

CONSTRAINT TREE EXPLORATION FOR LEARNING FROM LANGUAGE FEEDBACK

Shaoang Li¹ Daniel R. Jiang² Jian Li¹

¹Stony Brook University

²Meta Platforms

{shaoang.li, jian.li.3}@stonybrook.edu

danielrjiang@gmail.com

ABSTRACT

Natural-language feedback in interactive learning often explains *why* an action failed by pointing to violated requirements. Misinterpreting this feedback can lead an agent to rule out valid solutions. We study this setting by modeling user intent as latent constraints over an action space and formulating learning from language feedback as pure exploration over feasible regions. We introduce TRACE, an algorithm that organizes candidate constraints in a tree and tests each proposed refinement by generating actions that satisfy it. TRACE commits to the refinement only if the resulting feedback does not contradict it over repeated tests. We distinguish two ways of using the same feedback: (i) *falsification*, which detects contradictions to the constraint set currently being tested, and (ii) *identification*, which may additionally name a violated constraint. We prove high-probability coverage bounds with dependence on the candidate class size H for TRACE-FALSIFICATION. With reliable identification, TRACE-IDENTIFICATION can replace this dependence by K/p_{ext} , where K is the number of latent constraints and p_{ext} lower-bounds the probability of extracting a missing true constraint from informative feedback. We evaluate TRACE across six language-feedback tasks. On RecMovie, TRACE-IDENTIFICATION achieves 73% and 86% final-output success under caps of 20 and 60 evaluated outputs, compared with at most 42% and 48% for the evaluated prompting baselines given the same feedback and output caps. Controlled identity-corruption experiments further show greater robustness than direct accumulation when the falsification detector remains reliable.

1 INTRODUCTION

Interactive agents built on large language models (LLMs) can receive feedback expressed in natural language rather than as a numerical reward (Cheng et al., 2023; Shinn et al., 2023). A user may reject a movie recommendation by saying “I wanted something from the 90s”, revise a travel plan by saying “this exceeds my budget”, or correct a generated poem by saying “the second line has too many syllables”. Such feedback can explain *why* an action failed by identifying an unmet requirement. Because this requirement also applies to other candidate actions, the feedback can help narrow the search for acceptable alternatives. The challenge is to infer useful constraints from incomplete or ambiguous feedback and determine when the available evidence justifies enforcing them.

Existing approaches have explored how natural-language feedback can be incorporated into LLM agents. Feedback-driven methods, including REFLEXION (Shinn et al., 2023), SELF-REFINE (Madaan et al., 2023), TEXTGRAD (Yuksekgonul et al., 2025), and language-agent tree search (LATS) (Zhou et al., 2024), use feedback to guide generation and search through revised prompts, reflective memories, textual gradients, or value estimates. Recent theoretical work provides complementary abstractions, including regret bounds under hindsight instructions (Misra et al., 2024) and consistency-based elimination over global hypotheses (Xu et al., 2025). We focus on pure exploration over feasible regions: the learner discovers, tests, and accumulates latent constraints to progressively restrict the action space. This formulation makes explicit both the information needed to propose a constraint and the evidence needed to commit to it.

We consider episodes with fixed latent requirements represented in a finite candidate class. We distinguish two oracle-access regimes induced by the same natural-language feedback. In the

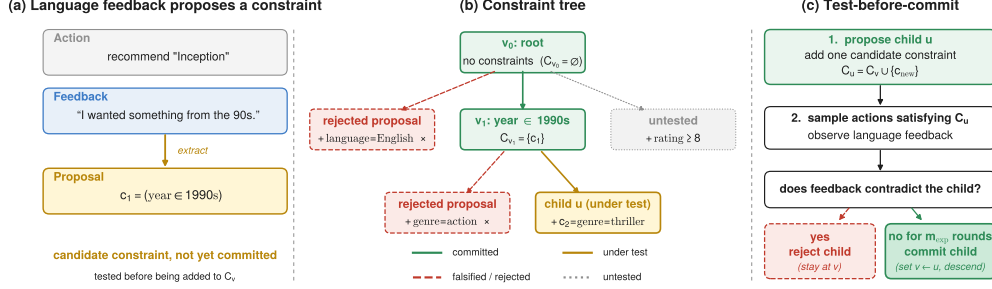


Figure 1: Constraint-tree exploration in TRACE. At a committed node v with constraint set C_v , a candidate constraint c_{new} proposes a child u with $C_u = C_v \cup \{c_{\text{new}}\}$. The child is tested by sampling actions satisfying C_u : feedback contradicting C_u rejects the child, while survival for a fixed number m_{exp} of tests leads to commitment. Extracted identities guide proposals; falsification tests determine commitment.

falsification setting, the learner has access to a violation oracle that tests whether feedback contradicts the constraint set currently being tested. In the *identification setting*, the learner additionally has access to an extractor that may name a violated atomic constraint. These two regimes specify what information the learner can obtain from the same raw feedback. Identification access can accelerate exploration by revealing which constraint to test next. But an extracted identity is only a proposal, not a fact: committing an incorrectly extracted constraint can permanently shrink the feasible region in the wrong direction. Without identities, however, discovering useful constraints may require blind search over a large candidate class. The challenge is therefore to benefit from named constraints without trusting them too early. This leads to the central question of the paper:

How can an agent exploit named constraints from language feedback without committing to them prematurely, and when does identification access reduce the sample complexity of exploration?

We answer this question with the *Test-and-Refine Algorithm for Constraint Exploration* (TRACE). The search state in TRACE is the committed constraint set, which indexes a surviving feasible region. Candidate refinements form children in a constraint tree, as illustrated in Figure 1. Rather than committing to a proposed constraint immediately, TRACE performs a multi-round test: it samples actions from the child region and rejects the child if feedback contradicts the tested constraint set. If the child survives repeated tests, the algorithm commits to it and descends the tree. Thus, extracted identities are treated as *proposals*, not facts. The tree separates proposal from commitment and organizes partially accumulated constraints. The same test-before-commit primitive instantiates the two oracle-access regimes formalized in Section 2.2: TRACE-FALSIFICATION (TRACE-F; Assumption 1 only), which uses the violation oracle for falsification, and TRACE-IDENTIFICATION (TRACE-I; Assumptions 1 and 2), which additionally prioritizes constraints extracted from feedback.

Our analysis separates the cost of blind falsification from the value of usable identities. In the falsification setting, TRACE-F discovers true constraints by proposing candidates from a finite class of size H , leading to sample complexity $\tilde{O}(H)$ up to the per-test cost of falsification (Theorem 2). In the identification setting, under sound extraction and an explicit discovery-progress condition, TRACE-I replaces this dependence by the effective discovery cost $\tilde{O}(K/p_{\text{ext}})$, where K is the number of latent constraints and p_{ext} lower-bounds the probability of extracting a missing true constraint from informative feedback (Theorem 3). These bounds quantify how usable identities can reduce the number of tested refinements relative to uniform blind search.

Empirically, we evaluate TRACE across five controlled language-feedback environments spanning six task instances: recommendation, two entity-deduction benchmarks, MASTERMIND, and constrained poem generation. On RecMovie, TRACE-I achieves 73% and 86% final-output success under common caps of 20 and 60 evaluated outputs, compared with at most 42% and 48% for the evaluated prompting baselines. Controlled MASTERMIND sweeps show that identification-guided proposals require fewer tested refinements as the candidate class grows. False-identity experiments further show that TRACE-I remains more robust than direct accumulation when extracted identities are corrupted and the falsification detector remains reliable.

In summary, we make three contributions. *First*, we formulate learning from language feedback as pure exploration over latent constraints, where feedback provides local evidence about feasible regions rather than scalar rewards. *Second*, we introduce TRACE, a parser-agnostic test-before-commit framework that treats extracted identities as proposals rather than facts. *Third*, we derive high-probability coverage bounds that quantify the benefit of identification access, and we evaluate proposal efficiency, interaction budgets, and robustness to identity corruption.

2 PROBLEM FORMULATION AND CONSTRAINT TREE

We formulate learning from language feedback as a pure-exploration problem over latent constraints. The learner never observes numerical utility values; instead, it receives natural-language feedback that may reveal violated requirements. We then introduce the constraint tree, which organizes partially accumulated constraints as feasible regions to be tested and refined.

2.1 LANGUAGE-FEEDBACK PURE EXPLORATION

Action space and latent utility. Let $\mathcal{X}$ be a finite action space and let $U : \mathcal{X} \rightarrow [0, 1]$ be an unknown latent utility function. The interaction proceeds for T rounds. In each round $t \in [T]$, an algorithm $\mathcal{A}$ selects an action $x_t \in \mathcal{X}$ based on the past observations $(x_1, f_1, \dots, x_{t-1}, f_{t-1})$, and receives natural-language feedback $f_t \in \mathcal{F}$. The algorithm never observes $U(x_t)$ directly.

Language feedback as constraint evidence. For each action $x \in \mathcal{X}$, feedback is drawn from a conditional distribution P_x over $\mathcal{F}$. Conditioned on x_t , the feedback f_t is independent of the past. We study the regime in which feedback may identify requirements that the action violates, provide qualitative corrections, or confirm that the action is close to satisfying the user’s intent. For example, a user may say “the recommendation is not from the 90s” or “the answer is good, but it misses the required attribute”. We treat such feedback as a *constraint signal*: progress comes from discovering and testing latent requirements, rather than from comparing observed numerical rewards.

Canonical constraint class. Let $\mathcal{H}$ be a finite class of canonical candidate constraints, with $H := |\mathcal{H}|$, within which equality and set membership are well-defined. Each constraint $c \in \mathcal{H}$ defines a region $\text{Region}(c) \subseteq \mathcal{X}$, and we write $\text{Sat}(x, c) := \mathbf{1}\{x \in \text{Region}(c)\}$ for the satisfaction indicator. For a constraint set $C \subseteq \mathcal{H}$, define $\text{Region}(C) := \bigcap_{c \in C} \text{Region}(c)$, with $\text{Region}(\emptyset) := \mathcal{X}$. The latent constraints below are assumed to lie in $\mathcal{H}$, i.e., the algorithm searches over a class that contains them.

Latent constraints. The user’s hard requirements are encoded by an unknown subset $C^* := \{c_1^*, \dots, c_K^*\} \subseteq \mathcal{H}$ of $K := |C^*|$ latent constraints, with feasible region $\text{Region}(C^*) = \bigcap_{k=1}^K \text{Region}(c_k^*)$. We assume that these constraints are necessary for optimality, i.e., every globally optimal action satisfies all of C^* : $\mathcal{X}^* := \arg \max_{x \in \mathcal{X}} U(x) \neq \emptyset$ and $\mathcal{X}^* \subseteq \text{Region}(C^*)$. Let $x^* \in \mathcal{X}^*$ and $U^* := U(x^*)$. Apart from this compatibility condition, the utility function U is arbitrary: actions outside $\text{Region}(C^*)$ may still have positive utility, and U may take any values in $[0, 1]$ rather than being restricted to $\{0, 1\}$. Thus, the constraints define the feasible region containing all global optima, while the utility function determines how costly it is to violate or miss them.

Coverage objective and regret consequence. After T rounds, the algorithm outputs a single action $\hat{x}$. The primary objective in this paper is *coverage*: reach a nonempty region contained in $\text{Region}(C^*)$ and sample the output from that region. The utility function is not observed during interaction; it is used only to express the consequence of coverage for simple regret,

$$\mathbb{E}[r_T] := \mathbb{E}[U^* - U(\hat{x})], \quad (1)$$

where the expectation is over the algorithm’s randomness and the feedback channel. As formalized below, if the algorithm reaches a region contained in $\text{Region}(C^*)$, then the remaining regret is controlled by the variation of U within the latent feasible region.

2.2 CONSTRAINT TREE REPRESENTATION

The constraint tree organizes partial information about the latent feasible region. Each node corresponds to a set of constraints accumulated so far and therefore to the region of actions satisfying them. Moving deeper in the tree adds constraints and shrinks the feasible region.

Nodes as constraint sets. A constraint tree $\mathcal{T}$ is a rooted tree of proposal paths. Each node v is associated with the accumulated constraint set $C_v \subseteq \mathcal{H}$ and feasible region

$$\mathcal{X}_v := \mathcal{X}(C_v) := \{x \in \mathcal{X} : \text{Sat}(x, c) = 1, \forall c \in C_v\} = \text{Region}(C_v). \quad (2)$$

The root v_0 has $C_{v_0} = \emptyset$ and $\mathcal{X}_{v_0} = \mathcal{X}$. Each child u of v adds one new constraint, so $C_u = C_v \cup \{c\}$ for some $c \in \mathcal{H} \setminus C_v$. The depth of v is $h(v) := |C_v|$. Different proposal orders may induce the same accumulated set and feasible region; we keep them as distinct tree nodes because they correspond to different search histories.

Viable and spurious nodes. For any node v with a nonempty feasible region, define

$$U(v) := \max_{x \in \mathcal{X}_v} U(x), \quad \Delta(v) := U^* - U(v). \quad (3)$$

We call v *viable* if $\Delta(v) = 0$, i.e., $\mathcal{X}_v$ contains a globally optimal action, and *spurious* if $\Delta(v) > 0$. The condition $C_v \subseteq C^*$ is sufficient for viability, but not necessary: a redundant or stronger-looking constraint outside C^* may still preserve at least one global optimum. The algorithm uses feedback to reject spurious branches while preserving viable ones.

Within-region spread. Even a viable region may contain actions of different utility. We define the within-node spread $\text{Spread}(v) := U(v) - \inf_{x \in \mathcal{X}_v} U(x)$, and, for the latent feasible region,

$$\text{Spread}(C^*) := \sup_{x \in \text{Region}(C^*)} U(x) - \inf_{x \in \text{Region}(C^*)} U(x). \quad (4)$$

If $\mathcal{X}_v \subseteq \text{Region}(C^*)$, then any $\hat{x} \in \mathcal{X}_v$ satisfies all latent constraints and has regret at most $\text{Spread}(C^*)$. Thus, the theory focuses on reaching a covered region; $\text{Spread}(C^*)$ captures the residual difficulty of choosing among feasible actions.

Oracle access and the two settings. We distinguish the raw natural-language feedback $f_t \sim P_x$ from the oracle information that the learner can reliably extract from it. The raw feedback is the same in both regimes; the regimes differ only in the oracle interfaces available to the learner.

Assumption 1 (Falsification access). *The learner has access to a violation detector $\text{Viol} : \mathcal{F} \times 2^{\mathcal{H}} \rightarrow \{0, 1\}$ such that $\text{Viol}(f; C) = 1$ only if f contradicts some constraint already in C . In particular, feedback that reveals a missing constraint outside C is discovery feedback, not a falsification of C .*

Assumption 2 (Identification access). *In addition to Assumption 1, the learner has an extractor $\text{Extract} : \mathcal{F} \rightarrow \mathcal{H} \cup \{\emptyset\}$ that returns a canonical violated constraint when one can be reliably identified, and returns $\emptyset$ otherwise.*

For example, consider a tested child with $C_u = \{\text{year}=1980\text{s}\}$. Feedback “I wanted something from the 90s” can both refute C_u and propose $\text{year}=1990\text{s}$. By contrast, feedback “I wanted a comedy” can reveal a missing genre constraint without contradicting C_u .

The *falsification setting* assumes only Assumption 1, equivalently using a null extractor. The *identification setting* assumes both Assumptions 1 and 2. The corresponding algorithmic variants are TRACE-F and TRACE-I. Quantitative reliability and soundness conditions, together with the oracle-level noise model used in our analysis, are stated in Appendix B.1.

Algorithmic interfaces. Beyond these oracles, TRACE uses a constrained generator Q_C for sampling actions satisfying a constraint set C , and a proposer Propose for selecting candidate constraints to test. These modules may be implemented by template decoders, LLM-based modules, database filters, constrained decoders, or domain-specific heuristics.

3 TEST-AND-REFINE ALGORITHM FOR CONSTRAINT EXPLORATION

We now describe the *Test-and-Refine Algorithm for Constraint Exploration* (TRACE). The goal is to reach a constraint-tree node whose feasible region is contained in the user’s latent feasible region. TRACE maintains a committed node v , proposes candidate constraints to refine C_v , and uses language feedback to decide whether each proposed refinement should be rejected or committed. The central design principle is that extracted identities serve as *proposals* rather than facts: they bias which child is tested next, but enter the committed set only after passing a multi-round falsification test.

Algorithm 1 Test-and-Refine Algorithm for Constraint Exploration (TRACE).

Require: horizon T , test length m_{exp} , max tests N_{max} , generators $\{Q_C\}$, detector Viol, proposer Propose, extractor Extract

- 1: Initialize $v \leftarrow v_0$ with $C_{v_0} = \emptyset$; $W \leftarrow \emptyset$; $R_{v'} \leftarrow \emptyset$ for every node v' ; $t \leftarrow 0$; $n \leftarrow 0$
- 2: **while** $t < T$ **and** $n < N_{\text{max}}$ **do**
- 3: $c_{\text{prop}} \leftarrow \text{Propose}(v, W, R_v)$; **break if** $c_{\text{prop}} = \emptyset$
- 4: $u \leftarrow \text{child of } v \text{ with } C_u = C_v \cup \{c_{\text{prop}}\}$
- 5: **if** $\mathcal{X}_u = \emptyset$ **then** $R_v \leftarrow R_v \cup \{c_{\text{prop}}\}$; **continue** *empty region: exclude at v, no budget*
- 6: $n \leftarrow n + 1$; $\text{pass} \leftarrow \text{true}$
- 7: **for** $s = 1$ to m_{exp} **do**
- 8: **if** $t \geq T$ **then** $\text{pass} \leftarrow \text{false}$; **break**
- 9: $t \leftarrow t + 1$; sample $x_t \sim Q_{C_u}$, receive f_t
- 10: $c_{\text{ext}} \leftarrow \text{Extract}(f_t)$; **if** $c_{\text{ext}} \neq \emptyset$ **then** $W \leftarrow W \cup \{c_{\text{ext}}\}$ *store extracted identity*
- 11: **if** $\text{Viol}(f_t; C_u) = 1$ **then** $\text{pass} \leftarrow \text{false}$; **break**
- 12: **if** $\text{pass} = \text{true}$ **then** $v \leftarrow u$ **else** $R_v \leftarrow R_v \cup \{c_{\text{prop}}\}$ *commit, or exclude at v*
- 13: **return** $\hat{x} \sim Q_{C_v}$

Feedback use and commitment. During a test of a proposed child u , feedback is used in two ways. The detector $\text{Viol}(f_t; C_u)$ decides whether u is falsified. When identification access is available, $\text{Extract}(f_t)$ may add a candidate constraint to a global working set W for future proposals. Crucially, entries in W are never committed directly: they only affect which child is tested next.

Test-before-commit. Given a proposed constraint c_{prop} , TRACE forms the child u with $C_u = C_v \cup \{c_{\text{prop}}\}$ and samples from Q_{C_u} for up to m_{exp} rounds. If any feedback contradicts C_u , the child is rejected and the algorithm remains at v . A per-node set R_v records the candidates ruled out at v (empty region or rejected child), so they are not proposed again at v ; this ensures termination and never excludes a missing true constraint while v is viable (Remark 1). If the child survives all m_{exp} tests, TRACE commits to u and moves deeper in the tree. At the end of the budget, it returns $\hat{x} \sim Q_{C_v}$, where v is the deepest committed node at termination. If $\text{Region}(C_v) \subseteq \text{Region}(C^*)$, then $\hat{x}$ satisfies all latent requirements; the remaining simple regret is controlled by $\text{Spread}(C^*)$.

Variants. Both variants propose from $\mathcal{H}_v := \mathcal{H} \setminus (C_v \cup R_v)$, and Propose returns $\emptyset$ when $\mathcal{H}_v = \emptyset$. TRACE-F uses a null extractor and proposes uniformly from $\mathcal{H}_v$. TRACE-I first proposes from $W \cap \mathcal{H}_v$ when nonempty, and otherwise falls back to uniform proposal over $\mathcal{H}_v$. Both variants use the same falsification test before commitment; operational properties of W , the detector contract, and the role of the counters t, s in Algorithm 1 are recorded in Appendix B.2.

Implementation and theory bridge. TRACE is parser- and generator-agnostic: in finite recommendation or entity-deduction tasks, Q_C can be a filtered catalog sampler, while in constrained generation it can be an LLM prompted with accumulated requirements or a constrained decoder. The detector and extractor can likewise be template-based, rule-based, LLM-based, or hybrid. Our experiments use lightweight decoders where possible, while Section 4 abstracts away these implementation choices and analyzes how many child tests are needed to reach a covered region.

4 THEORETICAL ANALYSIS

We analyze the two oracle-access regimes introduced in Section 2.2. The analysis abstracts away implementation details of the generator, detector, extractor, and proposer, and asks how many interaction rounds suffice for TRACE to reach a covered region. TRACE-F (Assumption 1 only) uses the violation oracle for falsification and searches over the candidate class by blind proposal. TRACE-I (Assumptions 1 and 2) additionally uses extracted constraint identities to bias proposals. Both analyses share the same multi-sample falsification argument; they differ in how many child tests are needed before the latent feasible region is covered. Full proofs are deferred to Appendix B.

4.1 ANALYSIS MODEL AND GUARANTEE

We work with the finite canonical constraint class $\mathcal{H} \supseteq C^*$ from Section 2.1. Throughout, $H := |\mathcal{H}|$ is the candidate class size and $K := |C^*|$ is the number of latent constraints; the bounds depend on these together with the extraction probability p_{ext} in the identification setting. The constrained

generator Q_C is supported on $\mathcal{X}(C) := \{x \in \mathcal{X} : \text{Sat}(x, c) = 1 \ \forall c \in C\}$ and returns independent samples on each call. The identification-setting theorem (Assumption 2) additionally requires sound extracted identities, i.e., Extract returns either a latent constraint or $\emptyset$. Thus the theorem captures missed identity discoveries; spurious extracted identities are treated empirically in Section 5.4. The implementation may run Extract on every feedback message; the theorem only requires that informative feedback events, i.e., violations observed while testing a spurious child, add missing true constraints to W with probability at least p_{ext} .

Conditions for the guarantees. The formal assumptions are stated in Appendix B.1. Here we summarize their roles.

▷ *Refutational feedback*: For a node v , let $q_v := \Pr[\text{Viol}(f; C_v) = 1 \mid x \sim Q_{C_v}, f \sim P_x]$ be the probability that one test sample falsifies the region. Viable regions are not incorrectly rejected by the contradiction detector: if $\Delta(v) = 0$, then $q_v = 0$. Conversely, spurious regions are falsifiable: if $\Delta(v) > 0$, then $q_v \geq \min\{1, c_{\text{fal}}\Delta(v)^\beta\}$ for constants $c_{\text{fal}}, \beta > 0$. The exponent β captures the resolution of the violation oracle (Assumption 1); larger β means small gaps are harder to detect.

▷ *ε -separation (gap structure)*: At the target tolerance ε , every child of a viable node either preserves a global optimum, $\Delta(u) = 0$, or excludes all optima by margin $\Delta(u) > \varepsilon$. This is a structural condition on $(\mathcal{X}, U, \mathcal{H}, C^*)$, not on the generator, detector, or extractor; it ensures that a finite test window can separate viable children from spurious ones.

▷ *Identification-access progress*: Theorem 3 additionally assumes two conditions. The first is structural (Assumption 8): every constraint in $\mathcal{H} \setminus C^*$ with a nonempty region excludes all global optima with gap larger than ε even when imposed alone, so adding it to any viable node yields an empty region or such a gap. In particular, the candidate class contains no *zero-gap distractors*, i.e., constraints outside C^* that preserve a global optimum when added to a viable node. The second concerns extraction (Assumption 3): on informative feedback, the extractor of Assumption 2 reveals a missing latent constraint with probability at least p_{ext} . Together, the two conditions force every incorrect candidate to be incompatible with the true requirements (Appendix B.1). Reliable template decoders can support the extraction condition, but not the structural one, which depends only on $(\mathcal{X}, U, \mathcal{H}, C^*)$. When either condition fails, the $\tilde{O}(K/p_{\text{ext}})$ guarantee of Theorem 3 does not apply.

Coverage guarantee. We say that an algorithm enjoys an (ε, δ) -coverage guarantee with budget T if, with probability at least $1 - \delta$, the deepest committed node v at termination satisfies

$$\text{Region}(C_v) \subseteq \text{Region}(C^*) \quad (5)$$

within T interaction rounds. This target is representation-invariant: exact syntactic recovery of C^* is sufficient, but not necessary. If $\hat{x} \sim Q_{C_v}$, coverage implies $\hat{x} \in \text{Region}(C^*)$ and therefore

$$\mathbb{E}[r_T] \leq \text{Spread}(C^*) + \delta. \quad (6)$$

The two theorems below establish coverage under the falsification and identification settings.

4.2 FALSIFICATION- AND IDENTIFICATION-SETTING GUARANTEES

Multi-sample falsification. The core observation is that a spurious child is unlikely to survive many independent feedback tests. We state the lemma for an arbitrary tested node; in Algorithm 1, this node is typically the proposed child.

Lemma 1 (Multi-sample falsification). *Let v be any node with $\Delta(v) > \varepsilon$. If the algorithm samples $x_s \sim Q_{C_v}$ for m independent rounds and observes feedback $f_s \sim P_{x_s}$, then*

$$\Pr[\text{Viol}(f_s; C_v) = 0 \text{ for all } s = 1, \dots, m] \leq \exp(-m c_{\text{fal}} \varepsilon^\beta). \quad (7)$$

Consequently, if $m \geq \log(1/\delta')/(c_{\text{fal}} \varepsilon^\beta)$, then a node with gap larger than ε survives m tests with probability at most δ' .

Falsification-setting guarantee. We first consider TRACE-F, which uses only the violation oracle of Assumption 1 and proposes uniformly from the finite candidate class.

Theorem 2 (Coverage in the falsification setting). *Suppose the conditions of Section 4.1 hold for refutational feedback and ε -separation. Fix $\delta \in (0, 1)$ and $\varepsilon \in (0, 1]$. With*

$$N_{\max} = \left\lceil H \log \frac{2 \max(K, 1)}{\delta} \right\rceil, \quad m_{\text{exp}} = \left\lceil \frac{\log(2N_{\max}/\delta)}{c_{\text{fal}} \varepsilon^\beta} \right\rceil, \quad T = N_{\max} m_{\text{exp}}, \quad (8)$$

TRACE-FALSIFICATION enjoys an (ε, δ) -coverage guarantee with $T = \tilde{O}(H/(c_{\text{fal}} \varepsilon^\beta))$.

The bound has two main factors. The *proposal cost* is $\tilde{O}(H)$: in the canonical finite-class proof, a sufficient route to coverage is to propose all latent constraints, which uniform proposal achieves in $\mathcal{O}(H \log K)$ calls. This cost is over proposal calls from $\mathcal{H}$, not over all ordered root-to-leaf paths in the constraint tree. The *testing cost* is $\tilde{O}(1/(c_{\text{fal}}\varepsilon^\beta))$: by Lemma 1, this many samples reject any child with gap larger than ε with the desired confidence. Viable children survive because $q_u = 0$ when $\Delta(u) = 0$, while spurious children with $\Delta(u) > \varepsilon$ are refuted with probability at least $c_{\text{fal}}\varepsilon^\beta$ per test. Once $\text{Region}(C_v) \subseteq \text{Region}(C^*)$, the returned action lies in the latent feasible region, giving $\mathbb{E}[r_T] \leq \text{Spread}(C^*) + \delta$.

Identification-setting guarantee. Theorem 2 shows the falsification setting alone suffices for coverage, but requires blind proposal over the ambient candidate class. In a favorable identification regime where extracted identities make reliable progress toward missing latent constraints, we show that this candidate-class dependence can be replaced by the effective discovery cost K/p_{ext} .

Theorem 3 (Coverage in the identification setting). *Suppose the conditions of Section 4.1 hold, including identification-access progress. Fix $\delta \in (0, 1)$ and $\varepsilon \in (0, 1]$. With*

$$N_{\text{max}}^{\text{id}} = \left\lceil \frac{64}{p_{\text{ext}}} \left(K + \log \frac{4}{\delta} \right) \right\rceil, \quad m_{\text{exp}}^{\text{id}} = \left\lceil \frac{\log(2N_{\text{max}}^{\text{id}}/\delta)}{c_{\text{fal}}\varepsilon^\beta} \right\rceil, \quad T = N_{\text{max}}^{\text{id}} m_{\text{exp}}^{\text{id}},$$

TRACE-IDENTIFICATION enjoys an (ε, δ) -coverage guarantee with $T = \tilde{O}(K/(p_{\text{ext}} c_{\text{fal}}\varepsilon^\beta))$.

Theorem 3 replaces the blind-search cost H by the effective discovery cost K/p_{ext} . The intuition is that, under identification access, diagnostic feedback can add a missing true constraint to W with probability p_{ext} ; once in W , that constraint is proposed before blind candidates and survives the same test window. Thus identification access is most useful when H is large relative to K/p_{ext} , and its advantage diminishes when the candidate class is small or extraction is unreliable. Section 5.3 validates this regime distinction. Section 5.4 tests the case where the soundness clause of Assumption 2 is not satisfied (the extractor returns spurious constraints).

5 EXPERIMENTS

We evaluate the practical benefits of TRACE’s constraint-based formulation and test-before-commit mechanism through four questions: **(Q1)** How does TRACE compare with prompting methods under different feedback interfaces and common output caps? **(Q2)** Does identification feedback reduce the number of candidate refinements tested as the candidate class grows? **(Q3)** When does test-before-commit improve over directly accumulating extracted constraints? **(Q4)** How robust is TRACE to incorrect extracted identities?

Setup. We evaluate on five environments spanning six task instances: LLF-Bench RecMovie (Cheng et al., 2023), a synthetic 20-Questions environment (20Q), Apple’s Entity-Deduction Arena (EDA THINGS) (Zhang et al., 2024), per-position MASTERMIND (Li & Zhu, 2024), and LLF-Bench Poem (Cheng et al., 2023) with Haiku and Tanka forms. The primary metric is *env.success* over 100 seeds per cell; for RecMovie this refers to the returned recommendation list (an interaction-budget any-success diagnostic is in Appendix C.3). We compare TRACE-F, which uses binary falsification with uniform proposals, with TRACE-I, which additionally prioritizes constraints extracted from past feedback. Prompt-based baselines RANDOM, DIRECTLLM, REFLEXION (Shinn et al., 2023), and SELF-REFINE (Madaan et al., 2023) use Qwen3-4B (Yang et al., 2025) under fixed per-environment interaction budgets ($T = 20$ on RecMovie) that are not matched to TRACE’s default configuration (see Section 5.2 for equal budgets); structured same-interface baselines use the same decoded feedback as TRACE but remove the test-before-commit step. Throughout this section, brackets in result tables denote Wilson 95% confidence intervals; hyperparameters, Viol/Extract implementation, full-table intervals, attempts, and soft-utility diagnostics are deferred to Appendix C.

5.1 MAIN COMPARISON: EXPLICIT CONSTRAINT STRUCTURE HELPS

Table 1 reports *env.success* on three task instances under three oracle-access regimes: *Natural* feedback, where prompt-based methods consume the environment’s raw text; *Falsification* access, corresponding to Assumption 1; and *Identification* access, corresponding to Assumptions 1+2. Within each regime, all methods share the same oracle interface, so cross-method comparisons within

Table 1: Main comparison under three oracle-access regimes. Random does not consume feedback. Wilson 95% intervals in Tables 7 and 10; MASTERMIND/Haiku/Tanka in Appendix C.2.

Method	RecMovie			20Q			EDA		
	Natural	Falsif.	Identif.	Natural	Falsif.	Identif.	Natural	Falsif.	Identif.
Random	5%	–	–	41%	–	–	74%	–	–
DirectLLM	17%	27%	40%	12%	43%	100%	43%	84%	100%
Reflexion	26%	35%	42%	11%	29%	100%	55%	69%	100%
Self-Refine	21%	26%	32%	73%	19%	100%	78%	40%	100%
TRACE	–	51%	90%	–	84%	100%	–	87%	99%

a regime are matched. TRACE-F and TRACE-I appear under the Falsification and Identification columns, respectively; they are the test-before-commit instantiations of the falsification and identification settings, not competing variants.

At each method’s default budget, within each oracle regime, TRACE improves over the strongest same-regime prompt baseline: TRACE-F by 16pp on RecMovie (51% vs. 35%), 41pp on 20Q, and 3pp on EDA; TRACE-I by 48pp on RecMovie (90% vs. 42%), and by at least 31pp when all methods share an equal output cap (Section 5.2). On 20Q and EDA, identity-feedback methods reach 99–100%, leaving little headroom for additional gains in these configurations. Self-Refine’s success on 20Q drops from 73% under natural feedback to 19% under binary projection, illustrating the importance of the information retained in the feedback interface. Three additional task instances (MASTERMIND with $K = 3$, Haiku, and Tanka) are deferred to Appendix C.2, where the qualitative pattern is identical: TRACE-I reaches 99–100% while natural-feedback baselines remain below 30%. The candidate-class scaling on MASTERMIND is studied in Section 5.3.

5.2 EQUAL OUTPUT BUDGETS AND INTERACTION COST

Table 1 uses method-specific default budgets. We therefore compare TRACE-I with prompting and sampling controls on RecMovie, which provides a non-saturated comparison under identification feedback. Figure 2a evaluates common caps of $B \in \{20, 60\}$ environment-evaluated outputs, scoring one returned output per seed: TRACE-I ($m_{\text{exp}} = 5$) uses at most $B - 1$ outputs for tests and returns one within the cap, while each iterative baseline returns its first accepted output or its last output at the cap. All baselines receive the same identity projection as TRACE-I; to run them with 60 outputs, DirectLLM-W8 and Self-Refine-W8 keep only the eight most recent interactions, and Reflexion-NoCycle disables early trial termination on repeated recommendations. TRACE-I reaches 73% at $B = 20$ and 86% at $B = 60$, versus at most 42% and 48% for the baselines evaluated at each cap; Best-of- N with an oracle selector reaches 17% and 21%.

Each test round shows a complete recommendation rather than asking the user to reconfirm a constraint, and smaller test windows trade reliability for fewer rounds (Figure 2b): with $m_{\text{exp}} = 1$, TRACE-I reaches 65% with 4.43 feedback rounds on average. Under a tight cap a larger window uses up more of the budget (78% for $m_{\text{exp}} = 3$ vs. 73% for $m_{\text{exp}} = 5$ at $B = 20$), whereas at $B = 60$ it reaches 86%. Appendix C.3 gives the full protocol, intervals, and costs, and shows that seeding the root with up to two oracle-provided constraints raises success to 90% at $B = 20$.

5.3 IDENTITY FEEDBACK REDUCES BLIND SEARCH

Theorem 3 predicts that identity-aware proposal can replace blind dependence on the ambient candidate class size H by the effective discovery cost K/p_{ext} . We test this in a controlled MASTERMIND environment with candidate class $\mathcal{H} = \{(i, c) : i \in [K], c \in \text{palette}\}$ and $|\mathcal{H}| = K \cdot n_{\text{colors}}$. Table 2 fixes $K = 3$ (with $N_{\text{max}} = 80$ enlarged from the headline $N_{\text{max}} = 15$ to give TRACE-F budget headroom at large $|\mathcal{H}|$) and varies $n_{\text{colors}} \in \{3, 6, 10, 15, 20\}$, thereby increasing H while keeping the number of latent constraints fixed; *attempts* is the number of proposed children actually tested. Under the uniform binary proposer, success drops from 100% at $|\mathcal{H}| = 9$ to 74% at $|\mathcal{H}| = 60$, and mean attempts grow from 9.08 to 50.82. In contrast, the decoded identity proposer remains at 99–100% success across the same sweep and uses substantially fewer attempts (3.91 to 18.08). These results support the proposal-efficiency advantage of identification access as the candidate class grows.

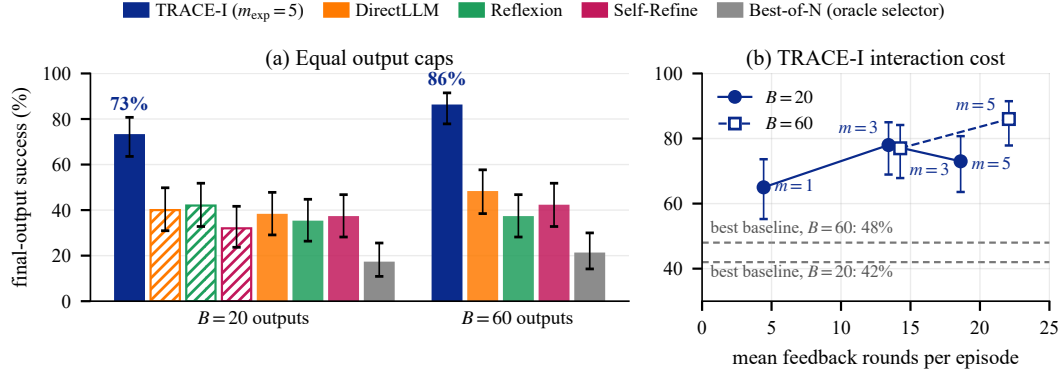


Figure 2: RecMovie under equal output budgets (100 seeds; final-output success with Wilson 95% intervals). (a) Common caps of $B = 20$ and $B = 60$ environment-evaluated outputs. Hatched bars are the Table 1 baseline configurations, which were run only with $T = 20$; solid baseline bars are the W8/NoCycle variants run with 60 outputs. (b) TRACE-I success versus mean feedback rounds for test windows $m = m_{\text{exp}} \in \{1, 3, 5\}$; dashed lines mark the strongest baseline at each cap.

Table 2: H -sweep on MASTERMIND ($K = 3$, $m_{\text{exp}} = 15$, $N_{\text{max}} = 80$). Wilson 95% intervals in Table 8.

n_{colors}	$ \mathcal{H} $	FALSIFICATION		IDENTIFICATION	
		env.succ.	attempts	env.succ.	attempts
3	9	100%	9.08	100%	3.91
6	18	100%	18.75	100%	4.84
10	30	98%	29.35	100%	6.77
15	45	88%	45.38	100%	10.54
20	60	74%	50.82	99%	18.08

5.4 WHY TEST-BEFORE-COMMIT MATTERS

We examine when test-before-commit improves performance by comparing TRACE-I with direct accumulation using the same decoder, followed by controlled corruption of extracted identities.

Same-decoder direct accumulation. Table 3 compares TRACE-I with Direct-Commit baselines that immediately accumulate constraints from the same decoded feedback. Under the reported default configurations, TRACE-I reaches 90% success on RecMovie, compared with 23% for the strongest direct-commit variant. On 20Q and EDA THINGS, where identity hints are essentially clean, both approaches reach 99–100%. These saturated cases leave little headroom for further gains from test-before-commit. We next examine the effect of incorrect identities through controlled corruption.

False extracted identities. In Table 4, we inject false identities into RecMovie by replacing each decoded feedback predicate with a wrong same-dimension predicate with probability p_{false} . TRACE-I uses corrupted identities only as proposals, whereas direct commit immediately accumulates them. As corruption increases, both methods degrade, but TRACE-I remains more robust: at $p_{\text{false}} = 0.5$, it still reaches 60% env.success, while the best direct-commit variant reaches only 8%. These results support testing extracted constraints before commitment when identity errors are possible.

Table 3: Direct-Commit baselines: best of $\{\text{AND, OR, RANKED}\}$ per environment; variants in Appendix C.3.

Method	Rec.	20Q	EDA
TRACE-I	90%	100%	99%
Direct-Commit	23%	100%	100%

Table 4: False-identity robustness on RecMovie ($p_{\text{false}} = 0$ row matches Table 1).

p_{false}	TRACE-I	Direct-Commit
0.0	90% [83,94]	23% [16,32]
0.2	74% [65,82]	16% [10,24]
0.5	60% [50,69]	8% [4,15]

Table 5: TRACE-I on TRAVELPLANNER, $n=50$. *Identification*: committed $C_v \supseteq C^*$. *Final Pass*: plan passes every env check.

LLM	Identification	Final Pass
Qwen3-4B	84% [72, 92]	16% [8, 29]
DeepSeek-V4-Flash	86% [74, 93]	90% [79, 96]

Additional diagnostics. Component ablations on RecMovie distinguish proposal quality from detector reliability. With the rule-based detector fixed, an LLM-based atomic proposer increases interaction-budget any-success from 57% under uniform proposals to 74%. Replacing the rule-based detector with an LLM detector instead reduces TRACE-F’s any-success from 57% to 21%. Appendix C reports these ablations, soft-utility results, direct-commit variants, feedback-omission robustness, and a complementary MASTERMIND K -sweep; Appendix C.5 provides an example execution trace.

5.5 TRAVELPLANNER: CROSS-LLM IDENTIFICATION

We use a feedback-enabled adaptation of TRAVELPLANNER’s sole-planning setting (Xie et al., 2024), supplying reference information and evaluating complete candidate plans. User-stated requirements form the candidate constraint class $\mathcal{H}$, and per-check evaluation messages provide feedback for testing candidate refinements. We evaluate on the same 50 validation queries with two LLM generators: Qwen3-4B (Yang et al., 2025) served locally, and DeepSeek-V4-Flash (DeepSeek-AI, 2026). We report the rate of committing all represented constraints, $C_v \supseteq C^*$, and final-plan success under this adapted protocol. The two generator configurations yield similar rates of committing all represented constraints, 84% and 86%, while their final-plan pass rates differ substantially, at 16% and 90%. These results distinguish explicit constraint commitment from producing a plan that passes every evaluation check. Both metrics are needed to assess this adaptation.

REFERENCES

- Jean-Yves Audibert, Sébastien Bubeck, and Rémi Munos. Best arm identification in multi-armed bandits. In *Proceedings of the 23rd Annual Conference on Learning Theory*, pp. 41–53, 2010.
- Sébastien Bubeck, Rémi Munos, Gilles Stoltz, and Csaba Szepesvári. X-armed bandits. *Journal of Machine Learning Research*, 12:1655–1695, 2011.
- Sébastien Bubeck, Tengyao Wang, and Nitin Viswanathan. Multiple identifications in multi-armed bandits. In *International Conference on Machine Learning*, pp. 258–265. PMLR, 2013.
- Lijie Chen, Jian Li, and Mingda Qiao. Towards instance optimal bounds for best arm identification. In *Conference on Learning Theory*, pp. 535–592. PMLR, 2017.
- Ching-An Cheng, Andrey Kolobov, Dipendra Misra, Allen Nie, and Adith Swaminathan. LLF-Bench: Benchmark for interactive learning from language feedback. *arXiv preprint arXiv:2312.06853*, 2023.
- DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026.
- Shivaram Kalyanakrishnan, Ambuj Tewari, Peter Auer, and Peter Stone. PAC subset selection in stochastic multi-armed bandits. In *Proceedings of the Twenty-ninth International Conference on Machine Learning*, pp. 655–662, 2012.
- Zohar Karnin, Tomer Koren, and Oren Somekh. Almost optimal exploration in multi-armed bandits. In *International Conference on Machine Learning*, pp. 1238–1246. PMLR, 2013.
- Julian Katz-Samuels and Clay Scott. Feasible arm identification. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 2535–2543. PMLR, 2018.

- Julian Katz-Samuels and Clayton Scott. Top feasible arm identification. In *The 22nd International Conference on Artificial Intelligence and Statistics*, pp. 1593–1601. PMLR, 2019.
- Robert Kleinberg, Aleksandrs Slivkins, and Eli Upfal. Bandits and experts in metric spaces. *Journal of the ACM*, 66(4):30:1–30:77, 2019.
- Akshay Krishnamurthy, Keegan Harris, Dylan J. Foster, Cyril Zhang, and Aleksandrs Slivkins. Can large language models explore in-context? In *Advances in Neural Information Processing Systems*, volume 37, pp. 120124–120158, 2024.
- Renyuan Li and Shenglong Zhu. Playing Mastermind with Wordle-like feedback. *The American Mathematical Monthly*, 131(5):390–399, 2024.
- Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, 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.
- Dipendra Misra, Aldo Pacchiano, and Robert E. Schapire. Provable interactive learning with hindsight instruction feedback. In *Proceedings of the 41st International Conference on Machine Learning*, pp. 35829–35850, 2024.
- Jie-Jing Shao, Bo-Wen Zhang, Xiao-Wen Yang, Bai-Zhi Chen, Si-Yu Han, Jing-Hao Pang, Wen-Da Wei, Guohao Cai, Zhenhua Dong, Lan-Zhe Guo, and Yu-Feng Li. ChinaTravel: An open-ended travel planning benchmark with compositional constraint validation for language agents. In *The Fourteenth International Conference on Learning Representations*, 2026.
- Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, 2023.
- Aleksandrs Slivkins. Introduction to multi-armed bandits. *Foundations and Trends in Machine Learning*, 12(1-2):1–286, 2019.
- Jian Xie, Kai Zhang, Jiangjie Chen, Tinghui Zhu, Renze Lou, Yuandong Tian, Yanghua Xiao, and Yu Su. TravelPlanner: A benchmark for real-world planning with language agents. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 54590–54613. PMLR, 2024.
- Wanqiao Xu, Allen Nie, Ruijie Zheng, Aditya Modi, Adith Swaminathan, and Ching-An Cheng. Provably learning from language feedback. *arXiv preprint arXiv:2506.10341*, 2025.
- An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.
- Shunyu Yao, Howard Chen, Austin W. Hanjie, Runzhe Yang, and Karthik R Narasimhan. COL-LIE: Systematic construction of constrained text generation tasks. In *The Twelfth International Conference on Learning Representations*, 2024.
- Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative AI by backpropagating language model feedback. *Nature*, 639(8055):609–616, 2025.
- Yizhe Zhang, Jiarui Lu, and Navdeep Jaitly. Probing the multi-turn planning capabilities of LLMs via 20 question games. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1495–1516. Association for Computational Linguistics, 2024.
- 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. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pp. 62138–62160, 2024.

A RELATED WORK

Learning from language feedback. The emerging language-feedback learning paradigm is tightly coupled with the rapid progress of language models. A first line of work treats natural-language feedback as a verbal reinforcement signal at inference time: REFLEXION (Shinn et al., 2023) maintains a self-reflective memory across trials, SELF-REFINE (Madaan et al., 2023) iteratively edits an output using its own critiques, and TEXTGRAD (Yuksekgonul et al., 2025) performs gradient-like updates over prompt tokens guided by language feedback. Tree-based agents such as LATS (Zhou et al., 2024) additionally search over reasoning steps, but rely on numeric value estimates rather than constraint signals. A complementary line studies whether large language models can themselves explore in interactive settings (Krishnamurthy et al., 2024), and a separate body of work develops constraint-rich language tasks that motivate our setup, including LLF-BENCH (Cheng et al., 2023), COLLIE (Yao et al., 2024), CHINATRAVEL (Shao et al., 2026), and TRAVELPLANNER (Xie et al., 2024). Two recent works establish formal guarantees for language-feedback learning under different formalisms: Misra et al. (2024) give $\sqrt{T}$ -regret bounds under hindsight instruction feedback, and HELIX (Xu et al., 2025) studies consistency-based elimination over a global hypothesis class. Our focus is orthogonal: we model a constraint-bearing regime where feedback provides local falsification and optional atomic identity evidence, and the algorithmic state is a partially committed constraint set rather than a full global hypothesis.

Pure exploration and identification problems. Our analysis belongs to the pure-exploration tradition in stochastic decision making, where the goal is to cover or identify a target object with high probability rather than to minimize cumulative reward. Best-arm identification under the fixed-budget setting was introduced by Audibert et al. (2010), with subsequent work establishing matching upper and lower bounds via sequential halving and multi-sample concentration arguments (Karnin et al., 2013; Chen et al., 2017). Among extensions, PAC top- K best-arm identification (Kalyanakrishnan et al., 2012; Bubeck et al., 2013) and feasible-arms identification (Katz-Samuels & Scott, 2018; 2019) share with our setting the goal of recovering a set-like target from a finite candidate class. We extend this perspective to language-feedback learning along two axes: progress comes from gap-dependent *falsification* of candidate constraint sets rather than reward concentration on individual arms, and identity-bearing feedback can propose atomic constraints that must still be tested before commitment. More broadly related are bandit problems over structured action spaces with numeric rewards (Slivkins, 2019; Bubeck et al., 2011; Kleinberg et al., 2019).

B COMPLETE THEORETICAL ANALYSIS

This appendix provides the complete theoretical analysis for constraint tree exploration. Section B.1 restates the assumptions used by the proofs. Section B.3 gives an exact-gap survival lemma and a safe-commitment lemma shared by both coverage proofs. Section B.4 analyzes an idealized sound-proposer regime. Section B.5 restates Theorem 2 in full as Theorem 7 and proves it for TRACE-F. Section B.6 restates Theorem 3 in full as Theorem 9 and proves it for TRACE-I. Section B.7 records an anytime gap-UCB bound derived from the same survival argument.

B.1 ASSUMPTIONS

We use the notation from Section 2. In particular, each tree node v is associated with a constraint set C_v , feasible region $\mathcal{X}_v = \text{Region}(C_v)$, node value $U(v) := \max_{x \in \mathcal{X}_v} U(x)$, and node gap $\Delta(v) := U^* - U(v)$. We write $\text{Viol}(f; C_v) = 1$ when the detector concludes that feedback f contradicts at least one constraint in the tested set C_v ; feedback about a missing but uncommitted constraint does not fire this predicate. The assumptions below also require a refutational feedback condition: if a tested committed set excludes all optima, feedback must contradict that set with positive probability. We write $\mathcal{H}$ for the finite candidate class used by the proposer. Throughout, we assume $K \geq 1$; if $K = 0$, then $\text{Region}(C^*) = \mathcal{X}$ and the root already satisfies the coverage target.

Oracle-level noise model. The raw message $f \sim P_x$ may be stochastic, but the analysis imposes reliability conditions on the oracle outputs: the firing probability of Viol and the extraction probability of Extract. The theoretical model allows false negatives but rules out false positives. This one-sided

noise model supports the multi-sample falsification argument, while Section 5.4 empirically evaluates robustness to spurious extracted identities.

Assumption 3 (Refutational feedback and extraction). *For any node v for which Q_{C_v} is defined, let*

$$q_v := \Pr[\text{Viol}(f; C_v) = 1 \mid x \sim Q_{C_v}, f \sim P_x].$$

There exist constants $c_{\text{fal}} > 0$ and $\beta > 0$ such that

$$q_v = 0 \quad \text{if } \Delta(v) = 0, \quad q_v \geq \phi(\Delta(v)) := \min\{1, c_{\text{fal}}\Delta(v)^\beta\} \quad \text{if } \Delta(v) > 0.$$

The equality $q_v = 0$ is the no-false-alarm side of the detector contract. The lower bound for $\Delta(v) > 0$ is an additional refutation-power assumption on the feedback distribution: a bad tested region must produce feedback that contradicts some committed constraint with probability controlled by its gap. Without this assumption, a user who only mentions missing uncommitted requirements could leave a bad committed constraint unfalsified. Without loss of generality, $c_{\text{fal}} \leq 1$: replacing c_{fal} by $\min\{c_{\text{fal}}, 1\}$ preserves this condition, and the normalization lets the rounding in m_{exp} be absorbed into the $O(\cdot)$ budget bounds below. We call this condition on q_v the refutational-feedback condition. Every result in this appendix except Theorem 9 uses only this condition; the extraction clauses (a)–(b) below are needed only in the identification setting, and TRACE-F uses a null extractor. In the identification setting, the extractor $\text{Extract} : \mathcal{F} \rightarrow \mathcal{H} \cup \{\emptyset\}$ satisfies the following two conditions.

- (a) **Extraction reliability.** *Conditional on the event $\text{Viol}(f_t; C_v) = 1$ for the constraint set currently being tested, the extractor returns a violated canonical constraint with probability at least $p_{\text{ext}} > 0$, independently across rounds. It returns $\emptyset$ otherwise.*
- (b) **Soundness.** *If $\text{Extract}(f_t) = c \neq \emptyset$, then $c \in C^*$ and $\text{Sat}(x_t, c) = 0$. In particular, extraction never returns a spurious constraint.*

Assumption 4 (Constrained generation). *For every constraint set C considered by the algorithm with nonempty feasible region, there exists a distribution Q_C supported on $\text{Region}(C) = \{x \in \mathcal{X} : \text{Sat}(x, c) = 1 \forall c \in C\}$. Each call to Q_C returns an independent sample.*

Assumption 5 (Sound proposer). *The proposer Propose is a possibly randomized map from nodes to $\mathcal{H} \cup \{\emptyset\}$ satisfying:*

- (a) **Completeness.** *If $C_v \subsetneq C^*$, then $\text{Propose}(v) \in C^* \setminus C_v$.*
- (b) **Termination marker.** *If $C_v = C^*$, then $\text{Propose}(v) = \emptyset$.*

Assumption 5 is an idealization. It is useful as a warm-up because it isolates the role of tree depth and within-node spread. The main robust analysis below replaces it with a noisy proposer.

Assumption 6 (Candidate class and noisy proposer). *The finite canonical constraint class $\mathcal{H}$ from Section 2.1 satisfies $C^* \subseteq \mathcal{H}$ and $|\mathcal{H}| = H \geq K$. For every $C \subseteq \mathcal{H}$ with nonempty feasible region, the generator Q_C is defined as in Assumption 4. At each proposal call from node v , the proposer returns an element of $\mathcal{H}_v = \mathcal{H} \setminus (C_v \cup R_v)$ sampled uniformly at random, independently of the past given (v, R_v) , where R_v is the per-node lazy filter of Algorithm 1. If $\mathcal{H}_v = \emptyset$, it returns $\emptyset$.*

Remark 1 (Per-node lazy filter). The set R_v in Algorithm 1 records the candidates that gave an empty region or were rejected at v ; it starts empty at each node, and proposals are drawn from $\mathcal{H}_v = \mathcal{H} \setminus (C_v \cup R_v)$. At a viable node, a missing true constraint never enters R_v : its child contains a global optimum, so the region is nonempty and, by Assumption 3, the child is never falsified (a test cut short by budget exhaustion also adds its candidate to R_v , but ends the run). Hence missing true constraints remain available at viable nodes, which is what the proofs of Theorems 7 and 9 use. Since each proposal at v either enters R_v or is committed, at most $|\mathcal{H} \setminus C_v| + 1$ proposal calls occur at each node, so Algorithm 1 terminates.

Assumption 7 (ε -separated spurious children). *Fix $\varepsilon \in (0, 1]$. For every viable node v (i.e., $\Delta(v) = 0$) and every $c \in \mathcal{H} \setminus C_v$ whose child $u = v \cup \{c\}$ has a nonempty feasible region $\mathcal{X}_u$, this child satisfies either $\Delta(u) = 0$ (it preserves a global optimum) or $\Delta(u) > \varepsilon$ (it excludes every optimum; we call such a child ε -spurious).*

Assumption 7 is a structural condition on $(\mathcal{X}, U, \mathcal{H}, C^*)$, not on the generator Q_C , detector Viol , extractor Extract , or proposer Propose . It plays the role of a sub-optimality gap: without it, children with arbitrarily small but positive $\Delta(u)$ would require unboundedly many samples to falsify. The gap $\Delta(u)$ is defined only when $\mathcal{X}_u \neq \emptyset$. Children with empty regions are never tested by Algorithm 1, so Assumptions 7 and 8 impose no condition on them.

Assumption 8 (No zero-gap distractors). *For the identification-setting analysis, every $c \in \mathcal{H} \setminus C^*$ with $\text{Region}(c) \neq \emptyset$ satisfies $\Delta(\{c\}) > \varepsilon$; that is, imposing c alone already excludes every global optimum with margin ε .*

Assumption 8 is the structural condition used by Theorem 9. It extends to every viable node v : a true constraint gives a child that keeps the optimum of v , while for $c \notin C^*$ with a nonempty child, $\text{Region}(C_v \cup \{c\}) \subseteq \text{Region}(c)$ gives $\Delta(v \cup \{c\}) \geq \Delta(\{c\}) > \varepsilon$. Hence it implies Assumption 7, and, since $K \geq 1$ makes the root such a node, it is equivalent to requiring the same property at every viable node that does not yet contain C^* . It depends only on $(\mathcal{X}, U, \mathcal{H}, C^*)$, and extractor reliability cannot ensure it; without it, an identification-aware proposer could spend budget committing harmless but non-informative constraints, reintroducing dependence on their number.

The extraction progress used by Theorem 9 follows from Assumption 3(a)–(b). Suppose a test of a child u of a viable node v starts with working set $W_{\text{start}} \subseteq C_v$ and ends with a violation. Conditional on the history and on u , the rounds of the test are independent, so the feedback at the first violation has the one-round law conditional on a violation, and by (a) the extractor then returns a constraint with probability at least p_{ext} . By (b) this constraint lies in C^* and is violated by the sampled action, which satisfies C_u ; hence it lies in $C^* \setminus C_u \subseteq C^* \setminus (C_v \cup W_{\text{start}})$, so $|C_v \cup W|$ has increased during the test, at this round or at an earlier round that returned the same constraint. No independence between different tests is needed.

The same conditions restrict the instances covered by Theorem 9: together with Assumption 3, Assumption 8 forces $\text{Region}(c) \cap \text{Region}(C^*) = \emptyset$ for every $c \in \mathcal{H} \setminus C^*$. Otherwise the node $D = C^* \cup \{c\}$ is nonempty with $\Delta(D) \geq \Delta(\{c\}) > \varepsilon$, so the refutational-feedback condition gives $q_D > 0$; but every action in $\text{Region}(D)$ satisfies all true constraints, so by (b) no extraction can succeed at D , contradicting (a). The identification guarantee therefore applies when every incorrect candidate is incompatible with the true requirements, as for wrong values of single-valued attributes; Theorem 7 does not use (a)–(b) and is unaffected.

B.2 ALGORITHMIC REMARKS

The remarks in this subsection record operational properties of Algorithm 1 that are referenced from the main text. They are not used in any proof.

Remark 2 (Working-set persistence and bounded growth). The working set W in Algorithm 1 is initialized once and updated only by union; it is never cleared. Four observations characterize its behavior. (a) $|W| \leq H$ since $W \subseteq \mathcal{H}$. (b) The identification proposer reads $W \cap \mathcal{H}_v$ rather than W directly, so already-committed and already-excluded constraints are filtered at proposal time. (c) Under sound extraction (Assumption 3(b)), every $c \in W$ is a true latent constraint; once proposed at a viable node v , the resulting child has $\Delta = 0$, is never falsified by the no-false-alarm clause of Assumption 3, and is committed at the end of its m_{exp} -round test window whenever the remaining budget covers that window. (d) Under empirically unsound extraction, false entries in W may be repeatedly proposed. Proposing such an entry at a viable node yields an empty child, which is excluded without a test, a zero-gap child, which may be committed but still contains a global optimum, or, by Assumption 7, a child with $\Delta > \varepsilon$. Lemma 5 allows any history-dependent proposer, so with m_{exp} as in Theorems 7 and 9 and a detector that still satisfies the refutational-feedback condition, no child with positive gap is committed with probability at least $1 - \delta/2$; false entries then cost test budget while the committed node remains viable. This controls harmful commitments only; the K/p_{ext} guarantee of Theorem 9 additionally requires sound extraction and Assumption 8. This is the algorithmic backing for the false-identity robustness check in Section 5.4. Combined with the per-node lazy filter (Remark 1), TRACE-I cannot loop on a spurious extracted identity once it has been falsified at v : the lazy filter excludes it from subsequent proposals at the same node.

Remark 3 (Detector contract, operational reading). Per Assumption 3, Viol is used only for falsification: feedback that reveals a missing requirement may update W via Extract, but does not reject the tested child unless it contradicts a constraint already in C_u . Equivalently, the detector fires on contradictions with the tested set, never on discovery feedback alone. As an illustration, if $C_u = \{\text{genre} = \text{comedy}\}$, then “I did not want a comedy” contradicts C_u and fires Viol, whereas “I also wanted something from the 1990s” is discovery feedback and must not fire Viol.

Remark 4 (Counters in Algorithm 1). The variable t is the global interaction-round counter running against the budget T ; the inner-loop index s counts tests of the current child running against the

per-child cap m_{exp} . The two counters are independent: s resets on each new child while t is monotone over the entire run. Both caps are needed because the algorithm enforces a global budget T via t and a per-child test window m_{exp} via s .

B.3 BASIC LEMMAS

Lemma 4 (Multi-sample falsification, exact-gap form; implies Lemma 1). *Let v be a node for which Q_{C_v} is defined, and let $x_s \sim Q_{C_v}$ and $f_s \sim P_{x_s}$ for m independent rounds. Then*

$$\Pr[\text{Viol}(f_s; C_v) = 0 \text{ for all } s = 1, \dots, m] = (1 - q_v)^m \leq \exp(-mc_{\text{fal}}\Delta(v)^\beta).$$

In particular, a node with $\Delta(v) > \varepsilon$ survives m tests with probability at most $\exp(-mc_{\text{fal}}\varepsilon^\beta)$, which is at most δ' when $m \geq \log(1/\delta')/(c_{\text{fal}}\varepsilon^\beta)$.

Proof. Since $c_{\text{fal}} \leq 1$ and $\Delta(v) \leq 1$, we have $c_{\text{fal}}\Delta(v)^\beta \leq 1$, so Assumption 3 gives $q_v \geq c_{\text{fal}}\Delta(v)^\beta$, also when $\Delta(v) = 0$. The rounds are independent by Assumption 4, and $1 - y \leq e^{-y}$ gives the bound. $\square$

Lemma 5 (Safe commitments). *Suppose the refutational-feedback condition of Assumption 3 and Assumptions 4 and 7 hold. Run Algorithm 1 with any history-dependent proposer, test length m , at most N tests, and $T = Nm$. Then each test has a complete window of m rounds, and the probability that some child with $\Delta > 0$ is committed is at most $N \exp(-mc_{\text{fal}}\varepsilon^\beta)$.*

Proof. The n -th test starts after at most $(n-1)m$ rounds, so it has m rounds available, and a child is committed only after m violation-free rounds. Before the first commitment of a child with $\Delta > 0$, every committed node is viable, so by Assumption 7 the child tested at that moment has $\Delta > \varepsilon$. Conditional on the history and the proposed child, the test rounds are fresh independent samples, so Lemma 4 bounds its acceptance probability by $\exp(-mc_{\text{fal}}\varepsilon^\beta)$. A union bound over the at most N tests at which this first commitment can occur gives the claim. $\square$

B.4 IDEALIZED SOUND-PROPOSER REGIME

We first analyze an idealized regime in which proposed constraints are always true. This result is mainly a warm-up: it shows that once proposals are sound, constraint-tree depth alone is enough to reach the latent feasible region.

Theorem 6 (Coverage under a sound proposer). *Suppose the refutational-feedback condition of Assumption 3 holds, together with Assumptions 4 and 5. Consider the idealized variant of TRACE that tests the current node for $m_{\text{exp}} \geq 1$ rounds and then calls Propose whenever the node survives those rounds without falsification. If the budget satisfies $T \geq Km_{\text{exp}}$, then with probability one the procedure reaches a node v with $C_v = C^*$. Consequently, if $\hat{x} \sim Q_{C_v}$, then $\hat{x} \in \text{Region}(C^*)$ and $U(\hat{x}) \geq U^* - \text{Spread}(C^*)$.*

Proof. We argue by induction over the committed path. Initially, the root satisfies $C_{v_0} = \emptyset \subseteq C^*$. Suppose the current node v satisfies $C_v \subseteq C^*$. Since $x^* \in \text{Region}(C_v)$, we have $U(v) = U^*$ and therefore $\Delta(v) = 0$. By the no-false-alarm part of Assumption 3, the node is never falsified: $q_v = 0$. Thus the node survives any test window of length m_{exp} .

If $C_v \subsetneq C^*$, then by Assumption 5 the proposer returns a constraint $c \in C^* \setminus C_v$. The algorithm moves to the child u with $C_u = C_v \cup \{c\}$. This child also satisfies $C_u \subseteq C^*$ and has depth one larger than v . Therefore, each successful expand step strictly increases $|C_v|$ while preserving the invariant $C_v \subseteq C^*$.

After at most K such expansions, the committed node satisfies $C_v = C^*$. Each expansion consumes exactly m_{exp} test rounds, so the budget $T \geq Km_{\text{exp}}$ suffices. Finally, by the support property of Q_{C_v} , $\hat{x} \sim Q_{C_v}$ implies $\hat{x} \in \text{Region}(C^*)$. By the definition of $\text{Spread}(C^*)$, $U(\hat{x}) \geq U^* - \text{Spread}(C^*)$. $\square$

B.5 NOISY-PROPOSER REGIME

We now analyze the robust test-before-commit algorithm under a noisy proposer. This is the main theoretical setting used in the paper.

Theorem 7 (Coverage in the falsification setting; full version of Theorem 2). *Suppose the refutational-feedback condition of Assumption 3 holds (its extraction clauses (a)–(b) are not needed), together with Assumptions 4, 6, and 7. Fix $\delta \in (0, 1)$ and $\varepsilon \in (0, 1]$. If*

$$N_{\max} := \left\lceil H \log \frac{2 \max(K, 1)}{\delta} \right\rceil, \quad m_{\exp} := \left\lceil \frac{\log(2N_{\max}/\delta)}{c_{\text{fal}} \varepsilon^\beta} \right\rceil, \quad T := N_{\max} m_{\exp},$$

then with probability at least $1 - \delta$ the deepest committed node v satisfies $\text{Region}(C_v) \subseteq \text{Region}(C^*)$. In the canonical finite-class proof, this region inclusion is obtained through the sufficient event $C_v \supseteq C^*$; the committed set may also include additional zero-gap constraints from $\mathcal{H} \setminus C^*$. Hence, if $\hat{x} \sim Q_{C_v}$, then $\hat{x} \in \text{Region}(C^*)$ and $U(\hat{x}) \geq U^* - \text{Spread}(C^*)$. With $L_F := \log(2K/\delta)$, the total interaction budget is

$$T = \mathcal{O}\left(\frac{H L_F \log(4H L_F / \delta)}{c_{\text{fal}} \varepsilon^\beta}\right).$$

Proof. Write $N := N_{\max}$ and $m := m_{\exp}$, and let $\mathcal{B}$ be the event that a child with $\Delta > 0$ is committed. By Lemma 5 and the choice of m , $\Pr(\mathcal{B}) \leq N \exp(-m c_{\text{fal}} \varepsilon^\beta) \leq \delta/2$.

Index the proposal calls, including those whose candidate gives an empty region, and let $\mathcal{F}_j$ be the history after j calls and their tests. Fix $c \in C^*$, and let $A_{j,c}$ be the event that no child with $\Delta > 0$ has been committed during the first j calls and c is still uncommitted after them. On $A_{j,c}$ the committed node v is viable, so its region contains a global optimum, which satisfies c ; hence the child $v \cup \{c\}$ is nonempty with $\Delta = 0$, and $c \in \mathcal{H}_v$ by Remark 1. For $j < N$, at most j tests and jm rounds have been used, so the next call takes place and any test it triggers has a complete window. This call proposes c with conditional probability $1/|\mathcal{H}_v| \geq 1/H$, and the resulting child then passes and is committed. Hence

$$\Pr(A_{j+1,c} \mid \mathcal{F}_j) \leq \left(1 - \frac{1}{H}\right) \mathbf{1}_{A_{j,c}}, \quad \text{so} \quad \Pr(A_{N,c}) \leq \left(1 - \frac{1}{H}\right)^N \leq e^{-N/H}.$$

If $\mathcal{B}$ does not occur but some $c \in C^*$ is never committed, the run makes at least N calls, since a viable run that misses a true constraint cannot stop earlier, and therefore $A_{N,c}$ occurs. By the choice of N ,

$$\Pr(C^* \not\subseteq C_v) \leq \Pr(\mathcal{B}) + \sum_{c \in C^*} \Pr(A_{N,c}) \leq \frac{\delta}{2} + K e^{-N/H} \leq \delta.$$

On the complement, $C_v \supseteq C^*$, so $\text{Region}(C_v) \subseteq \text{Region}(C^*)$; the support of Q_{C_v} gives $\hat{x} \in \text{Region}(C^*)$, and the definition of $\text{Spread}(C^*)$ gives $U(\hat{x}) \geq U^* - \text{Spread}(C^*)$. Each of the at most N tests uses at most m rounds, so $T = Nm$ suffices. $\square$

Corollary 8 (Expected simple regret). *Under the setting of Theorem 7, the expected simple regret satisfies*

$$\mathbb{E}[r_T] \leq \text{Spread}(C^*) + \delta.$$

Proof. Let E be the success event in Theorem 7. Then $\Pr(E) \geq 1 - \delta$. On E , the output satisfies $r_T = U^* - U(\hat{x}) \leq \text{Spread}(C^*)$. On E^c , since $U \in [0, 1]$, we have $r_T \leq 1$. Therefore

$$\mathbb{E}[r_T] \leq (1 - \delta) \text{Spread}(C^*) + \delta \cdot 1 \leq \text{Spread}(C^*) + \delta.$$

$\square$

B.6 IDENTIFICATION SETTING

This section analyzes TRACE-I, which augments the falsification-setting test-before-commit core with an identification-aware proposer that biases its candidates toward constraints extracted from past feedback.

Identification-aware proposer. The proposer maintains a running set $W_t \subseteq \mathcal{H}$ initialized to $\emptyset$. After every round in which $\text{Extract}(f_t) = c \neq \emptyset$, it updates $W_{t+1} \leftarrow W_t \cup \{c\}$, matching Line 10 of Algorithm 1; under the soundness clause of Assumption 3, every such c lies in C^* , so the update is sound regardless of whether $\text{Viol}(f_t; C) = 1$ also fires, and the discovery argument below only uses the subset of these updates that occur on rounds with $\text{Viol}(f_t; C) = 1$. At each proposal call from node v , it returns a uniform random element of $W_t \cap \mathcal{H}_v$ when this set is nonempty, and a uniform random element of $\mathcal{H}_v$ otherwise, with $\mathcal{H}_v = \mathcal{H} \setminus (C_v \cup R_v)$ as in Algorithm 1; it returns $\emptyset$ if $\mathcal{H}_v = \emptyset$.

Theorem 9 (Coverage in the identification setting; full version of Theorem 3). *Suppose Assumptions 3, 4, and 8 hold (in particular $C^* \subseteq \mathcal{H}$; Assumption 8 implies Assumption 7). Fix $\delta \in (0, 1)$ and $\varepsilon \in (0, 1]$. With*

$$N_{\max}^{\text{id}} := \left\lceil \frac{64}{p_{\text{ext}}} \left(K + \log \frac{4}{\delta} \right) \right\rceil, \quad m_{\text{exp}}^{\text{id}} := \left\lceil \frac{\log(2N_{\max}^{\text{id}}/\delta)}{c_{\text{fal}}\varepsilon^\beta} \right\rceil, \quad T := N_{\max}^{\text{id}} m_{\text{exp}}^{\text{id}},$$

TRACE-IDENTIFICATION enjoys an (ε, δ) -coverage guarantee: with probability at least $1 - \delta$, the deepest committed node v satisfies $\text{Region}(C_v) \subseteq \text{Region}(C^*)$. In the canonical proof this follows from the sufficient event $C_v \supseteq C^*$. Consequently, if $\hat{x} \sim Q_{C_v}$, then $\hat{x} \in \text{Region}(C^*)$ and $U(\hat{x}) \geq U^* - \text{Spread}(C^*)$. With $L_I := K + \log(4/\delta)$, the total interaction budget is

$$T = \mathcal{O}\left(\frac{L_I \log(4L_I/(p_{\text{ext}}\delta))}{p_{\text{ext}} c_{\text{fal}}\varepsilon^\beta}\right) = \tilde{\mathcal{O}}\left(\frac{K}{p_{\text{ext}} c_{\text{fal}}\varepsilon^\beta}\right).$$

Proof. Let $N := N_{\max}^{\text{id}}$, $m := m_{\text{exp}}^{\text{id}}$, and $p := p_{\text{ext}}$. Assumption 8 implies Assumption 7, so by Lemma 5 and the choice of m , with probability at least $1 - \delta/2$ no child with $\Delta > 0$ is committed during the first N tests, and each of these tests has a complete window. Call *recovery* the first time at which $C^* \subseteq C_v$.

Before recovery and before any child with $\Delta > 0$ is committed, every committed node is viable and, by Assumption 8, every committed constraint is true, so $C_v \subseteq C^*$; soundness gives $W_t \subseteq C^*$. We track the potential

$$\Psi := |C_v| + |C_v \cup W_t| \leq 2K,$$

which never decreases and gains one unit when a true constraint is discovered and one unit when it is committed. Consider a test in this period. A prioritized test proposes a constraint of $W_t \cap \mathcal{H}_v \subseteq C^* \setminus C_v$, which is committed, so Ψ increases. Otherwise $W_t \cap \mathcal{H}_v = \emptyset$, which at a viable node means $W_t \subseteq C_v$, because missing true constraints never enter R_v (Remark 1). A fallback proposal in C^* is then committed and increases Ψ by two. A fallback proposal outside C^* is either committed, which is a commitment with $\Delta > 0$, or rejected after a violation; in the latter case, with conditional probability at least p , the extractor returns a true constraint that is new relative to the working set at the start of the test, so Ψ increases during this test (see the discussion after Assumption 8). Hence, given the history, each such test increases Ψ or commits a child with $\Delta > 0$ with probability at least p .

For $n \leq N$, let $Y_n := 1$ if the n -th test increases Ψ or commits a child with $\Delta > 0$, and let $Y_n := 1$ if this test starts after recovery or after such a commitment, or does not take place. Before these events the run does not stop within N tests: a missing true constraint stays in $\mathcal{H}_v$, and the budget covers N complete windows. Therefore $\Pr(Y_n = 1 \mid \mathcal{F}_{n-1}) \geq p$ for the history $\mathcal{F}_{n-1}$ before the n -th test, and the tower property gives

$$\mathbb{E}\left[\exp\left(-\lambda \sum_{n=1}^N Y_n\right)\right] \leq (1 - p + pe^{-\lambda})^N \quad \text{for every } \lambda > 0.$$

Since $Np \geq 64(K + \log(4/\delta))$, we have $2K \leq Np/2$ and $e^{-Np/8} \leq \delta/2$, so the multiplicative Chernoff lower-tail bound gives $\Pr[\sum_{n=1}^N Y_n < 2K] \leq \delta/2$.

If recovery does not occur within the first N tests and no child with $\Delta > 0$ is committed, then every $Y_n = 1$ records an increase of Ψ , which starts at 0 and stays at most $2K - 1$ because $|C_v| \leq K - 1$; hence $\sum_{n=1}^N Y_n < 2K$. Therefore recovery occurs within the first N tests, and thus within $T = Nm$ rounds, with probability at least $1 - \delta$. Later commitments only add constraints, so at termination $C_v \supseteq C^*$ and $\text{Region}(C_v) \subseteq \text{Region}(C^*)$; the support of Q_{C_v} and the definition of $\text{Spread}(C^*)$ give the remaining claims, as in Corollary 8. $\square$

B.7 ANYTIME GAP-UCB BOUND

The survival argument also yields an anytime confidence bound on the gap of a tested node. It is not needed for Theorems 7 and 9, but it reads the test of Algorithm 1 as a confidence rule.

Lemma 10 (Anytime gap upper confidence bound). *Fix a node v for which Q_{C_v} is defined, and consider an independent sequence of rounds at v . For $m \geq 1$, let $A_m := \{\text{Viol}(f_s; C_v) = 0 \text{ for all } 1 \leq s \leq m\}$ be the event that the first m rounds at v produce no violation signal. For $\delta \in (0, 1)$, define*

$$\hat{\Delta}_{\text{UCB}}(v; m, \delta) := \left(\frac{\log(1/\delta)}{c_{\text{fal}} m} \right)^{1/\beta}.$$

Then, with probability at least $1 - \delta$, $\Delta(v) \leq \hat{\Delta}_{\text{UCB}}(v; m, \delta)$ holds simultaneously for every $m \geq 1$ such that A_m occurs. The events A_m concern the initial violation-free prefix; violation-free runs that restart after a violation are not covered, since each restart would require additional confidence.

Proof. If $\Delta(v) = 0$ there is nothing to prove. Otherwise let $a := c_{\text{fal}} \Delta(v)^\beta$. Lemma 4 gives $\Pr(A_m) \leq e^{-ma}$. The radius decreases in m , and $\Delta(v) > \hat{\Delta}_{\text{UCB}}(v; m, \delta)$ holds exactly when $m \geq m_* := \lceil \log(1/\delta)/a \rceil + 1$. Because $A_{m+1} \subseteq A_m$, the failure event $\{\exists m \geq 1 : A_m \text{ occurs and } \Delta(v) > \hat{\Delta}_{\text{UCB}}(v; m, \delta)\}$ equals A_{m_*} , whose probability is at most $e^{-m_* a} < \delta$. $\square$

Interpretation. The bound in Lemma 10 is the gap-dependent analogue of a UCB confidence radius. Because the prefix events are nested, no union bound over m is needed, and for fixed δ the radius decays as $\mathcal{O}(m^{-1/\beta})$ without a logarithmic factor in m ; larger β corresponds to a more ambiguous feedback channel and slower decay.

Remark 5 (Anytime reading of the test in Algorithm 1). Accepting a child once $\hat{\Delta}_{\text{UCB}}(v; m, \delta) \leq \varepsilon$ and rejecting it at the first violation is exactly the test of Algorithm 1: the radius depends on the data only through survival, so acceptance happens after the fixed window $\lceil \log(1/\delta)/(c_{\text{fal}} \varepsilon^\beta) \rceil$, which equals m_{exp} when δ is replaced by $\delta/(2N_{\text{max}})$. Lemma 10 therefore certifies, at every round of a test, how large the gap of a surviving child can still be.

C ADDITIONAL EXPERIMENTAL DETAILS

This appendix provides the full experimental details omitted from the main text. Section C.1 describes the environments and implementation choices. Section C.2 reports the three remaining task instances of the main comparison and Wilson confidence intervals for the main success-rate tables. Section C.3 gives the RecMovie ablations, robustness checks, and equal-output-budget comparisons. Section C.4 covers 20Q, EDA, MASTERMIND, and Poem. Section C.5 gives an example execution trace, and Section C.6 lists the prompts of the LLM-based components.

C.1 ENVIRONMENT AND IMPLEMENTATION DETAILS

Common setup. Unless otherwise noted, TRACE uses $m_{\text{exp}} = 15$ and $N_{\text{max}} = 15$. The headline Viol and Extract modules are rule-based: a regex over each environment’s feedback template, with task-specific decoders that map matched phrases to canonical predicates in $\mathcal{H}$. TRACE-FALSIFICATION (TRACE-F) uses a uniform proposer; TRACE-IDENTIFICATION (TRACE-I) uses the same decoder to extract identities and prioritize them in the proposer. Qwen3-4B served locally is used for (i) the prompt-based LLM baselines (DIRECTLLM, REFLEXION, SELF-REFINE), (ii) the Qwen3-4B configuration of TRAVELPLANNER’s constrained plan generator Q_C in Section 5.5, and (iii) the LLM-based Viol / Extract ablations in Tables 12 and 13; these ablations confirm that headline TRACE rows do not rely on LLM oracle modules, since substituting the LLM detector for the rule-based detector drops TRACE-F’s interaction-budget any-success from 57% to 21% on RecMovie. The MASTERMIND column of the main comparison (Table 6) fixes $K = 3$ and $n_{\text{colors}} = 6$; Haiku targets the 5–7–5 syllable pattern and Tanka the 5–7–5–7–7 pattern. Wilson 95% intervals for all main-table cells are reported in Section C.2; small differences within overlapping intervals should not be over-interpreted. Prompt-based baselines (DIRECTLLM, REFLEXION, SELF-REFINE) are operational comparisons under a fixed local model, rather than a multi-model robustness

study. All reported results use 100 seeds per cell, except TRAVELPLANNER, which uses 50 validation queries (Section 5.5). The primary metric is *env.success*, the fraction of seeds on which a method produces a satisfying output within its interaction budget; for RecMovie TRACE rows this is the final-output success of the algorithm’s converged recommendation list. Section C.3 additionally reports an interaction-budget any-success diagnostic and a soft-utility breakdown. When available, we also report $\text{superset}(C^*)$, which checks whether the committed constraint set contains all true constraints and serves as a sufficient syntactic proxy for region inclusion.

Compute resources. All experiments were run on a local workstation with a single NVIDIA GeForce RTX 5080 GPU. LLM-based baselines, the TRAVELPLANNER plan generator Q_C , and the LLM-based Viol / Extract ablation rows used Qwen3-4B on this workstation, and the DeepSeek-V4-Flash configuration of TRAVELPLANNER called the DeepSeek API; headline TRACE runs and the remaining environment simulation, rule-based decoding, and tabulation code ran on CPU.

LLM baselines. LLM baselines use Qwen3-4B. We compare against DIRECTLLM, REFLEXION (Shinn et al., 2023), and SELF-REFINE (Madaan et al., 2023). DIRECTLLM conditions on the full interaction history. REFLEXION follows the original trial-based design with sliding-window reflective memory; SELF-REFINE follows Madaan et al. (2023) with full-history critique-and-revise.

RecMovie. Each seed samples a user profile across multiple attribute dimensions (movie type, year range, genre, age restriction); the corresponding latent constraint set C^* collects the dimensions actually set in the profile, so $K = |C^*|$ varies per seed. On each round, the agent proposes three movie titles, and the environment returns template-based verbal feedback. All LLM baselines run for $T = 20$ total interaction steps on RecMovie; trial-based Reflexion partitions this budget across trials. Across 100 seeds, the LLM baselines that succeed do so within roughly 3 steps on average (DirectLLM mean 3.7, Self-Refine mean 2.9), while failed seeds exhaust nearly the full budget. Appendix C.3 compares all methods under common caps of 20 and 60 outputs; there, the windowed baseline variants gain 2–10pp from the larger cap.

20Q. 20Q is a controlled synthetic environment with five categorical attribute dimensions yielding $|\mathcal{H}| = 17$ candidate predicates over a dense $|\mathcal{X}| = 432$ -entity catalog (every attribute combination has exactly one entity). Each episode samples a profile inducing C^* whose size $K = |C^*|$ varies per seed; feedback is templated per dimension, e.g., “the target is not a mammal”. Because the catalog is dense, $\text{Region}(C_v)$ is nonempty for every consistent constraint set C_v .

EDA THINGS. We use the 99-entity test set of Apple’s Entity-Deduction Arena (EDA THINGS) (Zhang et al., 2024), annotated by us with five categorical attribute dimensions; annotations were initialized by Qwen3-4B and manually corrected. Each episode samples a profile inducing C^* whose size $K = |C^*|$ varies per seed. The catalog is sparse, so when a committed constraint set yields an empty region, the generator falls back to the entity satisfying the largest number of committed predicates.

MASTERMIND. We use a per-position MASTERMIND environment, following the *Clear Mastermind* feedback style (Li & Zhu, 2024). The hidden code has length K , and each position takes one color from a palette of size n_{colors} . Feedback is per-position: each proposed position is marked as “exact”, “present”, or “absent”. The candidate class is $\mathcal{H} = \{(i, c) : i \in [K], c \in \text{palette}\}$ with $|\mathcal{H}| = K \cdot n_{\text{colors}}$. This environment allows controlled sweeps over both K and $|\mathcal{H}|$.

LLF-Bench Poem. For LLF-Bench Poem (Cheng et al., 2023), we use the Haiku and Tanka tasks without changing their reward, feedback content, or target pattern. Haiku has target syllable pattern 5–7–5 and thus $K = 3$ line-level constraints. Tanka has target pattern 5–7–5–7–7 and thus $K = 5$ constraints. We apply a runtime patch to the upstream `cmudict` lookup so that missing words fall back to the existing heuristic syllable estimator; this patch is used only for words outside the CMU dictionary. The generator is a word-bucket generator that samples words from syllable-indexed buckets and composes each line to satisfy a proposed syllable count. The predicate class is $\mathcal{H} = \{(\text{line}_i, \text{syll}_s) : i \in [K], s \in \{3, 4, 5, 6, 7, 8\}\}$ with $|\mathcal{H}| = 6K$. TRACE reads only feedback, not the task prompt, so the target syllable pattern is discovered from the environment replies.

K across environments. $K = |C^*|$ is fixed per experiment in MASTERMIND (main comparison: $K = 3$, $n_{\text{colors}} = 6$; H-sweep: $K = 3$; K-sweep: $K \in \{2, 4, 6, 8, 10\}$) and Poem (Haiku: $K = 3$; Tanka: $K = 5$). In RecMovie, 20Q, and EDA THINGS, K is sampled per seed by the environment’s profile sampler.

C.2 REMAINING MAIN-COMPARISON TASKS AND CONFIDENCE INTERVALS

Remaining task instances. For headline focus, Table 1 reports only the three tasks where the falsification-to-identification gap is non-saturated and the lossy-extraction story (RecMovie) is empirically separable. Table 6 records the corresponding natural-feedback main-comparison numbers on the three remaining task instances: MASTERMIND ($K = 3$, $n_{\text{colors}} = 6$), LLF-Bench Haiku ($K = 3$ syllable constraints), and LLF-Bench Tanka ($K = 5$). TRACE-I reaches 99–100% on all three; natural-feedback prompt baselines remain below 30%. Wilson 95% intervals are in Table 7 (Mmind / Haiku / Tanka columns).

Table 6: Main-comparison env.success on the three task instances moved out of Table 1 for headline focus. Entries are env.success within each method’s interaction budget; higher is better.

Method	Mastermind	Haiku	Tanka
Random	4%	7%	0%
DirectLLM	15%	20%	4%
Reflexion (Shinn et al., 2023)	9%	2%	1%
Self-Refine (Madaan et al., 2023)	29%	14%	0%
TRACE-F	54%	54%	14%
TRACE-I	99%	100%	100%

Wilson confidence intervals. All success rates in the main tables use 100 seeds per cell, except Table 5 (50 queries). Tables 7–10 report Wilson 95% confidence intervals rounded to the nearest percentage point. These intervals are intended for interpreting success-rate differences, not for changing the primary point estimates shown in the main text.

Table 7: Wilson 95% confidence intervals for natural-feedback baselines and TRACE rows. The first three columns correspond to the Natural columns (for baselines) and the *Falsif./Identif.* columns (for TRACE) of Table 1; the other three columns correspond to the columns of Table 6. Entries are intervals for env.success percentages.

Method	Rec.	20Q	EDA	Mmind	Haiku	Tanka
Random	[2, 11]	[32, 51]	[65, 82]	[2, 10]	[3, 14]	[0, 4]
DirectLLM	[11, 26]	[7, 20]	[34, 53]	[9, 23]	[13, 29]	[2, 10]
Reflexion	[18, 35]	[6, 19]	[45, 64]	[5, 16]	[1, 7]	[0, 5]
Self-Refine	[14, 30]	[64, 81]	[69, 85]	[21, 39]	[9, 22]	[0, 4]
TRACE-F	[41, 61]	[76, 90]	[79, 92]	[44, 63]	[44, 63]	[9, 22]
TRACE-I	[83, 94]	[96, 100]	[95, 100]	[95, 100]	[96, 100]	[96, 100]

Table 8: Wilson 95% confidence intervals for the MASTERMIND fixed- K H-sweep in Table 2. Entries are intervals for env.success percentages.

n_{colors}	$ \mathcal{H} $	TRACE-F CI	TRACE-I CI
3	9	[96, 100]	[96, 100]
6	18	[96, 100]	[96, 100]
10	30	[93, 99]	[96, 100]
15	45	[80, 93]	[96, 100]
20	60	[65, 82]	[95, 100]

Table 9: Wilson 95% confidence intervals for the same-decoder structured comparison in Table 3. Entries are intervals for env.success percentages.

Configuration	RecMovie	20Q	EDA THINGS
TRACE-I	[83, 94]	[96, 100]	[95, 100]
Direct-Commit (best variant)	[16, 32]	[96, 100]	[96, 100]

Table 10: Wilson 95% confidence intervals for the binary and identity oracle baselines in the *Falsif.* and *Identif.* columns of Table 1. Entries are intervals for env.success percentages.

Method	Rec.	20Q	EDA
<i>Group A: binary oracle</i>			
DirectLLM	[19, 36]	[34, 53]	[76, 90]
Reflexion	[26, 45]	[21, 39]	[59, 77]
Self-Refine	[18, 35]	[13, 28]	[31, 50]
TRACE-F	[41, 61]	[76, 90]	[79, 92]
<i>Group B: identity oracle</i>			
DirectLLM	[31, 50]	[96, 100]	[96, 100]
Reflexion	[33, 52]	[96, 100]	[96, 100]
Self-Refine	[24, 42]	[96, 100]	[96, 100]
TRACE-I	[83, 94]	[96, 100]	[95, 100]

C.3 FULL RECMOVIE RESULTS

Soft-utility breakdown. Table 11 extends the main RecMovie comparison with a soft-utility breakdown of the final recommendation list. The soft utility U_{final} of a list is the mean, over its three recommended titles, of the fraction of canonical C^* predicates satisfied by each title; soft regret is $1 - U_{\text{final}}$. The **final env.** column reproduces the headline final-output success used in the main tables.

Table 11: RecMovie final-output soft-score / soft-regret over 100 seeds. U_{final} is mean partial-credit utility and $1 - U_{\text{final}}$ is soft regret. The **final env.** column matches the headline RecMovie numbers in the main tables.

Method	final env.	U_{final}	soft regret
TRACE-F	51%	0.641	0.359
TRACE-I	90%	0.831	0.169
Direct-AND	15%	0.496	0.504
Direct-OR	23%	0.528	0.472
Direct-RANKED	22%	0.527	0.473
Random	5%	0.328	0.672

Interaction-budget any-success. For completeness, the alternative interaction-budget any-success diagnostic (the fraction of seeds on which any tested recommendation list during the run was accepted by the env, regardless of whether the algorithm subsequently moved past it) gives TRACE-F: 57% and TRACE-I: 95% on the same 100 seeds; LLM baselines and direct-commit variants return on first success, so for them this any-success rate coincides with the final-output rate above.

Proposer ablation. Table 12 fixes the TRACE commit/reject scaffold and varies how candidate identities are proposed. The uniform proposer realizes the falsification setting; Incremental-LM extracts predicate candidates from each feedback message with an LLM; the decoded proposer uses the task’s canonical feedback templates.

Detector ablation. Table 13 compares detector implementations on RecMovie while fixing the algorithm to TRACE-F. The rule-based detector is stronger in this semi-templated setting.

Table 12: Proposer variants inside TRACE on RecMovie with the same rule-based detector; entries are interaction-budget any-success.

Proposer	any-success
TRACE-F (uniform)	57%
Incremental-LM (atomic per-feedback)	74%
TRACE-I (decoded)	95%

Table 13: Detector variants inside TRACE-F on RecMovie. *Overall*: any-success. *Final Output*: final-output env.success (matches Table 1).

Detector	Overall	Final Output
Rule-based detector	57%	51%
LLM, zero-shot	21%	12%
LLM, chain-of-thought	21%	8%

Direct-commit variants. The headline same-decoder comparison is reported in Table 3 of the main text. Table 14 separates the three RecMovie direct-commit variants: *AND* over all wanted predicates, *OR* over at least one predicate per dimension, and a *RANKED* top- k match score.

Table 14: Same decoder, with versus without the TRACE test-before-commit scaffold, on RecMovie. Entries are final-output env.success over 100 seeds; for direct-commit baselines, final-output success coincides with any-success since they return on first hit.

Configuration	env.success
TRACE-I	90%
Direct-AND (strict)	15%
Direct-OR (per-dim disjunction)	23%
Direct-RANKED (top- k match score)	22%

Feedback omission. We test an i.i.d. feedback-dropout model in which each feedback message is replaced by the empty string with probability p . Table 15 reports RecMovie results.

False identities. Table 16 isolates false extracted identities from feedback omission. For every decoded feedback predicate, with probability p_{false} we replace it by a uniformly sampled wrong predicate from the same attribute dimension that is not in the seed’s canonical C^* . The TRACE-I variant uses these corrupted identities only in the proposer while keeping the rule-based detector clean; direct baselines accumulate the same corrupted identities without test-before-commit. To capture degradation across the entire run rather than just the converged output, this sweep uses the interaction-budget any-success diagnostic; at $p_{\text{false}} = 0$ it therefore reports the 95% any-success value rather than the 90% final-output headline. As false identities increase, TRACE-I degrades gradually (95% $\rightarrow$ 67% any-success from $p_{\text{false}} = 0$ to 0.5), whereas all direct-commit variants fall to 8%. The direct methods can still have high $\text{superset}(C^*)$ because they accumulate many constraints, including the true ones; the any-success column reveals the cost of unfiltered false positives, while TRACE-I’s test-before-commit filters them out before commitment.

Equal output budgets. All runs use the same 100 RecMovie seeds and report final-output success, i.e., whether the single output returned by a method satisfies the user. The common budget unit is a generated output that the environment evaluates. Under a cap B , TRACE-I spends at most $B - 1$ outputs on feedback-bearing tests and returns one final output, sampled from its deepest committed node, within the cap; each seed and cap is run independently. Iterative baselines stop at their first accepted output and otherwise return their output at the cap. Best-of- N draws B independent generations and is granted an oracle selector that returns a satisfying sample whenever one exists. All baselines receive the environment’s feedback mapped to the same canonical identity projection over the 36 predicates used by the TRACE-I decoder. The baseline configurations of Table 1 were

Table 15: Interaction-budget any-success of TRACE-I on RecMovie under i.i.d. feedback omission, 100 seeds per cell. Performance is flat through 20% omission and degrades gradually afterwards.

Omission rate p	any-success
0%	95%
10%	95%
20%	95%
30%	94%
50%	93%
70%	87%

Table 16: False-identity-only RecMovie sweep over 100 seeds. Each extracted predicate is independently replaced by a wrong same-dimension predicate with probability p_{false} . Success is interaction-budget any-success, unlike the final-output rates in Table 4.

p_{false}	Method	any-succ.	superset(C^*)
0.0	TRACE-I	95%	59%
0.0	Direct-AND	15%	94%
0.0	Direct-OR	23%	94%
0.0	Direct-RANKED	22%	94%
0.1	TRACE-I	90%	54%
0.1	Direct-AND	12%	86%
0.1	Direct-OR	12%	93%
0.1	Direct-RANKED	19%	94%
0.2	TRACE-I	83%	49%
0.2	Direct-AND	8%	78%
0.2	Direct-OR	12%	92%
0.2	Direct-RANKED	16%	93%
0.3	TRACE-I	85%	53%
0.3	Direct-AND	8%	73%
0.3	Direct-OR	8%	90%
0.3	Direct-RANKED	12%	92%
0.5	TRACE-I	67%	38%
0.5	Direct-AND	7%	58%
0.5	Direct-OR	7%	90%
0.5	Direct-RANKED	8%	93%

run only with $T = 20$. To evaluate baselines with 60 outputs we use three variants of the same methods: DirectLLM-W8 and Self-Refine-W8 condition only on the eight most recent interactions, and Reflexion-NoCycle keeps four trials but disables the early termination of a trial that repeats the same recommendation. For these variants, success at $B = 20$ is read from the same 60-output runs as whether the first accepted output occurs within the first 20 outputs. Tables 17 and 18 give the numbers plotted in Figure 2, and Table 19 reports resource use.

Table 17: Final-output success on RecMovie under common output caps (100 seeds; Wilson 95% intervals). The Table 1 configurations were run only with $T = 20$.

Method	$B = 20$	$B = 60$
TRACE-I ($m_{\text{exp}} = 5$)	73% [64, 81]	86% [78, 91]
DirectLLM (Table 1 config.)	40% [31, 50]	n/a
Reflexion (Table 1 config.)	42% [33, 52]	n/a
Self-Refine (Table 1 config.)	32% [24, 42]	n/a
DirectLLM-W8	38% [29, 48]	48% [38, 58]
Reflexion-NoCycle	35% [26, 45]	37% [28, 47]
Self-Refine-W8	37% [28, 47]	42% [33, 52]
Best-of- N (oracle selector)	17% [11, 26]	21% [14, 30]

Table 18: TRACE-I test-window sweep on RecMovie under common output caps (final-output success with Wilson 95% intervals; feedback rounds per episode).

m_{exp}	B	Success	Mean rounds	P90 rounds
1	20	65% [55, 74]	4.43	5
3	20	78% [69, 85]	13.42	19
3	60	77% [68, 84]	14.26	22
5	20	73% [64, 81]	18.61	19
5	60	86% [78, 91]	22.08	27

Table 19: Resource use per seed on RecMovie, averaged over each method’s complete 100-seed logs (baselines: 60-output runs). *Evals*: environment-evaluated outputs. *Gen.* and *Aux.*: LLM calls for candidate generation and for reflection or critique.

Method	Evals	Gen. LLM	Aux. LLM	LLM tokens	Wall (s)
TRACE-I ($m_{\text{exp}} = 5$)	22.1	0	0	0	0.4
TRACE-I ($m_{\text{exp}} = 15$, default)	60.6	0	0	0	0.9
DirectLLM-W8	36.3	36.3	0	21,275	14.4
Reflexion-NoCycle	39.9	39.9	2.0	25,845	14.2
Self-Refine-W8	38.4	38.4	38.0	62,133	42.4
Best-of- N ($N = 60$)	60.0	60.0	0	13,479	16.7

Trusted initial constraints. Requirements that are already known, for example stated in the user’s initial request or verified in an earlier session, can seed the root: TRACE then starts from $C_{v_0} = C_{\text{init}}$, never re-tests these constraints, and searches only for the remaining ones; $C_{\text{init}} = \emptyset$ recovers Algorithm 1. If $C_{\text{init}} \subseteq C^*$, the root is viable because $\mathcal{X}^* \subseteq \text{Region}(C^*) \subseteq \text{Region}(C_{\text{init}})$, and the proofs of Theorems 2 and 3 apply to the residual search, since they only require every committed node, including the root, to be viable. In Table 20, TRACE-I ($m_{\text{exp}} = 5$) receives $\min(2, |C^*|)$ oracle-provided true constraints at initialization (the four seeds with $|C^*| = 1$ receive one). Because these constraints are supplied by the oracle, the gain is an upper bound on the value of verified context.

Table 20: TRACE-I ($m_{\text{exp}} = 5$) on RecMovie with an oracle-trusted root (final-output success with Wilson 95% intervals; mean feedback rounds per episode).

Root	$B = 20$	$B = 60$	Rounds ($B = 20$)	Rounds ($B = 60$)
$C_{\text{init}} = \emptyset$	73% [64, 81]	86% [78, 91]	18.61	22.08
$ C_{\text{init}} \leq 2$, oracle-trusted	90% [83, 94]	91% [84, 95]	10.09	10.77

C.4 FULL 20Q, EDA, MASTERMIND, AND POEM RESULTS

20Q and EDA: coverage versus env.success. Table 21 compares env.success and the $\text{superset}(C^*)$ coverage diagnostic on 20Q and EDA. The theorem directly controls coverage; env.success may also depend on catalog grounding and generator fallback behavior.

Table 21: Coverage diagnostic versus env.success on 20Q and EDA. On sparse EDA, catalog grounding can decouple env.success from coverage; the theorem-aligned diagnostic is $\text{superset}(C^*)$.

Configuration	20Q		EDA THINGS	
	env.succ.	$\text{superset}(C^*)$	env.succ.	$\text{superset}(C^*)$
TRACE-F (uniform)	84%	82%	87%	84%
TRACE-I (decoded)	100%	100%	99%	99%

MASTERMIND: fixed-budget K -sweep. Table 22 varies K at fixed $n_{\text{colors}} = 6$. Since $|\mathcal{H}| = 6K$, this sweep varies K and H jointly. It is therefore not a direct test of the theorem’s logarithmic

K factor; rather, it is a fixed-budget stress test showing how the uniform proposer degrades as the candidate class grows. The complementary H -sweep at fixed $K = 3$ is reported in Table 2 of the main text; on that sweep, $\text{superset}(C^*)$ tracks env.success exactly except at the largest setting $n_{\text{colors}} = 20$, where TRACE-F reaches env.success 74% but $\text{superset}(C^*) = 72\%$ (TRACE-I has both at 99%).

Table 22: Fixed-budget K -sweep on MASTERMIND with $n_{\text{colors}} = 6$ fixed, so $|\mathcal{H}| = 6K$.

K	TRACE-F succ.	Binary attempts	TRACE-I succ.	Identity attempts
2	73%	10.09	99%	5.14
4	35%	13.84	100%	5.56
6	4%	14.95	100%	7.29
8	1%	15.00	100%	9.11
10	0%	15.00	100%	10.98

Poem: open-vocabulary action generation. For LLF-Bench Poem, the action is free-form English text rather than a finite catalog item. We implement (i) a word-bucket generator that composes each line with a proposed syllable count by sampling CMU-dict words from syllable-indexed buckets; (ii) a rule-based detector that regex-parses native feedback of the form “*The line: X has n syllables. It should only have m syllables.*”; and (iii) TRACE-F and TRACE-I variants over line-syllable predicates.

Table 23 reports Haiku and Tanka results. TRACE-I covers C^* in every seed on both forms. TRACE-F degrades from Haiku to Tanka as K and $|\mathcal{H}| = 6K$ increase under the fixed proposal budget.

Table 23: TRACE on LLF-Bench Poem with Haiku and Tanka forms. We use 100 seeds per cell and $m_{\text{exp}} = N_{\text{max}} = 15$. Brackets are Wilson 95% confidence intervals. TRACE-I exploits per-line target syllable counts revealed in the feedback; TRACE-F must search over $|\mathcal{H}| = 6K$ predicates.

Method	Haiku ($K = 3$)		Tanka ($K = 5$)	
	env.succ.	attempts	env.succ.	attempts
TRACE-F	54% [44,63]	12.57	14% [8,22]	14.77
TRACE-I	100% [96,100]	3.80	100% [96,100]	5.81

The Poem experiment shows that TRACE is not limited to catalog recommendation. With a suitable constrained generator, the same commit/reject scaffold applies to open-vocabulary text generation. The experiment is still structured: the feedback contains parseable line-level syllable information. Richer unstructured natural-language feedback, where rule-based extraction is unlikely to suffice, remains an important direction for future work.

C.5 EXAMPLE EXECUTION TRACE ON RECMOVIE

To make the algorithm’s per-round behavior concrete, the listing below shows the full execution trace of TRACE-I on RecMovie seed 16 with the headline default $m_{\text{exp}} = N_{\text{max}} = 15$ (rule-based detector + decoded proposer). The latent constraint set is $C^* = \{\text{type_}=TV \text{ show, year}=80s, \text{genre}=Sport\}$. The trace illustrates three mechanisms central to the framework:

- **Test-before-commit catches a spurious child cheaply.** Attempt $n=1$ proposes the spurious constraint $\text{type_}=movie$; the very first feedback contradicts it (“not all TV shows”) and also reveals two missing constraints (“not exclusively Sport TV shows”; “not from the 80s”), so Viol fires after one round. The child is rejected after consuming only 1 round of the per-test cap $m_{\text{exp}} = 15$.
- **Multi-sample falsification on viable children is silent.** Attempts $n = 2, 3, 4$ each propose a true latent constraint; feedback is consistent with the tested set on all 15 rounds (we elide the 14 duplicate rounds), so each child survives the test window and is committed.
- **Identification access guides the proposer.** The decoded proposer reads the feedback for extracted identities; by attempt $n=2$ the working set W already contains the $\text{type_}=TV \text{ show}$ identity extracted from the attempt 1 feedback, leading to a true child being proposed and committed. Subsequent commits (year=80s, genre=Sport) follow the same pattern.

After 4 attempts and 46 interaction rounds (well under the budget $T = N_{\text{max}} \cdot m_{\text{exp}} = 225$), the deepest committed node $v^\dagger$ satisfies $C_{v^\dagger} = C^*$ and the output sample $\hat{x}$ consists of three Sport TV shows from the 80s, satisfying every latent constraint.

Listing: TRACE-I execution trace on RecMovie seed 16

```
=====
TRACE-I run on RecMovie | seed=16 m_exp=15 N_max=15
=====
C* (latent constraints) : {genre=Sport, type_=TV show, year=80s}
|H| (candidate class)   : 36

-----

Attempt n=1 (parent C_v = {}, depth=0)
Proposed: type_=movie [spurious]
Child C_u = {type_=movie}
s=1 feedback: "The recommendations are not all TV shows. The
suggested titles aren't exclusively Sport TV shows. The
recommendations are not from the 80s." [Viol fired]
-> REJECT (rounds used: 1/15)

-----

Attempt n=2 (parent C_v = {}, depth=0)
Proposed: type_=TV show [true latent]
Child C_u = {type_=TV show}
s=1 feedback: "You've recommended TV shows, which is awesome!
Among the recommended titles, you'll find more than Sport
TV shows. The recommendations are not from the 80s."
... (s=2..15: feedback unchanged, child survives)
-> COMMIT (rounds used: 15/15)
C_v advances to depth 1: {type_=TV show}

-----

Attempt n=3 (parent C_v = {type_=TV show}, depth=1)
Proposed: year=80s [true latent]
Child C_u = {type_=TV show, year=80s}
s=1 feedback: "You've recommended TV shows, which is awesome!
Among the recommended titles, you'll find more than Sport
TV shows. The recommendations are all from the 80s, great!"
... (s=2..15: feedback unchanged, child survives)
-> COMMIT (rounds used: 15/15)
```

```

C_v advances to depth 2: {type_=TV show, year=80s}

-----
Attempt n=4    (parent C_v = {type_=TV show, year=80s}, depth=2)
Proposed:  genre=Sport      [true latent]
Child C_u = {genre=Sport, type_=TV show, year=80s}
  s=1  feedback: "You've recommended TV shows, which is awesome!
            The recommendations are all Sport, nice! The recommendations
            are all from the 80s, great!"
  ... (s=2..15: feedback unchanged, child survives)
-> COMMIT    (rounds used: 15/15)
      C_v advances to depth 3: {genre=Sport, type_=TV show, year=80s}

-----
Final v_dagger: C = {genre=Sport, type_=TV show, year=80s}
Output x_hat:   "Sport i 2'eren", "ZDF Sport Extra", "A Question of
                Sport"
Termination:    empty_region_cap (no more children to test)
Total rounds:   46 / 225          Total attempts:  4 / 15
Coverage (C* subset C_{v_dagger}): True
=====

```

C.6 PROMPTS FOR LLM-BASED COMPONENTS

For reproducibility we record the verbatim prompts used by the LLM-based ablations in Appendix C.3 and by the TRAVELPLANNER runs in Section 5.5. The prompts were written for Qwen3-4B served locally; the DeepSeek-V4-Flash configuration of Section 5.5 sends the same TRAVELPLANNER prompts through the DeepSeek API with thinking mode enabled; the trailing “/no_think” marker disables Qwen3’s chain-of-thought block when not needed.

LMViolationDetector: zero-shot (Table 13, 21%)

```

[system] You are a careful logical judge. Answer strictly as
        instructed. /no_think

[user] A movie recommender has committed to the following
       assumptions about the user's preferences:
       - <predicate 1>
       - <predicate 2>
       ...

       The user responded with this feedback after seeing the
       latest recommendations:
       """<feedback>"""

       QUESTION: Does the feedback CONTRADICT any of the
       committed assumptions?
       - Answer YES if the feedback indicates that at least one
       committed assumption is wrong (e.g. the feedback says
       the recommendations were not of a type/era/genre the
       assumption claimed).
       - Answer NO if the feedback is consistent with the
       assumptions, or only complains about a dimension not
       in the assumptions.

       Answer with a single word: YES or NO. /no_think

```

LMViolationDetector: chain-of-thought (Table 13, 21%)

[system + user same as zero-shot, replacing the final line:]

First, in ONE short sentence, explain your reasoning.
Then on a new line, write ANSWER: YES or ANSWER: NO.
/no_think

IncrementalLMProposer: per-feedback extraction (Table 12, 74%)

[system] You extract structured preferences from feedback.
Follow the output format precisely. /no_think

[user] You are analysing ONE feedback message from a user to a movie recommender.

CRITICAL CONVENTION (RecMovie): feedback of the form
"The recommendations are not X" / "not all X" /
"not from X"
is a CORRECTION meaning the user WANTS X on that dimension. Positive phrasing ("... wonderful!" /
"... great!") also confirms the user wants that value.
Phrase "past few years" refers to the year bucket "recent".

Dimensions you can emit (use EXACTLY these names):

- type_ (values: movie, TV show)
- year (values: 50s, 60s, 70s, 80s, 90s, 2000s, recent)
- genre (values: Action, Comedy, Drama, Thriller, Horror, Sci-Fi, ...)

Feedback:
""<feedback>""

List the user's preferences revealed by this feedback.
Output ONE line per preference in the format
`dim=value`. If no preference is revealed, output nothing. No other text, no prose. /no_think

TravelPlanner plan generator: system prompt (tp_v6_no_cost_prompt, used in Section 5.5)

You are a meticulous travel-planner that produces machine-readable JSON plans.

You will receive:

- A travel query (origin, destination, days, budget).
- Optional committed constraints the user requires.
- Reference information listing the only valid entities (flights, restaurants, attractions, accommodations) you may use.

Output rules (every plan MUST satisfy ALL of these):

1. JSON-only output: a single object {"plan": [...]} with one entry per day. No prose, no markdown.
2. Entity-format rule: for breakfast / lunch / dinner / attraction / accommodation fields, write "Name, City" (Name copied verbatim from reference). Multiple attractions per day separated by semicolons.

3. Travel-day current_city format: "from CityA to CityB" on travel days; the city name only on stationary days.
4. Transportation paradigm: pick ONE paradigm for the trip (Flight or Self-driving) and stick to it. Stationary days MUST use transportation = "-".
5. Closed-circle trip: day 1 starts at the origin city; the final day returns to the origin.
6. Sandbox-only entities: every Name MUST appear verbatim in the reference; on any "from/to" pattern only use city names from the reference.
7. Diversity & accommodation: do not repeat restaurants or attractions; use the same accommodation across consecutive nights; respect the reference's minimum-nights column.
8. Completeness: fill every field; use "-" only when the field genuinely does not apply.
9. Day-1 accommodation: never write "-".
10. Budget is a meaningful target, not an upper bound: prefer balanced choices when budget allows; never exceed the budget; if spending under 50% of budget, reconsider.

The user prompt is rendered by `build_user_prompt` with fields Query / Origin / Destination / Days / People / Budget, the canonicalized Committed block (predicates already accepted by TRACE), the Reference Information (optionally pre-filtered by the committed constraints), and the closing instruction "Respond with only the JSON object: {"plan": [...]}" . Plans are evaluated by the env's built-in commonsense and hard-constraint evaluators; we do not implement an additional commonsense prompt of our own.