifiability}, \qquad \text{progress}, \qquad \text{diversity}.$$

The literature already contains many empirical demonstrations that extra inference-time compute can improve LLM performance: sampling more candidates, voting, ranking, searching, debating, using tools, or adding agents. Our goal is not to claim novelty for any of these practices. Rather, the framework in this paper gives a role-wise account of one mechanism behind the gains and its limits: proposer coverage creates an oracle gap, local identifiability recovers part of that gap, progress composes local decisions into trajectories, and diversity controls the ceiling through shared blind spots.

G.1 Classical boosting, weak learning, and ensemble methods

Classical boosting is the historical source of the weak-to-strong amplification analogy. The PAC weak-learning theorem of [1] shows that weak and strong learnability are equivalent under the classical PAC formalism. [2] and [3] developed algorithms for combining weak hypotheses, with AdaBoost becoming the canonical practical method. Bagging, stacking, and ensemble methods more broadly provide related mechanisms for variance reduction and model combination [53–55].

The analogy to LLM agent systems is useful but partial. Classical boosting assumes labeled examples and a weak learner that returns hypotheses in an evaluable label space. In our setting, there may be no label for an intermediate state unless the task exposes a local verifier. Moreover, the system must repeatedly generate and select local moves along a trajectory. Thus the classical weak edge is split into several quantities: local proposal coverage, local identification edge, progress depth, and dependence/diversity. This is the first way in which verifier-backed committee search differs from ordinary supervised boosting.

G.2 LLMs as weak learners, weak supervision, and boosting-style LLM methods

Several papers explicitly use weak-learning, weak-supervision, or boosting language for language models. [16] show that prompt-based language models can act as weak learners inside a classical boosting pipeline for tabular classification. In their setting, the LLM produces textual summaries or templates that serve as weak classifiers, and boosting is applied in a supervised learning loop. This is close in terminology but different in object: our system performs black-box inference-time search rather than training-time tabular boosting.

LLMBoost [18] is another close terminological neighbor. It proposes an ensemble fine-tuning framework that uses cross-model attention, chain training, error-suppression objectives, and near-parallel inference. This is a representation-sharing and training/fine-tuning approach. By contrast, our framework treats the generator, critic, and comparator as black-box inference-time components. We therefore do not require internal hidden states, cross-model attention, or parameter updates.

Weak-to-strong generalization is also conceptually adjacent. [17] study whether weak model supervision can elicit strong capabilities from a stronger model. Iterated amplification [46] similarly studies how weak experts can supervise stronger learners through recursive decomposition. These works share our interest in using weak signals to elicit stronger behavior, but the object is different: they study supervision and training protocols, whereas we study fixed-weight inference-time committees operating over verifier-backed search trajectories.

Recent self-improvement and sharpening-style works are also related. The sharpening mechanism studies how a model can use verifier-like feedback to concentrate probability mass on high-quality outputs and amortize expensive inference-time computation [56]. These works are aligned with our interest in eliciting latent capability, but our focus is a theorem for committee search and its ceilings: coverage, local identifiability, progress, and shared blind spots.

G.3 Multi-sample inference: self-consistency, best-of-N, and repeated sampling

A large line of work improves LLM reasoning by sampling multiple outputs. Chain-of-thought prompting elicits intermediate reasoning traces [57]; self-consistency then samples multiple reasoning paths and marginalizes over answers rather than using greedy decoding [4]. This is one of the clearest empirical antecedents of our coverage story. The method works when at least some sampled traces reach the correct answer and the majority/consistency heuristic can identify the right answer.

Rationale-augmented ensembles sample rationales and aggregate predictions, showing that rationale diversity can improve few-shot robustness [19]. Universal Self-Consistency extends the aggregation idea to settings where answers are not trivially comparable by exact match, using an LLM to select the most consistent answer among free-form candidates [20]. Best-of-N, pass@k, and oracle-selection evaluations in code and reasoning instantiate the same principle: repeated sampling reveals latent correct mass.

Large Language Monkeys [6] is especially close to our benchmarking section. It studies repeated sampling as inference-time compute scaling and measures *coverage*, the fraction of problems solved by at least one sample. In verifier-backed domains such as coding and formal proofs, this coverage can directly translate into performance if correct samples can be identified. The paper also finds that in domains without automatic verifiers, majority voting and reward models may plateau, leaving a generation-verification gap. Our theory formalizes this distinction: oracle best-of-k estimates latent boostable capability, while practical selectors recover only part of the oracle gap.

G.4 More agents, compound inference systems, and voting laws

More Agents Is All You Need [5] studies a simple Agent Forest method: instantiate many LLM agents, sample outputs, and aggregate by voting. The paper reports that performance scales with the number of agents and that the degree of improvement correlates with task difficulty. This is one of the clearest empirical motivations for our theory. In our language, adding agents helps when it increases coverage or diversity; it saturates when additional agents are shared-blind-spot clones.

[7] study scaling laws for compound inference systems, focusing on Vote and Filter-Vote architectures. They show that increasing the number of calls can yield non-monotone performance because easy and hard instances respond differently to majority voting. This complements our analysis: their theory focuses on flat voting systems and majority aggregation, while ours studies sequential verifier-backed search with explicit identifiability and progress conditions.

Recent surveys on test-time scaling, LLM ensembles, and verifier design organize this rapidly expanding landscape [21, 58, 59]. These surveys emphasize that additional inference compute can be allocated in many ways: parallel sampling, sequential refinement, search, verification, aggregation, routing, fusion, or distillation. Our paper contributes a mechanism-level theorem for one important slice of that landscape.

G.5 Verifiers, reward models, process supervision, and judges

Verifier-based selection is a central precedent for our identifiability assumption. [10] train verifiers for math word problems and show that generating many candidate solutions and selecting with a verifier improves GSM8K performance. This is the flat-output analogue of our proposer-plus-critic decomposition.

Process supervision moves verification from final answers to intermediate steps. [22] compare outcome and process supervision and release PRM800K, demonstrating that process-supervised reward models can improve mathematical reasoning. [23] construct automatic process supervision for math reasoning and use Math-Shepherd both for reranking and reinforcement. These works are strongly aligned with our local-identifiability story: step-level verifiers instantiate local signals that can reject or score partial reasoning moves.

LLM-as-a-judge and pairwise ranking methods provide another route to identifiability. LLM-Blender [24] uses a PairRanker to compare candidate outputs and a GenFuser to synthesize improved outputs. Multi-Agent Verification [25] scales test-time compute by adding multiple aspect verifiers and combining them with best-of-n sampling. Weaver [26] studies the generation-verification gap and combines many weak verifiers using weak supervision, showing that verifier aggregation can

recover much of the gap between pass@1 and oracle selection. Very recent verifier-ensemble work such as FUSE [60] pushes this idea further by ensembling verifiers without ground-truth labels.

Reward-model and judge benchmarks show that selection models have their own biases and failure modes. MT-Bench and Chatbot Arena use LLM-as-a-judge as a scalable approximation to human preferences [27]; RewardBench evaluates reward models for preference modeling and RLHF-style selection [28]; recent surveys map the broader LLM-as-a-judge landscape [61]. These works reinforce a central point of our paper: selection is not a cosmetic add-on. It is a separate resource. Our bridge theorem and black-box impossibility result formalize this separation: local proposal coverage does not imply a usable critic or comparator edge.

G.6 Tool-backed checking, self-correction, and verification loops

Several works improve LLM outputs by adding external feedback, tools, or iterative verification. CRITIC [29] uses tool-interactive critiquing to validate and revise outputs. Chain-of-Verification [30] asks models to draft answers, generate verification questions, answer them independently, and then revise. Self-Refine [36] iteratively generates feedback and refines outputs without additional training data. Reflexion [35] stores verbal feedback in memory to improve subsequent trials.

Tool-using systems also change the effective verification and proposal process. Toolformer [62] trains models to decide when and how to call external APIs; HuggingGPT [63] uses an LLM as a controller over a collection of external models; ReAct-style systems interleave reasoning and environment actions. In our terminology, tools can improve local identifiability, increase proposal coverage, reduce blind spots, or change the decomposition so that progress becomes easier to certify.

These systems instantiate repeated local improvement, but their reliability depends on whether feedback exposes true errors and whether revisions make progress. Our framework makes those requirements explicit. A feedback loop is useful only insofar as it increases local identifiability, improves proposal coverage on later attempts, changes the state decomposition, or reduces the cost of reaching the same boostable ceiling.

G.7 Search over partial states and planning-style inference

Search-based prompting and agent methods are natural neighbors of our valid-state framework. Least-to-most prompting decomposes hard problems into easier subproblems solved sequentially [31]. Tree of Thoughts generalizes chain-of-thought into a search over intermediate thoughts, using evaluation and backtracking [9]. Self-evaluation-guided beam search uses stepwise self-evaluation to guide stochastic beam search through reasoning chains [64]. RAP uses the LLM as both world model and reasoning agent, and applies Monte Carlo tree search [32]. LATS integrates reasoning, acting, planning, self-reflection, and MCTS-style search [33]. ReAct interleaves reasoning traces with external actions [34].

These works already embody the idea that reasoning is not merely one-shot generation. Our paper contributes a set of sufficient and necessary conditions under which such search-like systems can amplify weak local reasoning. In particular, decomposition alone is not enough. The decomposition must expose progressing-sound actions that the proposer can sample and that local signals can identify.

G.8 General multi-agent orchestration, routing, and model aggregation

A broad multi-agent literature studies how LLM agents should communicate, specialize, coordinate, route tasks, and use tools. CAMEL introduces role-playing communicative agents [37]; AutoGen provides a programmable multi-agent conversation framework [38]; MetaGPT encodes standardized operating procedures for collaborative software development [39]; ChatDev studies communicative agents for software development workflows [65]; AgentVerse studies multi-agent collaboration and emergent behavior [66]. Mixture-of-Agents constructs layered multi-model aggregation where each layer conditions on outputs from the previous layer [40]. Archon searches over inference-time architectures combining generation, ranking, fusion, verification, and other inference-time techniques under compute budgets [41]. Routing methods such as Smoothie study how to choose among LLMs without labeled data or with weak supervision [42].

These works explore design spaces rather than proving a universal theory. Our framework offers a diagnostic lens for them. A role or subagent is valuable if it improves one of the load-bearing quantities: coverage, identifiability, progress, or diversity per unit cost. Additional agents are not inherently useful; they help if they reduce blind spots, provide independent evidence, improve selection, expose new decompositions, or approach the same boostable ceiling with lower budget.

G.9 Software-engineering agents and verifier-backed code benchmarks

Software engineering is one of the best testbeds for our theory because candidate artifacts can often be checked by execution, tests, type checkers, or repository-specific validators. Codex [44] and AlphaCode [43] already showed the importance of sampling, filtering, and program behavior for code generation; AlphaCode in particular generated many candidate programs, filtered by behavior, and clustered candidates before submission. SWE-bench [12] introduced realistic GitHub issue resolution tasks that require repository navigation, code editing, and execution-based validation. SWE-agent [13] shows that agent-computer interface design can substantially improve software-engineering agents. AutoCodeRover [14] combines LLMs with code search and program analysis. Agentless [15] argues that simpler localization-repair-validation pipelines can achieve strong SWE-bench Lite performance with lower cost than more complex autonomous agents.

These systems support one of our practical implications: better agent systems are not necessarily more elaborate systems. They are systems that move the coverage-identifiability-progress-diversity frontier efficiently. A simple pipeline with strong localization and validation can beat a more ornate agent if it recovers the oracle gap with less cost.

G.10 Inference-time alignment, reward-model limitations, and verifier aggregation

Inference-time alignment studies how to use additional compute and imperfect reward models to improve responses without updating the base model. [8] analyze best-of-N alignment and show that coverage over high-quality responses is crucial; they also show that naively increasing N can suffer from reward hacking under imperfect reward models. This is conceptually close to our separation between oracle best-of-k and implemented selection. In our terminology, a large oracle gap indicates latent correct mass, but a weak selector may fail to recover it or may select reward-hacked candidates.

Verifier aggregation methods such as Weaver and FUSE [26, 60] are especially close to our identification story. They treat imperfect verifiers as weak signals and combine them to approach stronger verification behavior. Our theory abstracts this into critic/comparator edges and highlights the missing robustness question: repeated verifiers can also share blind spots, just as repeated proposers can.

Reward-model and judge benchmarks such as RewardBench and LLM-as-a-judge evaluations further show that selection models have their own biases and failure modes [27, 28, 61]. This reinforces the need to treat identifiability as a separate assumption rather than deriving it from generation.

G.11 Benchmarking implications

The literature increasingly evaluates models and systems using pass@1, best-of-N, pass@k, oracle selection, verifier selection, and budgeted success. Our framework suggests a unified reporting scheme for verifier-backed benchmarks:

$$p_1(P), \qquad p_{\mathrm{oracle}}(K; P), \qquad p_{\mathrm{system}}(K, m, r; P), \qquad \mathrm{Rec}(K, m, r; P),$$

together with token/call/latency budgets. This separates one-shot capability, boostable capability, selector quality, and efficiency. Prior work often reports one or two of these quantities; our theory explains why all four matter. In particular, oracle best-of-k measures latent proposal mass, residual oracle failure estimates blind-spot mass, implemented system performance measures harness quality, and budgeted curves measure how efficiently the system approaches the boostable ceiling.