# GLaCS: Grounded Constraint-State Projection for Global-Consistency Reasoning

**Anonymous authors**
Paper under double-blind review

## Abstract

Global-consistency reasoning requires an answer supported by a joint assignment that satisfies interdependent constraints. We study how to select such an answer from a fixed candidate set. We propose GLaCS, a grounded constraint-state projection framework that separates candidate generation from structured authorization. GLaCS extracts variables and relations from the input and compiles them into executable state computations and factorized consistency scores. Task adapters use constrained assignment search, shared numerical repair, or deterministic execution to score the fixed candidates. The lowest-energy candidate is selected through a shared scoring and intervention interface. On identical reflection self-consistency pools, GLaCS improves accuracy from 0.965 to 0.995 on arithmetic chains and from 0.658 to 0.775 on logic-grid transfer. On four extended BIG-Bench Hard tasks, LLM-extracted graphs yield 0.941 accuracy, compared with 0.711 for an LLM selector given the same graphs. Component interventions, graph perturbations, and stem-only extraction audits characterize the roles of state repair, constraint fidelity, and candidate coverage in these gains.

## 1 Introduction

Large language models (LLMs) solve arithmetic, logical, and multi-step tasks using intermediate computation. Scratchpads, chain-of-thought (CoT), and least-to-most prompting expose reasoning in text (Nye et al., 2021; Wei et al., 2022; Kojima et al., 2022; Zhou et al., 2022); self-consistency aggregates answers from multiple trajectories (Wang et al., 2023). Yet performance on BIG-Bench and BIG-Bench Hard remains brittle when several relations must jointly define one valid state (Srivastava et al., 2023; Suzgun et al., 2023). Individually plausible steps can yield a globally inconsistent configuration, and a plausible CoT explanation may diverge from the evidence that determines the answer (Dziri et al., 2023; Turpin et al., 2023).

Prior work improves reasoning reliability by expanding, revising, or scoring textual trajectories. Tree- and graph-based methods explore alternative paths (Yao et al., 2023; Besta et al., 2024); reflection and self-refinement regenerate traces from natural-language feedback (Shinn et al., 2023; Madaan et al., 2023); and verifiers or reward models rank completed traces or individual steps (Cobbe et al., 2021; Uesato et al., 2022; Lightman et al., 2024). These methods strengthen search and selection over generated paths. An explicit state provides another useful interface: its variables can be updated jointly under the problem's constraints.

We call a task *globally consistent* when its answer depends on a joint assignment over interdependent variables and must be checked against their relations together. Arithmetic chains, logic-grid assignments, and ordered swaps have this structure. Neurosymbolic methods already translate language into explicit inference structures or solver programs (Creswell et al., 2023; Pan et al., 2023; Olausson et al., 2023), including declarative specifications in SatLM (Ye et al., 2023) and reusable tools in LATM (Cai et al., 2024). Factor graphs and energy-based models supply decomposed compatibility scores and global optimization (Kschischang et al., 2001; LeCun et al., 2006; Bai et al., 2019). We study a specific decision problem within this setting: given an existing candidate pool and an extracted constraint graph, how well can each candidate be supported after the underlying assignments are repaired or completed?

We propose GLaCS, a grounded constraint-state projection framework for finite-candidate global-consistency reasoning. A single graph supplies the grounded factors for all candidates. Task adapters



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

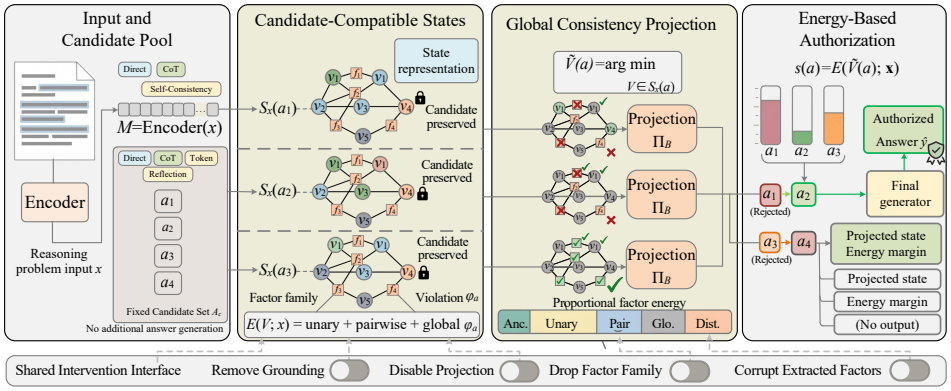


Figure 1: Overview of GLaCS. A normalized constraint graph is adapted into grounded states and factors; task adapters compute a constraint score for each fixed candidate, and the lowest-energy candidate is selected.

compute consistency scores through candidate-restricted search, shared state repair, or execution followed by candidate checks. The lowest-energy candidate is selected while the answer pool stays fixed. This interface applies to provided options and reranking existing model proposals; fixing the pool also isolates selection gains from candidate coverage. The authorization rule and constraint-program interface are shared across tasks, with task-specific executable semantics.

We evaluate GLaCS on arithmetic chains, logic-grid transfer, six BBH order/tracking variants, and four extended BBH tasks. With matched candidates, it improves reflection self-consistency from $0.965$ to $0.995$ on arithmetic chains and from $0.658$ to $0.775$ on logic-grid transfer. Using a text-only Qwen2.5 extractor restricted to JSON constraint graphs (Yang et al., 2024), projection reaches $0.941$ on the four extended tasks, compared with $0.711$ for a 14B selector given the same graphs. On order/tracking tasks, accuracy is $0.857$ with option-visible extraction and $0.758$ with stem-only extraction. These protocols distinguish selection given the full question from extraction without access to answer options.

**Contributions.**

- We formulate fixed-pool authorization through grounded constraint scores, with candidate-restricted optimization and execution-based scoring as concrete task instantiations.
- We instantiate this formulation in GLaCS, with a shared interface for grounded factors, state computation, candidate scoring, and interventions across numerical, assignment, and trajectory states.
- We evaluate authorization under matched candidate pools and learned graphs, using graph-aware selectors, component ablations, and option-free audits to identify the sources and limits of the gains.

## 2 Method

### 2.1 Overview and Problem Setup

GLaCS evaluates fixed candidates using grounded state computations and constraint scores. The adapter determines whether the state is obtained by assignment search, numerical repair, or deterministic execution. Figure 1 summarizes this authorization pipeline.

For example, suppose the clues state that A precedes B and B precedes C, and the query asks who comes first. Under a hard candidate restriction, an initial order $(A, C, B)$ violates one clue; projection can repair it to $(A, B, C)$ without changing the answer. Every B-first order violates the A-before-B clue. With unit violation costs, the two candidates therefore have minimum energies 0 and 1. The score reflects the best compatible completion.

Let $x$ denote a reasoning problem and $\mathcal{A}_x = \{a_n\}_{n=1}^{N_x}$ its finite set of distinct candidate answers. Candidate construction precedes authorization, and $\mathcal{A}_x$ remains fixed throughout selection.

A complete state is $\mathbf{v} = (v_1, \ldots, v_{K_x}) \in \mathcal{X}_x$, where $\mathcal{X}_x$ is the task-specific state space and $K_x$ is the number of state variables. Let $h_x : \mathcal{X}_x \to \mathcal{Y}_x$ be the answer map, where $\mathcal{Y}_x$ is the answer space and $\mathcal{A}_x \subseteq \mathcal{Y}_x$. The states compatible with candidate $a$ are

$$\mathcal{S}_x(a) = \{\mathbf{v} \in \mathcal{X}_x \mid h_x(\mathbf{v}) = a\}. \tag{1}$$

Two representations fit this interface. In a direct answer state, $h_x$ reads the queried variable. In an execution-based adapter, the state is $\mathbf{v} = (\mathbf{u}, c)$, where $\mathbf{u}$ is the task state and $c$ is an explicit candidate claim: $h_x(\mathbf{u}, c) = c$. A query factor measures whether $\mathbf{u}$ supports $c$. Thus, compatibility fixes the candidate being evaluated; its consistency is assessed by energy.

### 2.2 Grounded Constraint Construction

We first convert the input into a representation with explicit objects and relations. This step consists of extraction and normalization:

$$\mathcal{G}_x \equiv \text{Normalize}\left(\text{Extract}(x)\right) = (\mathcal{O}_x, \mathcal{R}_x, q_x, \tau_x), \tag{2}$$

where $\mathcal{O}_x$ contains the resolved entities, objects, quantities, values, or events; $\mathcal{R}_x$ contains the normalized relations, operations, or state updates; $q_x$ is the normalized query; and $\tau_x$ identifies the task family.

We implement $\text{Extract}$ with either a structured parser or an LLM restricted to structured constraint records. Normalization resolves entity references and task-specific relations into a shared outer interface and rejects records that cannot be normalized before projection.

The normalized graph is converted into an executable constraint state through a task adapter:

$$\Gamma_{\tau_x}(\mathcal{G}_x) \longrightarrow (\mathcal{X}_x, \mathcal{F}_x, h_x, \iota_x, \Pi_{\tau_x}), \tag{3}$$

where $\mathcal{X}_x$ is the state space introduced in Section 2.1, $\mathcal{F}_x$ is the grounded factor set, $h_x$ is the state-to-answer map, $\iota_x$ is the candidate-state initializer, and $\Pi_{\tau_x}$ is the task-specific projection procedure associated with task type $\tau_x$. For each candidate with $\mathcal{S}_x(a) \neq \varnothing$, the initializer constructs

$$\mathbf{v}_a^0 = \iota_x(a) \in \mathcal{S}_x(a), \tag{4}$$

The initializer fixes the answer component to the candidate and constructs the remaining variables using the task-specific state constructor.

The shared layer handles graph storage, candidate compatibility, energy aggregation, projection calls, tie breaking, and interventions. Each task adapter supplies its relation vocabulary, state constructor, answer map, factor compiler, and projection operator (Appendix Table 30). These components are reused across instances within a family, including the three-, five-, and seven-object BBH variants. Supporting a new relation vocabulary requires an adapter that defines its executable semantics.

### 2.3 Factorized Energy and Candidate-Preserving Projection

**Factorized energy.** The task adapter represents the constraints in $\mathcal{G}_x$ as $\mathcal{F}_x = \{f_j\}_{j=1}^{J_x}$, with each grounded factor $f_j = (\rho_j, \mathcal{I}_j, \phi_j, w_j)$, where $\rho_j$ is a grounding record containing the normalized relation or operation used by the factor together with its provenance in the input; $\mathcal{I}_j \subseteq \{1, \ldots, K_x\}$ indexes the state variables within its scope; $w_j \geq 0$ is its non-negative weight; and $\phi_j(\mathbf{v}_{\mathcal{I}_j}; \rho_j) \geq 0$ is a violation function.

The consistency energy of a complete state is

$$E(\mathbf{v}; x) = \sum_{j=1}^{J_x} w_j \, \phi_j\left(\mathbf{v}_{\mathcal{I}_j}; \rho_j\right), \tag{5}$$

where $\mathbf{v}_{\mathcal{I}_j}$ denotes the variables within the scope of factor $j$. The dependence on $\rho_j$ links each violation to its normalized constraint and supporting input evidence.

Factors may be unary, pairwise, or higher-order. Anchor factors tie variables to input values. Numerical factors use residual penalties; discrete factors use binary or count-based violations.

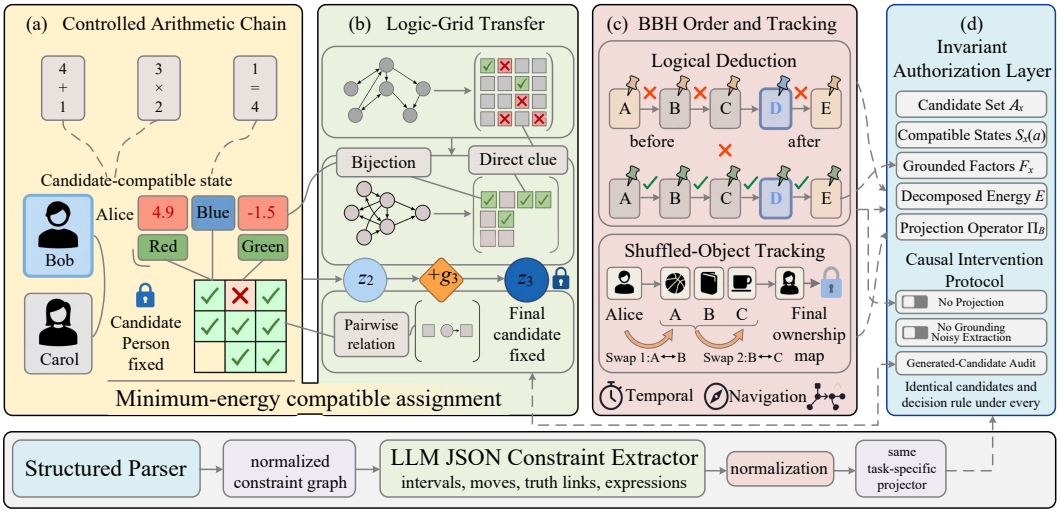


Figure 2: Task-specific instantiations of the shared authorization interface. Different state and factor representations share candidate-preserving projection and energy-based authorization.

**Candidate-preserving projection.** For each candidate with $\mathcal{S}_x(a) \neq \varnothing$, the adapter computes a scored state under a budget $B$:

$$\widetilde{\mathbf{v}}_B(a) = \Pi_{\tau_x,B}\left(\mathbf{v}_a^0; \mathcal{F}_x, \mathcal{S}_x(a)\right), \qquad \widetilde{\mathbf{v}}_B(a) \in \mathcal{S}_x(a). \tag{6}$$

Here, $B$ counts search, update, or simulation operations. All candidates share $\mathcal{F}_x$, and the returned state retains candidate $a$. Adapters can share intermediate computation across candidates.

Exact search minimizes energy over $\mathcal{S}_x(a)$. Shared numerical repair supplies an approximate state for scoring; deterministic adapters execute the graph and evaluate candidate-consistency factors. The concrete procedures are specified below.

$$\begin{aligned} s(a) &= E(\widetilde{\mathbf{v}}_B(a); x), \\ \widehat{y} &= \arg\min_{a \in \mathcal{A}_x} s(a). \end{aligned} \tag{7}$$

For any candidate with $\mathcal{S}_x(a) = \varnothing$, we set $s(a) = +\infty$ and skip initialization and projection. If every compatible set is empty, we record a constraint-construction failure. All candidates are evaluated using the same grounded factor set, projection budget, and scoring rule. Logic-grid and BBH ties retain the base prediction when it is among the minima, then use a canonical candidate key. The controlled-chain runs resolve numeric ties by first occurrence in the deduplicated sample pool.

### 2.4 Task Instantiations

We instantiate this interface with numerical repair, discrete assignment search, and deterministic execution with candidate checks. Figure 2 illustrates how these task-specific states connect to the shared authorization layer.

**Controlled arithmetic chains.** For an arithmetic chain with $T$ grounded operations, the state is

$$\mathbf{z} = (z_0, z_1, \ldots, z_T),$$

where $z_t$ denotes the numerical value after transition $t$. Let $g_0$ be the grounded initial value, $g_t$ the operand at transition $t$, and $o_t \in \{+, \times\}$ the corresponding operation. For this task, $\mathcal{X}_x = \mathcal{X}_x^{\text{chain}}$, and the answer map and candidate-compatible set become

$$h_x(\mathbf{z}) = z_T, \qquad \mathcal{S}_x(a) = \left\{\mathbf{z} \in \mathcal{X}_x^{\text{chain}} \;\middle|\; z_T = a\right\}. \tag{8}$$

The generic factorized energy in Equation 5 becomes

$$E_{\text{chain}}(\mathbf{z}; x) = w_0 \left(z_0 - g_0\right)^2 + \sum_{t=1}^{T} w_t \Big[ \mathbb{I}[o_t = +] \left(z_t - z_{t-1} - g_t\right)^2 + \mathbb{I}[o_t = \times] \left(z_t - z_{t-1} g_t\right)^2 \Big]. \tag{9}$$

Here, $\mathbb{I}[\cdot]$ denotes the indicator function.

The first term anchors the state to the initial value. Each remaining term checks a transition against its normalized operation. The implementation repairs a shared reference chain $\bar{\mathbf{z}}_B$ with two forward sweeps, then scores $\widetilde{\mathbf{z}}_B(a) = (\bar{z}_0, \ldots, \bar{z}_{T-1}, a)$. Each candidate reuses the repaired intermediate values; this is a shared-reference approximation to candidate-restricted minimization.

**Logic-grid reasoning.** A logic-grid state $\pi \in \mathcal{X}_x^{\text{grid}}$ is a complete bijective assignment between input entities and attribute values. The state space enforces bijectivity. From the grounded clue set $\mathcal{C}_x$, the adapter constructs direct, negative, and pairwise clue factors.

The answer map $h_x(\pi)$ evaluates the normalized query on assignment $\pi$. For this task, $\mathcal{X}_x = \mathcal{X}_x^{\text{grid}}$, and the compatible assignment set is

$$\mathcal{S}_x(a) = \left\{ \pi \in \mathcal{X}_x^{\text{grid}} \;\middle|\; h_x(\pi) = a \right\}. \tag{10}$$

For $\tau_x = \text{grid}$, we denote the corresponding budgeted execution of $\Pi_{\tau_x}$ by $\Pi_{\text{grid},B}$. Let $\pi_a^0 = \iota_x(a)$ denote the candidate-compatible initial assignment. The projected assignment is

$$\widetilde{\pi}_B(a) = \Pi_{\text{grid},B}\left(\pi_a^0; \mathcal{F}_x^{\text{grid}}, \mathcal{S}_x(a)\right), \tag{11}$$

where $\mathcal{F}_x^{\text{grid}}$ contains the grounded clue factors. Its projected score is

$$s_{\text{grid}}(a) = \sum_{c \in \mathcal{C}_x} w_c \mathbb{I}\left[\widetilde{\pi}_B(a) \not\models c\right], \qquad w_c \geq 0. \tag{12}$$

The reported logic-grid runs exhaustively enumerate all bijective states, group them by the query answer, and retain each candidate's minimum energy. With $m$ people and $d$ attribute categories, this visits $(m!)^d$ states; the evaluated $m = 4, d = 3$ grids have 13,824 states.

**Object tracking.** For shuffled-object tracking, the extracted initial ownership map $\bar{m}_0$ and ordered swaps $(p_t, r_t)$ determine a rollout:

$$\bar{m}_t = \text{Swap}(\bar{m}_{t-1}; p_t, r_t), \qquad t = 1, \ldots, T_s. \tag{13}$$

The adapter uses a claim-augmented state $(\bar{\mathbf{m}}, a)$ and a unit query-consistency factor:

$$s_{\text{track}}(a) = \mathbb{I}[a \neq \text{Query}(\bar{m}_{T_s}; q_x)], \tag{14}$$

after normalizing candidate and object names. The rollout is shared across candidates. Faithful, complete extraction gives the correct candidate score zero and conflicting candidates score one.

**BBH logical deduction.** The adapter enumerates all object permutations $\pi$. If candidate $a$ asserts an object–position pair, its score is $s(a) = \min_\pi [10 C_x(\pi) + \mathbb{I}[\pi \not\models a]]$, where $C_x$ counts violated ordering constraints. The candidate claim is fixed, and agreement with the permutation is a soft factor. This prioritizes ordering consistency and then tests support for the candidate assertion.

**Other BBH families.** Temporal reasoning checks each candidate interval against day bounds and busy intervals. Navigation, Web of Lies, and Boolean tasks compute a displacement, truth assignment, or expression value, then score candidate agreement. These use the same fixed-pool selection rule with task-specific execution and query factors (Appendix A).

*(a) Generation and selection*

| Method | Ctrl. | Logic |
|---|---|---|
| *Candidate generation* | | |
| Direct | 0.231 | 0.320 |
| CoT | 0.368 | 0.396 |
| CoT-SC | 0.865 | 0.689 |
| Refl-SC | 0.965 | 0.658 |
| *Same Refl-SC pool* | | |
| ToT-vote | 0.866 | 0.691 |
| Self-Refine | 0.771 | 0.377 |
| GLaCS | **0.995** | **0.775** |

*(b) Learned-graph selectors by BBH family*

| Method | Logic | Track | O/T | Temp. | Nav. | Web | Bool. | Ext. |
|---|---|---|---|---|---|---|---|---|
| *Prompting baseline* | | | | | | | | |
| CoT-SC | 0.542 | 0.318 | 0.430 | 0.297 | 0.406 | 0.266 | 0.812 | 0.445 |
| *Learned listwise selectors* | | | | | | | | |
| Listwise (14B) | 0.740 | 0.208 | 0.474 | 0.859 | 0.609 | 0.500 | 0.781 | 0.688 |
| Graph-listwise (14B) | 0.729 | 0.260 | 0.495 | 0.922 | 0.609 | 0.531 | 0.781 | 0.711 |
| *Graph-based methods* | | | | | | | | |
| Direct graph (14B) | 0.755 | 0.885 | 0.820 | *abst.* | 0.875 | 0.891 | 1.000 | 0.691 |
| GLaCS (14B graph) | **0.776** | **0.938** | **0.857** | **0.984** | **0.875** | **0.906** | **1.000** | **0.941** |

Table 1: Main exact-match results. Panel (a) separates candidate generation from selection on shared Refl-SC pools. Panel (b) covers six order/tracking variants (O/T) and four extended tasks (Ext.). GLaCS and Graph-listwise use option-visible extraction; Direct graph derives answers without the original options at execution and counts temporal abstentions as errors. Table 4 reports stem-only extraction.

# 3 Experiments

## 3.1 Experimental Setup

We evaluate GLaCS on controlled arithmetic chains, logic-grid transfer, and BIG-Bench Hard (BBH). The procedural testbeds probe mutable-state repair under numeric and ordering transformations and fixed-pool authorization under four logic-grid transformations. The main benchmark includes six BBH variants—logical deduction and tracking shuffled objects with three, five, or seven objects—and four additional tasks: temporal sequences, navigate, web of lies, and Boolean expressions (Srivastava et al., 2023; Suzgun et al., 2023). Appendix boundary audits cover ASDiv-A, DROP, GSM8K, MATH-500, MultiArith, ProofWriter, and SVAMP (Miao et al., 2020; Dua et al., 2019; Cobbe et al., 2021; Hendrycks et al., 2021; Roy & Roth, 2015; Tafjord et al., 2021; Patel et al., 2021).

Qwen2.5-VL-3B-Instruct generates controlled and logic-grid candidates from text-only inputs (Bai et al., 2025). Qwen2.5-3B/7B/14B-Instruct extracts BBH graphs under a JSON-only output constraint (Yang et al., 2024). Controlled and logic-grid selectors rerank generated answers; fixed-option BBH selectors score the full provided option set. Within each comparison, selectors receive identical candidate pools. Baselines include Direct, CoT (Wei et al., 2022), CoT-SC (Wang et al., 2023), reflection self-consistency, ToT-style voting (Yao et al., 2023), Self-Refine (Madaan et al., 2023), and Qwen2.5-14B listwise selectors. We report exact match, paired significance tests, confidence intervals, and computational cost. A backbone-refresh audit uses Qwen3.5-2B/4B/9B (Team, 2026) for BBH generation and extraction with matched non-thinking decoding. Appendices B.5 and A give the full protocols.

**Projection procedures.** We use the task-specific computations in Section 2.4. Appendix Table 5 records each solver's search extent and budget.

*(a) Accuracy ablations*

| | Refl-SC pool | | 14B graph | |
|---|---|---|---|---|
| Variant | Ctrl. | Logic | O/T | Ext. |
| Full GLaCS | **0.995** | **0.775** | **0.857** | **0.941** |
| No projection | 0.808 | 0.621 | 0.430 | 0.488 |
| No grounding | 0.845 | 0.650 | 0.432 | 0.543 |
| Drop core factors | 0.864 | 0.650 | 0.370 | 0.500 |

*(b) Preservation semantics*

| Metric | Preserving | Unconstr. |
|---|---|---|
| Exact (C/L) | 0.995/0.775 | 0.992/1.000 |
| Identity change (C/L) | 0/0 | 0.305/0.370 |
| Collapse (C/L) | 0/0 | 0.366/0.299 |
| Tie (C/L) | –/0 | 0.366/0.299 |
| Valid output (C/L) | 1.000/1.000 | 0.987/0.775 |

*Preserving*: fixed-pool authorization; *Unconstr.*: unrestricted solving. Logic-grid and BBH predictions are candidate-order invariant under the order-independent tie rule; see Appendix Table 6.

Table 2: Component and candidate-preservation ablations. Panel (a) reports exact match on Refl-SC pools and Qwen2.5-14B BBH graphs. Panel (b) audits preservation semantics; C/L denotes Controlled/Logic-grid.

| Scope | Ext. | CoT-SC | Proj. | Gain | JSON | Graph |
|---|---|---|---|---|---|---|
| Core six | 2B | 0.701 | 0.870 | +0.169 | 1.000 | 0.945 |
| Core six | 4B | 0.701 | 0.940 | +0.240 | 1.000 | 0.990 |
| Core six | 9B | 0.701 | **0.945** | **+0.245** | 1.000 | 0.997 |
| Extended | 9B | 0.668 | **0.957** | **+0.289** | 1.000 | 0.988 |

Table 3: Learned text-to-graph extraction with Qwen3.5 models on fixed Qwen3.5-2B CoT-SC candidate pools, with 64 examples per task family. Ext. denotes extractor size. Scaling the extractor improves graph validity and exact match without changing candidate generation.

### 3.2 Main Results

**Fixed-Pool Candidate Authorization.** Table 1 compares selectors on shared candidate pools. On Refl-SC pools, GLaCS improves exact match from 0.965 to 0.995 on controlled chains and from 0.658 to 0.775 on logic-grid transfer. Every matched selector receives the same candidates in the same order, isolating the effect of authorization. Against Refl-SC, GLaCS records $91/0$ and $60/0$ wins/losses in the two regimes.

**Comparison with Strong Selectors.** GLaCS also outperforms generative and listwise selectors. On extended BBH, a Qwen2.5-14B listwise verifier reaches 0.688. Giving it the same extracted JSON graph raises accuracy to 0.711, while factorized projection reaches 0.941. The corresponding O/T results are 0.474, 0.495, and 0.857. These comparisons measure the contribution of executable constraint scoring beyond access to the extracted graph.

**Backbone Refresh.** The same pattern holds with Qwen3.5 (Table 3). With a fixed Qwen3.5-2B candidate pool, larger text-only extractors improve graph validity and end-to-end projection accuracy on the six core and four extended BBH families. On the extended families, projection reaches 0.957; Qwen3.5-9B listwise and graph-listwise selectors both reach 0.742. Stem-only graph-induced authorization is within 0.005 of visible-option extraction. The hidden-option extended audit reaches $0.936 \pm 0.018$ over three seeds (Appendix Tables 28 and 29).

### 3.3 Analysis and Ablations

**Learned Extraction and Option-Free Generalization.** Projection gains depend on extraction quality (Table 4). The 3B extractor achieves 0.633 graph success and performs slightly below CoT-SC. The 7B and 14B extractors exceed 0.99 graph success and reach 0.797 and 0.857 exact match. Semantic fidelity explains this difference: JSON validity is 1.000 at every scale, while factor F1 rises from 0.699 to 0.850 and 0.879, and exact graph match rises from 0.163 to 0.438 and 0.675. Missing factors and entity/query mismatches account for most residual errors (Appendix Table 34).

Stem-only extraction retains 0.941 accuracy on the extended families. On O/T, hiding options reduces fixed-option projection from 0.857 to 0.758, while graph validity remains near 0.995. Option visibility therefore affects the extracted representation beyond JSON or graph validity. With both extraction

*(a) Extractor scaling*

| Source | Graph success | Exact | Δ |
|---|---|---|---|
| CoT-SC | – | 0.430 | – |
| Qwen2.5-3B graph | 0.633 | 0.424 | -0.006 |
| Qwen2.5-7B graph | 0.992 | 0.797 | +0.367 |
| Qwen2.5-14B graph | 0.997 | **0.857** | +0.427 |

*(b) Option-free audit*

| Setting | Exact | Graph success | Gold coverage |
|---|---|---|---|
| Fixed options, full graph | 0.857 | 0.997 | 1.000 |
| Fixed options, stem-only graph | 0.758 | 0.995 | 1.000 |
| Graph-induced candidates | 0.737 | 0.995 | 0.997 |

Table 4: Learned-extraction audits on 384 BBH order/tracking examples. Panel (a) scales the text-only extractor. Panel (b) compares extraction from the full input with stem-only extraction and candidate induction.

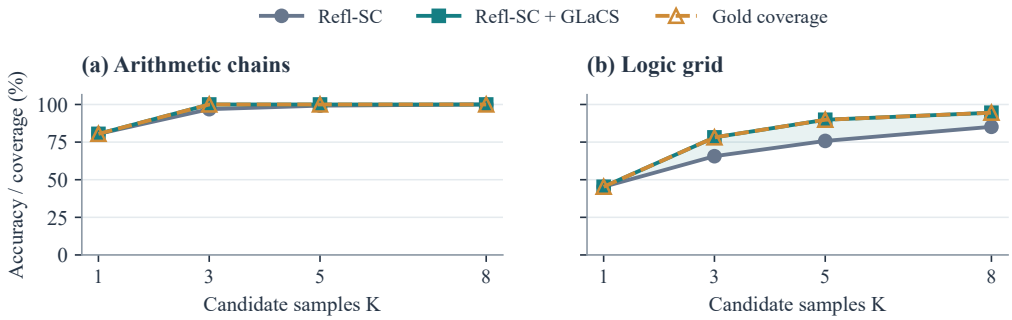


Figure 3: Candidate-budget sensitivity using prefixes of one frozen eight-sample Refl-SC run (128 examples per task). Shading marks the accuracy gain over Refl-SC; GLaCS coincides with gold coverage. Exact values and computational costs are in Appendix Table 16.

and candidate construction restricted to the stem, accuracy is 0.737 and gold coverage is 0.997. These stem-only runs measure performance without answer-option information during extraction.

**Direct Structured Execution.** The direct graph audit reaches 0.820 on O/T with original option labels hidden at execution. On navigation, Web of Lies, and Boolean expressions, direct execution averages 0.922, compared with 0.927 for GLaCS. The Extended aggregate gap (0.691 versus 0.941) is driven mainly by temporal graphs, which describe feasible intervals: the direct executor abstains when they do not identify a unique answer interval, while candidate-conditioned evaluation tests the supplied choices. Direct execution is thus useful when the graph determines an answer; fixed-pool authorization addresses choosing among available proposals (Appendix Tables 47 and 48).

**Core Components and Causal Grounding.** Projection, grounding, and task-specific factors each contribute to accuracy. On Controlled@Refl-SC, removing them reduces exact match from 0.995 to 0.808, 0.845, and 0.864, respectively. Logic-grid tasks and learned BBH graphs show comparable degradation (Table 2). Factor interventions also reveal candidate-specific effects: deleting a candidate-matched factor changes hard-negative energy by 0.545 on average, compared with 0.261 for unrelated factors. The paired difference is 0.283 (bootstrap 95% CI: $[0.146, 0.464]$; Appendix Figure 7). A generic constraint-program interpreter matches specialized decisions on 448 audited examples using task-specific translators (Appendix Table 31).

**Candidate Preservation and Coverage.** The correct answer appears in 397 of 512 logic-grid pools, and GLaCS selects it in all 397 cases, each with a unique minimum energy. The mean margin over the lowest-energy wrong candidate is 0.909; all 115 remaining errors arise from missing gold candidates. Unrestricted solving reaches 1.000 accuracy by allowing answers outside the pool (Table 2). In this regime, improving end-to-end accuracy requires increasing candidate coverage or allowing new answers. Across candidate sources and sampling budgets, projection similarly tracks the coverage ceiling (Figure 3 and Appendix Tables 15–16).

For learned BBH graphs, minimum-energy ties occur in 20.6% of O/T examples and 0.8% of Extended examples. Reapplying the stated tie rule to cached scores retains the base prediction on 17.4% and 0.4% of all examples, respectively. Unique-minimum cases have accuracies 0.954 and 0.945; Appendix Table 7 reports the full decomposition.

**Robustness.** Across four transformations in the three-seed logic-grid audit, GLaCS improves Refl-SC by $0.115$–$0.132$; no-projection and drop-pairwise variants remain near the base selector (Appendix Table 32). Stem-only BBH accuracy is $0.930 \pm 0.010$ across extraction seeds. Logic-grid and BBH predictions are invariant to 20 candidate-order permutations; scaling each factor family by $0.5$ or $2$ preserves accuracy in all regimes (Appendix Tables 6 and 8).

Performance declines gradually under graph corruption and remains above CoT-SC with moderate missing- or spurious-factor noise (Appendix Table 45). Semantic negative controls isolate the role of input alignment: within-task shuffled graphs retain 0.997/0.996 validity, yet O/T and Extended accuracy fall to $0.408 \pm 0.012$ and $0.452 \pm 0.034$, close to empty-graph controls. These results link the gains to input-matched constraints (Appendix Table 46).

**Computational Cost.** On 512 controlled-chain examples, reranking reuses generated candidates and adds 0.335 seconds of CPU time. Full-pipeline timing also accounts for generation and graph extraction (Appendix Table 14). For BBH O/T and Extended, graph-based projection uses five LLM calls per example, compared with six for graph-listwise selection; CPU projection takes 5.11 and 0.07 milliseconds per example, respectively. Corresponding total wall times are 2880.6/1591.4 seconds for projection and 3016.0/1685.9 seconds for graph-listwise selection. These serial timing runs use separate accuracy measurements. Logic-grid projection reuses the same generation calls and tokens, adding 51.68 milliseconds per example for exhaustive assignment search.

## 4 Related Work

**Chain-of-thought and self-consistency.** CoT and self-consistency generate and aggregate textual trajectories (Wei et al., 2022; Wang et al., 2023). Verifiers, Tree-of-Thought, Reflexion, and Self-Refine add selection, search, or revision (Cobbe et al., 2021; Yao et al., 2023; Shinn et al., 2023; Madaan et al., 2023). GLaCS scores candidates from these generators using projected constraint energy.

**Latent CoT and continuous thought.** Coconut, CODI, SoftCoT++, and PCCoT perform reasoning in continuous hidden states (Hao et al., 2024; Shen et al., 2025; Xu et al., 2025; Wu et al., 2025). GLaCS uses input-anchored variables and factors that support explicit constraint interventions.

**Neurosymbolic and solver-mediated reasoning.** Selection–inference, Logic-LM, and LINC translate language into symbolic inference or solver programs (Creswell et al., 2023; Pan et al., 2023; Olausson et al., 2023). SatLM generates declarative specifications for an automated theorem prover (Ye et al., 2023); LATM constructs reusable Python tools for related requests (Cai et al., 2024). GLaCS builds on this separation of language understanding and execution to study minimum-violation selection over a fixed candidate pool. Its adapters optimize assignments or execute state updates, then score each candidate under shared constraints. This candidate-conditioned use of constraint optimization and factorized energies (Kschischang et al., 2001; LeCun et al., 2006) is evaluated with matched pools, same-graph selectors, and direct execution.

## 5 Conclusion

GLaCS separates candidate generation from authorization through grounded state computation and factorized candidate scores. It improves fixed-pool selection on controlled, logic-grid, and BBH tasks, with residual errors tied to candidate coverage or graph quality. Ablations, paired comparisons, and 397/397 conditional correctness support the contribution of constraint-based selection. These results support constraint-state authorization for finite-candidate global-consistency reasoning.

## AI USE STATEMENT

Generative AI tools were used solely to assist with and polish the writing of this manuscript. The authors reviewed and revised all AI-assisted content and take full responsibility for the final manuscript.

## REPRODUCIBILITY STATEMENT

The paper specifies the finite candidate pools, grounded constraint graphs, factorized energies, projection budgets, tie-breaking rules, prompts, decoding settings, seeds, and evaluation tasks. Appendix A contains the reproducibility and evaluation protocols, including the Qwen2.5-14B extraction audits and candidate-construction details. The repository contains the manuscript source, figures, tables, and references; it does not include the complete experiment runner, model checkpoints, API snapshots, or all generated candidate records. Exact replication therefore requires the stated model versions, public task data, implementation artifacts, and inference settings.

## REFERENCES


Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In *NeurIPS*, pp. 688–699, 2019.

Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, Humen Zhong, Yuanzhi Zhu, Ming-Hsuan Yang, Zhaohai Li, Jianqiang Wan, Pengfei Wang, Wei Ding, Zheren Fu, Yiheng Xu, Jiabo Ye, Xi Zhang, Tianbao Xie, Zesen Cheng, Hang Zhang, Zhibo Yang, Haiyang Xu, and Junyang Lin. Qwen2.5-vl technical report. *CoRR*, abs/2502.13923, 2025.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. Graph of thoughts: Solving elaborate problems with large language models. In *AAAI*, pp. 17682–17690. AAAI Press, 2024.

Tianle Cai, Xuezhi Wang, Tengyu Ma, Xinyun Chen, and Denny Zhou. Large language models as tool makers. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/ed91353f700d113e5d848c7e04a858b0-Abstract-Conference.html.

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

Antonia Creswell, Murray Shanahan, and Irina Higgins. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *ICLR*. OpenReview.net, 2023.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 2368–2378, 2019.

Nouha Dziri, Ximing Lu, Melanie Sclar, Xiang Lorraine Li, Liwei Jiang, Bill Yuchen Lin, Sean Welleck, Peter West, Chandra Bhagavatula, Ronan Le Bras, et al. Faith and fate: Limits of transformers on compositionality. *Advances in neural information processing systems*, 36:70293–70332, 2023.

Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *CoRR*, abs/2412.06769, 2024.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *NeurIPS Datasets and Benchmarks*, 2021.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35: 22199–22213, 2022.

Frank R Kschischang, Brendan J Frey, and H-A Loeliger. Factor graphs and the sum-product algorithm. *IEEE Transactions on information theory*, 47(2):498–519, 2001.

Yann LeCun, Sumit Chopra, Raia Hadsell, M Ranzato, Fujie Huang, et al. A tutorial on energy-based learning. *Predicting structured data*, 1(0), 2006.

Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *International Conference on Learning Representations*, volume 2024, pp. 39578–39601, 2024.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in neural information processing systems*, 36:46534–46594, 2023.

Shen-Yun Miao, Chao-Chun Liang, and Keh-Yih Su. A diverse corpus for evaluating and developing english math word problem solvers. In *Proceedings of the 58th annual meeting of the Association for Computational Linguistics*, pp. 975–984, 2020.

Maxwell I. Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. Show your work: Scratchpads for intermediate computation with language models. *CoRR*, abs/2112.00114, 2021.

Theo Olausson, Alex Gu, Ben Lipkin, Cedegao Zhang, Armando Solar-Lezama, Joshua Tenenbaum, and Roger Levy. Linc: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5153–5176, 2023.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. Logic-lm: Empowering large language models with symbolic solvers for faithful logical reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 3806–3824, 2023.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. Are nlp models really able to solve simple math word problems? In *Proceedings of the 2021 conference of the North American chapter of the association for computational linguistics: human language technologies*, pp. 2080–2094, 2021.

Subhro Roy and Dan Roth. Solving general arithmetic word problems. In *Proceedings of the 2015 conference on empirical methods in natural language processing*, pp. 1743–1752, 2015.

Zhenyi Shen, Hanqi Yan, Linhai Zhang, Zhanghao Hu, Yali Du, and Yulan He. Codi: Compressing chain-of-thought into continuous space via self-distillation. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 677–693, 2025.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: language agents with verbal reinforcement learning. In *NeurIPS*, 2023.

Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *Transactions on machine learning research*, 2023.

Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed H Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 13003–13051, 2023.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. Proofwriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 3621–3634, 2021.

Qwen Team. Qwen3. 5: Accelerating productivity with native multimodal agents, february 2026, 2026.

Miles Turpin, Julian Michael, Ethan Perez, and Samuel Bowman. Language models don't always say what they think: Unfaithful explanations in chain-of-thought prompting. *Advances in Neural Information Processing Systems*, 36:74952–74965, 2023.

Jonathan Uesato, Nate Kushman, Ramana Kumar, H. Francis Song, Noah Y. Siegel, Lisa Wang, Antonia Creswell, Geoffrey Irving, and Irina Higgins. Solving math word problems with process- and outcome-based feedback. *CoRR*, abs/2211.14275, 2022.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V. Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *ICLR*. OpenReview.net, 2023.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Haoyi Wu, Zhihao Teng, and Kewei Tu. Parallel continuous chain-of-thought with jacobi iteration. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 914–926, 2025.

Yige Xu, Xu Guo, Zhiwei Zeng, and Chunyan Miao. Softcot++: Test-time scaling with soft chain-of-thought reasoning. *CoRR*, abs/2505.11484, 2025.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report. *CoRR*, abs/2412.15115, 2024. URL https://arxiv.org/abs/2412.15115.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36:11809–11822, 2023.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. SatLM: Satisfiability-aided language models using declarative prompting. In *Advances in Neural Information Processing Systems*, volume 36, pp. 45548–45580, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/8e9c7d4a48bdac81a58f983a64aaf42b-Abstract-Conference.html.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc V Le, et al. Least-to-most prompting enables complex reasoning in large language models. In *The eleventh international conference on learning representations*, 2022.

## A Reproducibility and Evaluation Protocols

This appendix specifies candidate construction, extraction, state computation, and evaluation. Each selection run assigns constraint scores to a finite candidate set $\mathcal{A}_x$. Logic-grid and BBH ties retain the base prediction when it is among the minima, then use a normalized candidate key; controlled-chain numeric ties use first occurrence in the sample pool.

**Factor weights and projection budgets.** Controlled-chain energies use normalized squared residuals with anchor/add/mul weights $0.5/1.0/1.0$, value scale $1000$, and projection step size $1.0$. The controlled-chain projection budget is two solver steps for the learned chain model; the projection-step ablations sweep $B = 0, \ldots, 5$, with $B = 0$ implemented as the exact no-projection endpoint. Logic-grid and BBH discrete projectors use unit 0/1 violation counts unless noted otherwise. For logical deduction, option scoring uses $10\times$ the number of violated grounded ordering constraints plus a unit option-consistency violation. This lexicographically prioritizes ordering consistency over candidate agreement. Tracking, navigation, Web of Lies, and Boolean execution use unit candidate-consistency violations. A temporal candidate scores zero if it lies within the day and overlaps no busy interval; otherwise it scores one plus the number of overlaps. Per-task weights are fixed independently of validation labels. All runs select from the fixed pool, with the optional proximity coefficient fixed at $\lambda = 0$.

**Candidate compatibility and tie breaking.** Logic-grid search restricts the queried answer to each candidate. BBH adapters retain the candidate claim and score its consistency with an optimized or executed task state. Controlled-chain reranking repairs a shared reference chain, writes each numeric candidate into the final slot, and scores the resulting state. Its repaired intermediates are shared across candidates. Numeric candidates are rounded to integers and deduplicated in first-seen order. Logic-grid candidates are valid person names; graph-induced BBH candidates use normalized surface forms.

| Family | State computation and candidate score | Search extent / budget |
|---|---|---|
| Controlled chain | Repair a shared reference, substitute the candidate final value, evaluate residual energy | Two forward sweeps; shared-reference approximation |
| Logic grid | Minimize clue violations with the queried answer fixed | Exhaustive $(m!)^d$ states; $m = 4, d = 3$ |
| BBH logical order | Minimize $10C_x(\pi) + \mathbb{I}[\pi \not\models a]$ over permutations | Exhaustive $m!$ permutations per candidate; $m = 3, 5, 7$ |
| Tracking / navigation | Execute swaps or moves; score candidate agreement | One pass through the ordered events |
| Temporal | Check day bounds and count busy-interval overlaps | All extracted busy intervals per candidate |
| Web of Lies | Propagate ordered truth links; score candidate agreement | One pass through the ordered links |
| Boolean | Evaluate the expression; score candidate agreement | One expression evaluation |

Table 5: Implemented state computations. The two exhaustive searches return exact minima for their stated scores without truncation. Execution-based rows evaluate the normalized graph directly; their correctness depends on the extracted records.

**Prompting and decoding.** Controlled-chain data use $512$ examples per seed for the main compute run, seeds $4121, 5121, 6121$ for the three-seed statistics, 1–5 chain steps, three distractors, and three CoT-SC samples. Logic-grid transfer uses seed $7103$, $128$ examples per variant, grid size $4$, pairwise depth $2$, and three distractors. The appendix logic-grid stability audit uses seeds $7103, 7203, 7303$, $512$ examples per variant, and only the reflection self-consistency (Refl-SC) candidate pool. Direct and single CoT-style generations use temperature $0$ and top-$p = 1.0$; self-consistency variants use temperature $0.7$, top-$p = 0.95$, and three samples unless a table states otherwise. The open-source-style listwise verifier and Self-Refine baselines use one greedy Qwen call over the same Reflection-SC candidate set. For the extended BBH same-graph verifier audit, Qwen2.5-14B-Instruct receives the

problem stem and answer options in the listwise condition, and additionally receives the Qwen2.5-14B extracted JSON graph in the graph-listwise condition. Both variants select their answers by greedy decoding with a maximum of 256 new tokens.

We use separate studies to evaluate breadth and stability. The controlled study uses three 512-example seeds. The seed-7103 logic-grid study uses 128 examples per transformation to support the full prompt-family sweep, while the larger three-seed audit fixes the Refl-SC pool and uses 512 examples per transformation. BBH uses 64 examples per public slice to match the fixed generation budget across families; structured projectors were additionally checked on all 250 available examples per extended family.

**Qwen3.5 backbone-refresh protocol.** We also evaluate the candidate-generation interface with Qwen3.5-2B, Qwen3.5-4B, and Qwen3.5-9B under Transformers 5. Native Qwen3.5 thinking is disabled, and the existing Refl-SC and CoT-SC prompts control reasoning. The logic-grid refresh uses seed 7103, 128 examples for each of the four transformations, three Refl-SC samples, temperature $0.7$, top-$p = 0.95$, and at most 1024 new tokens. The BBH refresh uses four concise CoT-SC samples per example, seeds $4203$ (O/T) and $9301$ (Extended), and 64 examples per family. A six-family parsing-validity pilot selects the first token ceiling for which every family has aggregate parse rate at least $0.875$ and sample validity at least $0.85$. The passing ceiling is 1536 tokens and is frozen before the formal BBH run. Terminal normalization accepts a multiple-choice label or Yes/No-style answer only when it is the complete final non-empty line.

**JSON extractors.** The BBH weak-extractor experiments use Qwen2.5-Instruct as a JSON graph extractor with one greedy sample, temperature $0$, top-$p = 1.0$, and maximum 512 new tokens. The fixed-option scaling table uses seed $9301$ and 64 examples for each of BBH Logic-3/5/7 and Track-3/5/7. The extended extractor table uses the same seed and 64 examples each for temporal sequences, navigation, web-of-lies, and Boolean expressions. The extractor prompt asks only for a constraint graph: logical-deduction JSON contains objects plus fixed/relative constraints; tracking JSON contains initial assignments, ordered swaps, and the queried participant; temporal JSON contains wake/close times and busy intervals; navigation JSON contains the orientation rule and ordered movement events; web-of-lies JSON contains base truth facts, ordered claims, and the queried person; Boolean JSON contains the expression before the word "is." Extracted names, times, moves, truth links, and expressions are normalized before scoring. Invalid JSON produces no projection result and counts as an error. Missing task-core fields are recorded as graph failures; when the adapter returns tied candidate scores, the same tie rule applies.

**Stem-only audit.** The stem-only generated-candidate audit uses Qwen2.5-14B-Instruct and seed $9501$. The extractor prompt contains only the problem stem, with answer options, option labels, and candidate names removed. Candidates are induced from the normalized graph as

$$\mathcal{A}_x^{\mathrm{ind}} = \mathcal{C}_{\tau_x}(\mathcal{O}_x, \mathcal{R}_x, q_x),$$

where $\mathcal{C}_{\tau_x}$ is the task-specific constructor for task type $\tau_x$. For logical deduction it returns resolved entities or queried positions in the ordering graph; for tracking it returns resolved owners, participants, or objects that can answer $q_x$. Duplicate surface forms are merged by canonicalization, and an empty set counts as an extraction failure. The candidate set comes directly from the extracted graph and deterministic graph hints, then passes through the same compatibility, projection, energy, and authorization rules. The fixed-option reference reported in the main stem-only audit uses the same stem-only extracted graph but projects over the original option set.

**Intervention definitions.** Every intervention preserves the candidate pool, candidate order, tie-breaking rule, and final authorization procedure. For the numerical and logic-grid state ablations, the no-projection score is

$$s_{\text{no-proj}}(a) = E(V_a^0; x), \tag{15}$$

which scores the initial candidate state directly. BBH no-projection rows use the stored CoT-SC base prediction, disabling structured execution and scoring. A no-grounding intervention removes or reassigns the evidence anchors $\rho_j$ while retaining factor scopes and projection. For factor family $r$, the retained set is

$$F_x^{(-r)} = \{f_j \in F_x \mid \tau_j \neq r\}, \tag{16}$$

where $\tau_j$ is the family label of factor $f_j$. The resulting differences isolate projection, grounding, or the designated factor family on a fixed candidate pool.

### A.1 Candidate-Order and Factor-Weight Audits

| Suite | Variant | Exact | Agreement | Tie rate |
|---|---|---|---|---|
| Logic grid | Full | $0.775 \pm 0.000$ | 1.000 | 0.030 |
| | No projection | $0.645 \pm 0.000$ | 1.000 | 0.147 |
| | No grounding | $0.658 \pm 0.000$ | 1.000 | 0.330 |
| | Drop core factors | $0.658 \pm 0.000$ | 1.000 | 0.330 |
| BBH O/T | Full | $0.857 \pm 0.000$ | 1.000 | 0.206 |
| | No projection | $0.430 \pm 0.000$ | 1.000 | – |
| | No grounding | $0.432 \pm 0.000$ | 1.000 | 1.000 |
| | Drop core factors | $0.370 \pm 0.000$ | 1.000 | 0.560 |
| BBH Extended | Full | $0.941 \pm 0.000$ | 1.000 | 0.008 |
| | No projection | $0.488 \pm 0.000$ | 1.000 | – |
| | No grounding | $0.543 \pm 0.000$ | 1.000 | 1.000 |
| | Drop core factors | $0.500 \pm 0.000$ | 1.000 | 0.750 |

Table 6: Candidate-order audit over 20 deterministic permutations. Exact match is mean $\pm$ standard deviation; agreement is with the stored-order prediction. A dash denotes a setting without candidate-energy ties to measure.

All predictions agree across the 20 permutations. The base prediction and canonical key resolve energy ties consistently. Removing grounding makes many candidates indistinguishable and increases tie rates, while the same tie rule preserves order invariance.

| | O/T | | Extended | |
|---|---|---|---|---|
| Subset | $n$ | Exact | $n$ | Exact |
| All examples | 384 | 0.857 | 256 | 0.941 |
| Unique minimum | 305 | 0.954 | 254 | 0.945 |
| Minimum-energy tie | 79 | 0.481 | 2 | 0.500 |
| Retain base prediction | 67 | 0.522 | 1 | 0.000 |
| Canonical tie-break | 12 | 0.250 | 1 | 1.000 |

Table 7: BBH decision decomposition from the archived Qwen2.5-14B candidate scores, reapplying the stated tie rule. Exact match is conditional on each subset; the last two rows partition the tied cases. Base retention occurs in $67/384 = 17.4\%$ of O/T examples and $1/256 = 0.4\%$ of Extended examples. All examples, including graph failures, remain in the denominator.

Unique minima account for 291 of the 329 correct O/T decisions and 240 of the 241 correct Extended decisions. This cache-only decomposition preserves candidate energies and reproduces the headline aggregate accuracies.

| Setting | Controlled | Logic grid | O/T | Extended |
|---|---|---|---|---|
| Uniform | 0.995 | 0.775 | 0.857 | 0.941 |
| Default | 0.995 | 0.775 | 0.857 | 0.941 |
| Each factor $\times 0.5$ | 0.995 | 0.775 | 0.857 | 0.941 |
| Each factor $\times 2$ | 0.995 | 0.775 | 0.857 | 0.941 |

Table 8: Fixed-candidate factor-weight sensitivity across all four suites. For perturbation rows, each factor family is changed separately while the candidate pool is held fixed, and the exact-match range is reported.

We perturb anchor, addition, and multiplication weights in the controlled study; person and pairwise factors in logic grids; ordering and tracking factors in O/T; and each task-consistency family in Extended. Every selected answer remains unchanged, showing that these rankings are stable over the tested weight range.

**Noisy extraction and broad checks.** The synthetic BBH stress test uses seeds 9101–9105, with missing, spurious, and mixed noise at rates 0.0, 0.1, and 0.2. The broader public-benchmark study uses a conservative adapter, which retains the base decision when reliable grounded factors are unavailable.

**Statistical reporting and hardware.** Bootstrap intervals use 2,000 row-level resamples. For paired binary exact-match outcomes, the reported two-sided exact McNemar test is the exact binomial test on discordant pairs (equivalently, the paired sign test used by the reporting scripts). Values written as mean ± standard deviation use the sample standard deviation over seeds. GPU generation and extraction run on one NVIDIA RTX PRO 6000 Blackwell Server Edition; deterministic projection, statistics, and table construction run on CPU.

# B Full Fixed-Pool Results

## B.1 Controlled-Chain Accuracy, Mechanisms, and Compute

| Variant | Direct | CoT | CoT-SC | Refl-SC | GLaCS@CoT-SC | GLaCS@Refl-SC |
|---|---|---|---|---|---|---|
| Numeric counterfactual | $0.236 \pm 0.015$ | $0.376 \pm 0.029$ | $0.854 \pm 0.021$ | $0.966 \pm 0.003$ | $0.968 \pm 0.004$ | $0.995 \pm 0.001$ |
| Order permuted | $0.225 \pm 0.013$ | $0.360 \pm 0.020$ | $0.875 \pm 0.012$ | $0.965 \pm 0.004$ | $0.974 \pm 0.003$ | $0.995 \pm 0.003$ |

Table 9: Controlled-chain exact match, reported as mean ± sample standard deviation over seeds 4121, 5121, and 6121 (512 examples per transformation and seed). GLaCS reranks the indicated fixed candidate pool.

| | Direct robustness | | | | CoT-SC reranking | | | Paired vs. CoT-SC | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Variant | Full | No grd. | No proj. | No factors | Full | No proj. | No factors | G-only | Base-only | $p$ |
| Numeric counterfactual | 0.997 | 0.000 | 0.000 | 0.000 | 0.968 | 0.680 | 0.719 | 176 | 0 | $< 10^{-12}$ |
| Order permuted | 0.995 | 0.001 | 0.000 | 0.000 | 0.974 | 0.681 | 0.747 | 153 | 1 | $< 10^{-12}$ |

Table 10: Controlled-chain mechanism evidence over the same three seeds. "Robust" evaluates transformed inputs directly; "Rerank" uses fixed CoT-SC candidate pools. Paired counts are GLaCS-only/CoT-SC-only, and $p$ is a two-sided exact McNemar test.

| Variant | Full | No proj. | No grd. | Drop core |
|---|---|---|---|---|
| Numeric cf. | $0.995 \pm 0.001$ | $0.802 \pm 0.028$ | $0.845 \pm 0.035$ | $0.858 \pm 0.010$ |
| Order perm. | $0.995 \pm 0.003$ | $0.813 \pm 0.026$ | $0.845 \pm 0.013$ | $0.870 \pm 0.014$ |

Table 11: Matched component ablations on the Controlled@Refl-SC headline candidate pools. Values are mean ± sample standard deviation over seeds 4121, 5121, and 6121.

Across all 3,072 paired examples, full GLaCS is correct while no-projection, no-grounding, and drop-core are wrong on 576, 461, and 402 examples, respectively; the reverse counts are zero and all exact McNemar $p$-values are below $10^{-120}$. Recomputed full predictions agree with the stored headline predictions on every example, and the $B = 0$ candidate ranking agrees exactly with the stored no-projection ranking.

| Regime | Full | No pres. | Identity | Collapse | Tie | Collision |
|---|---|---|---|---|---|---|
| Controlled | 0.995 | 0.992 | 0.305 | 0.366 | 0.366 | 0.366 |
| Logic grid | 0.775 | 1.000 | 0.370 | 0.299 | 0.299 | 0.299 |

Table 12: Removing candidate preservation on matched Refl-SC pools. Identity is the fraction of candidate instances whose answer changes. Collapse, tie, and collision are fractions of all examples; they coincide with the multi-candidate subset because every candidate then maps to one unconstrained state.

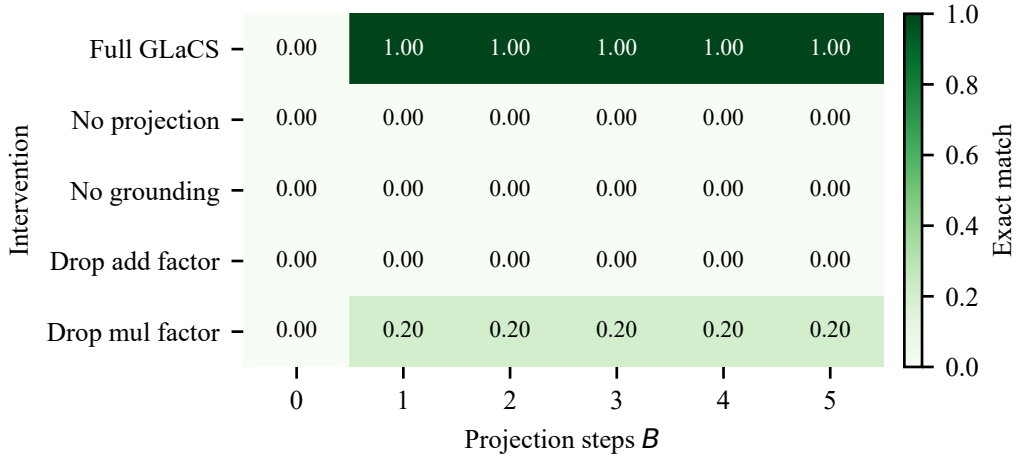


Figure 4: Controlled-chain accuracy under projection budgets and component interventions, including the no-projection endpoint $B = 0$.

| Method | Exact | Total tok./ex. | Gen. tok./ex. | Wall sec. | Rerank sec. |
|---|---|---|---|---|---|
| Direct | 0.218 | 194.1 | 7.1 | 43.4 | – |
| CoT | 0.354 | 228.0 | 32.1 | 250.9 | – |
| CoT-SC | 0.862 | 819.0 | 231.2 | 1159.8 | – |
| Reflection | 0.951 | 337.5 | 129.6 | 455.5 | – |
| Refl-SC | 0.962 | 1087.1 | 463.3 | 1777.3 | – |
| GLaCS@CoT-SC | 0.972 | 819.0 | 231.2 | 1160.2 | 0.327 |
| GLaCS@Refl-SC | 0.995 | 1087.1 | 463.3 | 1777.6 | 0.335 |

Table 13: Controlled-chain compute for seed 4121, averaged over numeric counterfactuals and order permutations. This timing run is distinct from the three-seed means in Table 10. Reranking reuses generated candidates; "Rerank sec." is total CPU time for 512 examples.

| Suite | Method | Calls/ex. | Prompt tok./ex. | Gen. tok./ex. | CPU proj. ms/ex. | Wall sec. | Peak MiB |
|---|---|---|---|---|---|---|---|
| Logic grid | Refl-SC | 3.00 | 602.2 | 1124.4 | – | 856.7 | 7417 |
| Logic grid | ToT-vote | 3.31 | 817.2 | 1127.9 | – | 883.8 | 7730 |
| Logic grid | Self-Refine | 4.00 | 1125.2 | 1195.5 | – | 952.7 | 7736 |
| Logic grid | Refl-SC+GLaCS | 3.00 | 602.2 | 1124.4 | 51.68 | 863.3 | 7417 |
| BBH O/T | CoT-SC | 4.00 | 885.9 | 1050.3 | – | 2318.8 | 7449 |
| BBH O/T | 14B graph-listwise | 6.00 | 1731.1 | 1194.7 | – | 3016.0 | 29129 |
| BBH O/T | graph+GLaCS | 5.00 | 1325.5 | 1183.1 | 5.11 | 2880.6 | 29086 |
| BBH Extended | CoT-SC | 4.00 | 547.0 | 702.1 | – | 1236.0 | 7416 |
| BBH Extended | 14B graph-listwise | 6.00 | 1044.3 | 819.3 | – | 1685.9 | 28936 |
| BBH Extended | graph+GLaCS | 5.00 | 766.8 | 812.4 | 0.07 | 1591.4 | 28759 |

Table 14: Full-pipeline compute measured serially on one `pro6000` server. Four-sample BBH generation feeds a 14B extraction call, followed by either a 14B graph-listwise call or CPU projection. Logic-grid ToT-vote skips selection when its pool is unanimous. Wall time sums the components, and peak CUDA allocation is their maximum. These batch-4 timing runs use separate accuracy measurements.

The timing table uses seed 4121; the mechanism table averages seeds 4121, 5121, and 6121. Each reranker reuses its generator's candidates and generation tokens.

Logic-grid assignment enumeration is the largest CPU component in Table 14. It reuses the generated pool. On the shared $K = 3$ timing pool, exact match is $0.656/0.664/0.391/0.781$ for Refl-SC, ToT-vote, Self-Refine, and Refl-SC+GLaCS. On the fresh BBH timing chain, CoT-SC, graph-listwise, and graph+GLaCS obtain $0.440/0.495/0.857$ on O/T and $0.449/0.750/0.934$ on Extended. These measurements account for the complete pipeline on the stated hardware and timing seeds.

### B.2 Logic-Grid Candidate Pools and Interventions

| Suite | Source | Base selector | Gold coverage | GLaCS | Conditional | Tokens/ex. |
|---|---|---|---|---|---|---|
| Controlled | CoT-SC | 1.000 | 1.000 | 0.992 | 0.992 | 2128.5 |
| Controlled | Direct-SC | 0.227 | 0.297 | 0.297 | 1.000 | 1522.6 |
| Controlled | Refl-SC | 1.000 | 1.000 | 1.000 | 1.000 | 2820.9 |
| Logic grid | CoT-SC | 0.836 | 0.930 | 0.930 | 1.000 | 4212.9 |
| Logic grid | Direct-SC | 0.297 | 0.719 | 0.719 | 1.000 | 1460.3 |
| Logic grid | Refl-SC | 0.852 | 0.945 | 0.945 | 1.000 | 4642.8 |

Table 15: Candidate-source audit at $K = 8$. Every selector reuses the frozen candidates generated by the indicated source. Conditional selection is exact match divided by gold coverage.

| Suite | $K$ | Gold coverage | Refl-SC | GLaCS | Conditional | Tokens/ex. | Projection ms/ex. |
|---|---|---|---|---|---|---|---|
| Controlled | 1 | 0.805 | 0.805 | 0.805 | 1.000 | 353.1 | 0.86 |
| Controlled | 3 | 1.000 | 0.969 | 1.000 | 1.000 | 1056.0 | 0.89 |
| Controlled | 5 | 1.000 | 0.992 | 1.000 | 1.000 | 1757.1 | 0.94 |
| Controlled | 8 | 1.000 | 1.000 | 1.000 | 1.000 | 2820.9 | 0.96 |
| Logic grid | 1 | 0.453 | 0.453 | 0.453 | 1.000 | 578.1 | 89.92 |
| Logic grid | 3 | 0.781 | 0.656 | 0.781 | 1.000 | 1751.3 | 123.63 |
| Logic grid | 5 | 0.898 | 0.758 | 0.898 | 1.000 | 2900.3 | 149.03 |
| Logic grid | 8 | 0.945 | 0.852 | 0.945 | 1.000 | 4642.8 | 158.14 |

Table 16: Refl-SC candidate-budget sensitivity. Candidate prefixes come from one frozen eight-sample generation run.

Figure 3 visualizes the accuracy and coverage trends.

| Candidate backbone | Refl-SC | Gold coverage | GLaCS | Gain |
|---|---|---|---|---|
| Qwen2.5-VL-3B | 0.656 | 0.781 | 0.781 | +0.125 |
| Phi-4-mini | 0.633 | 0.867 | 0.867 | +0.234 |

Table 17: Cross-backbone logic-grid audit at $n = 128$, seed 7103, and Refl-SC $K = 3$. Both rows use the same deterministic projection and differ only in the frozen candidate generator.

Projection reaches the coverage ceiling for both candidate backbones in this 128-example audit. The shared selector handles both generators' candidate formats.

| Variant | Direct | CoT | CoT-SC | Refl-SC | Refl-SC+GLaCS | Oracle prompt | Oracle GLaCS |
|---|---|---|---|---|---|---|---|
| Answer counterfactual | 0.320 | 0.367 | 0.695 | 0.672 | 0.805 | 0.695 | 0.805 |
| Base | 0.352 | 0.445 | 0.711 | 0.656 | 0.781 | 0.711 | 0.781 |
| Distractor augmented | 0.281 | 0.375 | 0.633 | 0.680 | 0.789 | 0.680 | 0.789 |
| Order permuted | 0.328 | 0.398 | 0.719 | 0.625 | 0.727 | 0.719 | 0.781 |
| All rows | 0.320 | 0.396 | 0.689 | 0.658 | 0.775 | 0.701 | 0.789 |

Table 18: Logic-grid exact match on 128 examples per transformation (seed 7103). The main comparison fixes the Refl-SC pool. Oracle columns give post-hoc upper bounds obtained by choosing the best tested prompt-family pool on the evaluation data.

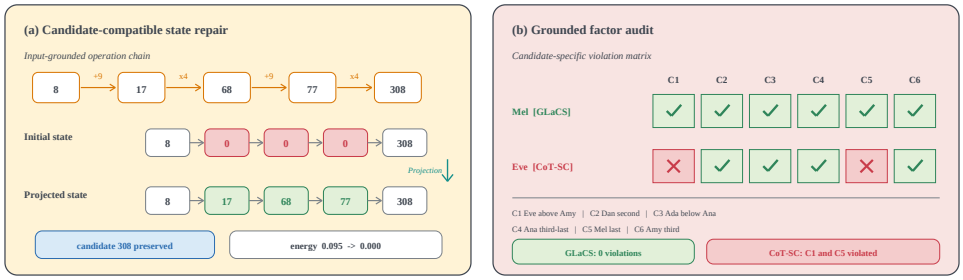


Figure 5: Qualitative analysis of candidate-preserving state projection. Left: the input operation chain makes explicit why projection changes $s_1, s_2, s_3$ while preserving the candidate answer 308. Right: short, input-grounded constraint labels expose why the GLaCS candidate has zero violations while the CoT-SC hard negative violates C1 and C5.

| Variant | Full GLaCS | No projection | No grounding / pairwise |
|---|---|---|---|
| Answer counterfactual | 0.805 | 0.641 | 0.648 |
| Base | 0.781 | 0.594 | 0.633 |
| Distractor augmented | 0.789 | 0.672 | 0.703 |
| Order permuted | 0.727 | 0.578 | 0.617 |
| All rows | 0.775 | 0.621 | 0.650 |

Table 19: Logic-grid ablations on the seed-7103 Refl-SC pools. No grounding and drop pairwise yield identical scores and predictions: every candidate can satisfy the remaining direct clues. The table groups these equivalent interventions in one column.

| Variant | Refl-SC | GLaCS | Covered | Covered selected | Margin | Pairwise changed | G-only | Refl-only |
|---|---|---|---|---|---|---|---|---|
| Answer counterfactual | 0.672 | 0.805 | 103/128 | 1.000 | 0.884 | 0.156 | 17 | 0 |
| Base | 0.656 | 0.781 | 100/128 | 1.000 | 0.950 | 0.148 | 16 | 0 |
| Distractor augmented | 0.680 | 0.789 | 101/128 | 1.000 | 0.943 | 0.086 | 14 | 0 |
| Order permuted | 0.625 | 0.727 | 93/128 | 1.000 | 0.857 | 0.109 | 13 | 0 |
| All rows | 0.658 | 0.775 | 397/512 | 1.000 | 0.909 | 0.125 | 60 | 0 |

Table 20: Logic-grid intervention evidence on fixed Refl-SC pools. “Covered” counts rows containing a correct generated candidate; “covered selected” is the fraction of those rows on which GLaCS selects it. The margin is the mean lowest-wrong minus correct projected energy in violation-count units. “Pairwise changed” is the fraction whose selected candidate changes after dropping pairwise factors. Paired counts are GLaCS-only/Refl-SC-only.

| Generator | $n$ | Refl-SC | Oracle | GLaCS | Gain | No projection | Drop pairwise |
|---|---|---|---|---|---|---|---|
| Qwen3.5-2B | 512 | 0.992 | 0.998 | 0.998 | +0.006 | 0.920 | 0.934 |
| Qwen3.5-4B | 512 | 1.000 | 1.000 | 1.000 | +0.000 | 1.000 | 1.000 |
| Qwen3.5-9B | 512 | 1.000 | 1.000 | 1.000 | +0.000 | 1.000 | 1.000 |

Table 21: Qwen3.5 logic-grid refresh on four seed-7103 transformations, with 128 examples each and three-sample Refl-SC pools. Native thinking is disabled. The 2B projector reaches candidate coverage; the 4B and 9B generators already attain perfect accuracy.

### B.3 Mechanism-Matched Proxy Audit

| Suite | $n$ | Refl-SC | ToT-vote | Self-Refine | GLaCS | GLaCS-only/ToT-only ($p$) |
|---|---|---|---|---|---|---|
| Controlled chain | 3072 | 0.965 | 0.866 | 0.771 | 0.995 | 397/0 ($< 10^{-12}$) |
| Logic grid | 512 | 0.658 | 0.691 | 0.377 | 0.775 | 47/4 ($< 10^{-3}$) |

Table 22: Strong Qwen baselines on the same Refl-SC candidate sets. ToT-vote uses one greedy listwise verifier call when candidates disagree; Self-Refine uses one greedy feedback–correction call. Paired counts and exact McNemar $p$-values compare GLaCS with ToT-vote.

| Suite | Method | $n$ | Exact (95% CI) | GLaCS − method | G-only/M-only | $p$ |
|---|---|---|---|---|---|---|
| Controlled chain | Refl-SC | 3072 | 0.965 [0.959, 0.972] | 0.030 | 91/0 | $< 10^{-12}$ |
| Controlled chain | Verifier-style proxy | 3072 | 0.864 [0.852, 0.876] | 0.131 | 402/0 | $< 10^{-12}$ |
| Controlled chain | Latent-search / EBM-style proxy | 3072 | 0.808 [0.793, 0.821] | 0.188 | 576/0 | $< 10^{-12}$ |
| Controlled chain | **GLaCS** | 3072 | 0.995 [0.992, 0.997] | – | – | – |
| Logic grid | Refl-SC | 512 | 0.658 [0.615, 0.697] | 0.117 | 60/0 | $< 10^{-12}$ |
| Logic grid | No-pairwise / ungrounded proxy | 512 | 0.650 [0.609, 0.693] | 0.125 | 64/0 | $< 10^{-12}$ |
| Logic grid | Latent-search / EBM-style proxy | 512 | 0.621 [0.578, 0.664] | 0.154 | 79/0 | $< 10^{-12}$ |
| Logic grid | **GLaCS** | 512 | 0.775 [0.740, 0.811] | – | – | – |

Table 23: Scoring proxies on fixed Refl-SC pools. Each proxy removes a component of the shared scoring procedure; the trained systems motivating their names are outside this comparison. Intervals are bootstrap 95% CIs. Paired counts are GLaCS-only/proxy-only, with two-sided exact McNemar $p$-values. Equivalent logic-grid switches share one row.

### B.4 Structured BBH Diagnostic Ceilings

| Task | Run | Prompted | Projection | Gain | Parser | No rel. | No fixed | No swap |
|---|---|---|---|---|---|---|---|---|
| BBH Logic-3 | CoT-SC | 0.750 | 1.000 | 0.250 | 1.000 | 0.484 | 0.781 | – |
| BBH Logic-5 | CoT-SC | 0.516 | 1.000 | 0.484 | 1.000 | 0.500 | 0.344 | – |
| BBH Logic-7 | CoT-SC | 0.359 | 1.000 | 0.641 | 1.000 | 0.562 | 0.281 | – |
| BBH Track-3 | CoT-SC | 0.422 | 1.000 | 0.578 | 1.000 | – | – | 0.281 |
| BBH Track-5 | CoT-SC | 0.297 | 1.000 | 0.703 | 1.000 | – | – | 0.109 |
| BBH Track-7 | CoT-SC | 0.234 | 1.000 | 0.766 | 1.000 | – | – | 0.109 |

Table 24: Deterministic-parser ceiling for BBH logical deduction and object tracking. The parser supplies grounded variables and constraints, and projection evaluates the explicit answer options. Ablations remove the named factor family.

| Task | $n$ | CoT-SC | Projection | Gain | Parser | No task factors |
|---|---|---|---|---|---|---|
| BBH Boolean | 64 | 0.812 | 1.000 | 0.188 | 1.000 | 0.641 |
| BBH Navigate | 64 | 0.406 | 1.000 | 0.594 | 1.000 | 0.438 |
| BBH Temporal | 64 | 0.297 | 1.000 | 0.703 | 1.000 | 0.328 |
| BBH Web Lies | 64 | 0.266 | 1.000 | 0.734 | 1.000 | 0.562 |
| All | 256 | 0.445 | 1.000 | 0.555 | 1.000 | 0.492 |

Table 25: Deterministic-parser ceiling on four extended BBH tasks. States encode time intervals, navigation displacement, truth links, or Boolean expressions. The final column removes the task's core factors.

Tables 24 and 25 evaluate deterministic, task-specific parsers. They establish an extraction ceiling for the induced constraint problems. Removing the relevant factor family lowers accuracy in each task.

Across the ten Qwen3.5-2B slices, deterministic parsing and option-level projection reach $1.000$, compared with $0.688$ for CoT-SC. The task-matched ablation reaches $0.508$. This study checks the projection interface and factor interventions with Qwen3.5 generations. Learned-extraction results appear in Section C.1.

| Task | $n$ | CoT-SC | Oracle | Projection | Gain | Parser | Key ablation |
|---|---|---|---|---|---|---|---|
| BBH Logic-3 | 64 | 0.938 | 1.000 | 1.000 | +0.063 | 1.000 | 0.938 |
| BBH Logic-5 | 64 | 0.719 | 0.859 | 1.000 | +0.281 | 1.000 | 0.750 |
| BBH Logic-7 | 64 | 0.594 | 0.766 | 1.000 | +0.406 | 1.000 | 0.625 |
| BBH Track-3 | 64 | 0.688 | 0.922 | 1.000 | +0.313 | 1.000 | 0.281 |
| BBH Track-5 | 64 | 0.656 | 0.891 | 1.000 | +0.344 | 1.000 | 0.109 |
| BBH Track-7 | 64 | 0.609 | 0.750 | 1.000 | +0.391 | 1.000 | 0.109 |
| BBH Boolean | 64 | 0.938 | 1.000 | 1.000 | +0.063 | 1.000 | 0.609 |
| BBH Navigate | 64 | 0.812 | 0.922 | 1.000 | +0.188 | 1.000 | 0.359 |
| BBH Temporal | 64 | 0.672 | 0.922 | 1.000 | +0.328 | 1.000 | 0.672 |
| BBH Web Lies | 64 | 0.250 | 0.703 | 1.000 | +0.750 | 1.000 | 0.625 |
| All | 640 | 0.688 | 0.873 | 1.000 | +0.313 | 1.000 | 0.508 |

Table 26: Qwen3.5-2B generation with deterministic BBH projectors. CoT-SC uses four samples; Oracle is any-sample accuracy. Projection evaluates the full explicit option set with a task-specific parser. Key ablations remove relative ordering, swaps, or the extended task's core factors.

### B.5 Qwen3.5 Learned Extraction and Strong Verifiers

All rows below use the pro6000 refresh with native Qwen3.5 thinking disabled. Candidate generators use four CoT-SC samples and a 1536-token ceiling. Learned extractors decode greedily with a 512-token ceiling. Extractor batch sizes $16/8/4$ for 2B/4B/9B change throughput only; prompts, seeds, candidate pools, and scoring remain fixed.

| Generator | $n$ | CoT-SC | Oracle | Projection | Gain | Key ablation |
|---|---|---|---|---|---|---|
| Qwen3.5-2B | 640 | 0.688 | 0.873 | 1.000 | +0.313 | 0.508 |
| Qwen3.5-4B | 640 | 0.927 | 0.986 | 1.000 | +0.073 | 0.630 |
| Qwen3.5-9B | 640 | 0.992 | 0.997 | 1.000 | +0.008 | 0.641 |

Table 27: Candidate-generation refresh across ten BBH variants. Projection uses the full option set and deterministic parsing, giving a diagnostic ceiling of 1.000. Higher generator accuracy leaves progressively less room for improvement.

| Method | $n$ | Exact | Gain over CoT-SC | Gap to GLaCS |
|---|---|---|---|---|
| CoT-SC | 256 | 0.668 | – | 0.289 |
| Qwen3.5-9B listwise | 256 | 0.742 | +0.074 | 0.215 |
| Qwen3.5-9B graph-listwise | 256 | 0.742 | +0.074 | 0.215 |
| **GLaCS learned projection** | 256 | **0.957** | **+0.289** | – |

Table 28: Qwen3.5-9B selector comparison on the four extended BBH tasks, with Qwen3.5-2B candidates fixed. Graph-listwise selection and GLaCS receive the same extracted graph and use generative selection and factorized projection, respectively.

| Audit | Scope | $n$ | Learned projection | Reference / ceiling | JSON valid | Graph valid |
|---|---|---|---|---|---|---|
| Visible options | Core six | 384 | 0.880 | – | 1.000 | 0.997 |
| Stem-only graph-induced | Core six | 384 | 0.875 | 0.880 ($-0.005$) | 1.000 | 0.997 |
| Hidden-option stability | Extended four | $3\times256$ | $0.936\pm0.018$ | $1.000\pm0.000$ | $1.000\pm0.000$ | $0.992\pm0.004$ |

Table 29: Option-hidden and multi-seed evaluations of Qwen3.5-9B extraction. The core study induces candidates from a stem-only graph. The extended study uses seeds 9301, 9302, and 9303, with 64 examples per task and seed; values are mean $\pm$ standard deviation. The parser reference is a deterministic extraction ceiling.

With Qwen3.5-2B candidates fixed, learned projection improves the core aggregate by $0.169$–$0.245$ as extractor capacity grows and the extended aggregate by $0.289$. It exceeds both 9B listwise variants by $0.215$ (Table 28). Table 29 measures the effects of hiding options and changing extraction seeds.

| Adapter | Records | State | Core factors | Projection primitive |
|---|---|---|---|---|
| Chain | 3 | numeric sequence | anchor/add/multiply | numeric repair |
| Logic grid | 3 | bijective assignment | direct/pairwise/bijection | assignment search |
| Logical order | 2 | permutation | fixed/relative rank | permutation search |
| Object tracking | 3 | ownership map | initial/swap/query | ordered simulation |
| Temporal | 3 | time intervals | wake/close/busy | interval arithmetic |
| Navigation | 3 | pose/displacement | facing/move/query | ordered simulation |
| Web of Lies | 3 | truth vector | base/claim/query | truth propagation |
| Boolean | 1 | Boolean value | expression | expression evaluation |

Table 30: Task-specific components of the common interface. Records counts normalized families of task-core records.

### B.6 Generic Constraint DSL Audit

The shared implementation handles candidate enumeration, compatibility checks, factor storage, energy aggregation, projection calls, tie breaking, and final selection. Table 30 lists the task-specific translators and projection primitives required by this interface.

| Family | $n$ | Specialized | Generic DSL | Gap | Agreement |
|---|---|---|---|---|---|
| Boolean | 64 | 1.000 | 1.000 | 0.000 | 1.000 |
| Logic Grid | 128 | 1.000 | 1.000 | 0.000 | 1.000 |
| Temporal | 64 | 1.000 | 1.000 | 0.000 | 1.000 |
| Tracking-five | 64 | 1.000 | 1.000 | 0.000 | 1.000 |
| Tracking-seven | 64 | 1.000 | 1.000 | 0.000 | 1.000 |
| Tracking-three | 64 | 1.000 | 1.000 | 0.000 | 1.000 |
| All | 448 | 1.000 | 1.000 | 0.000 | 1.000 |

Table 31: Generic constraint DSL audit. Task-specific translators emit the same constraint-program interface, and one interpreter authorizes candidates by lowest projected energy.

For Table 31, task-specific translators convert Temporal, Boolean, Tracking, and logic-grid inputs into a common constraint program. One interpreter then evaluates all candidates with the same projected-energy rule. It matches the specialized solvers' accuracy and decisions on all $448$ examples.

### B.7 Logic-Grid n512 Multi-Seed Stability

| Transformation | Refl-SC | GLaCS | No projection | Drop pairwise |
|---|---|---|---|---|
| Base | $0.66\pm0.03$ | $\mathbf{0.79\pm0.01}$ | $0.66\pm0.02$ | $0.66\pm0.02$ |
| Order perm. | $0.66\pm0.02$ | $\mathbf{0.78\pm0.01}$ | $0.63\pm0.02$ | $0.65\pm0.01$ |
| Answer CF | $0.64\pm0.03$ | $\mathbf{0.76\pm0.04}$ | $0.64\pm0.03$ | $0.64\pm0.04$ |
| Distractor aug. | $0.61\pm0.01$ | $\mathbf{0.73\pm0.02}$ | $0.60\pm0.01$ | $0.61\pm0.01$ |

Table 32: Logic-grid stability on fixed Refl-SC candidate pools. Results are mean $\pm$ standard deviation over three seeds, with $512$ examples per transformation. CF denotes answer counterfactual.

The larger logic-grid audit expands each transformation from $n = 128$ to $512$ examples for each of three seeds, using Refl-SC candidates throughout. Across the twelve seed–transformation combinations, Refl-SC averages $0.643$ and GLaCS averages $0.764$, a gain of $0.121$. Per-transformation gains range from $0.115$ to $0.132$. Removing projection or pairwise factors returns performance close to the base selector, consistent with the smaller study.

| Slice | $n$ | CoT-SC | 3B projection | 7B projection | 14B projection | 14B graph ok |
|---|---|---|---|---|---|---|
| Logical | 192 | 0.542 | 0.427 | 0.755 | 0.776 | 0.995 |
| Tracking | 192 | 0.318 | 0.422 | 0.839 | 0.938 | 1.000 |
| All | 384 | 0.430 | 0.424 | 0.797 | 0.857 | 0.997 |
| Logic-3 | 64 | 0.750 | 0.531 | 0.891 | 0.875 | 1.000 |
| Logic-5 | 64 | 0.516 | 0.359 | 0.719 | 0.719 | 1.000 |
| Logic-7 | 64 | 0.359 | 0.391 | 0.656 | 0.734 | 0.984 |
| Track-3 | 64 | 0.422 | 0.500 | 0.844 | 1.000 | 1.000 |
| Track-5 | 64 | 0.297 | 0.375 | 0.922 | 0.922 | 1.000 |
| Track-7 | 64 | 0.234 | 0.391 | 0.750 | 0.891 | 1.000 |

Table 33: Text-only extraction on 384 BBH examples. Each model emits a JSON graph, and a shared projector evaluates fixed options. Graph validity measures required task fields. JSON validity is 1.000 at every scale; aggregate graph validity is 0.633/0.992/0.997 for 3B/7B/14B.

# C Learned Extraction and Leakage Audits

## C.1 Detailed Learned-Extraction Results

The 3B extractor performs slightly below CoT-SC, while the 7B and 14B extractors improve substantially. The paired 7B-versus-3B counts are 158/15 ($p = 2.8 \times 10^{-31}$, exact McNemar test). The 14B-versus-7B counts are 36/26 ($p = 0.253$). Thus the clearest scaling gain occurs between 3B and 7B; the next increment has greater uncertainty.

*(a) Graph-semantic fidelity*

| Extractor | $n$ | JSON | Core fields | Entity F1 | Factor F1 | Exact graph | End-to-end |
|---|---|---|---|---|---|---|---|
| Qwen2.5-3B | 640 | 1.000 | 0.553 | 0.876 | 0.699 | 0.163 | 0.594 |
| Qwen2.5-7B | 640 | 1.000 | 0.648 | 0.921 | 0.850 | 0.438 | 0.847 |
| Qwen2.5-14B | 640 | 1.000 | 0.998 | 0.945 | 0.879 | 0.675 | 0.891 |

*(b) Exclusive error decomposition*

| Extractor | Correct | Coverage | Invalid | Entity/query | Missing | Spurious | Projection | Other |
|---|---|---|---|---|---|---|---|---|
| Qwen2.5-3B | 380 | 0 | 0 | 156 | 104 | 0 | 0 | 0 |
| Qwen2.5-7B | 542 | 0 | 0 | 47 | 48 | 0 | 3 | 0 |
| Qwen2.5-14B | 570 | 0 | 0 | 20 | 50 | 0 | 0 | 0 |

Table 34: Semantic fidelity of answer-free BBH constraint extraction against the deterministic structured graph. JSON/core-field success is separated from semantic entity and factor agreement. Entity and factor scores are micro F1. Panel (b) assigns each end-to-end error to one exclusive first-applicable category.

All outputs parse at every model size, while semantic fidelity varies substantially. Exact graph match rises from 0.163 to 0.438 and 0.675, and factor F1 from 0.699 to 0.850 and 0.879. Missing factors and entity/query mismatches dominate the remaining errors; the 7B run also has three projection errors. A deterministic review covers 160 sorted cases with a graph mismatch or answer error and retains their normalized graph comparisons.

| Group | CoT-SC | Parser projection | LLM projection | Gain | JSON | Graph |
|---|---|---|---|---|---|---|
| Temporal | 0.297 | 1.000 | 1.000 | 0.703 | 1.000 | 1.000 |
| Navigate | 0.406 | 1.000 | 0.875 | 0.469 | 1.000 | 1.000 |
| Web Lies | 0.266 | 1.000 | 0.891 | 0.625 | 1.000 | 0.984 |
| Boolean | 0.812 | 1.000 | 1.000 | 0.188 | 1.000 | 1.000 |
| All | 0.445 | 1.000 | 0.941 | 0.496 | 1.000 | 0.996 |

Table 35: Extended text-only Qwen2.5-14B extraction. Parser projection is a diagnostic ceiling. "JSON" is syntactic validity; "Graph" is task-core graph validity; gain is LLM projection minus CoT-SC. Each family has 64 examples.

| Group | Cand. oracle | Graph-induced | Fixed options | Difference | JSON | Graph |
|---|---|---|---|---|---|---|
| Logical | 1.000 | 0.776 | 0.771 | 0.005 | 1.000 | 1.000 |
| Tracking | 0.995 | 0.698 | 0.745 | -0.047 | 1.000 | 0.990 |
| All | 0.997 | 0.737 | 0.758 | -0.021 | 1.000 | 0.995 |

Table 36: Stem-only candidate audit with Qwen2.5-14B. The extractor and candidate constructor see no options, labels, or candidate-name list. "Cand. oracle" is gold-answer coverage. "Graph-induced" uses candidates derived from the extracted graph; "Fixed options" uses the same stem-only graph but the original option set. JSON and Graph are validity rates.

Tracking candidates retain 0.995 gold coverage. The 0.047 gap between induced and fixed candidates therefore points to answer canonicalization and selection under the extracted graph.

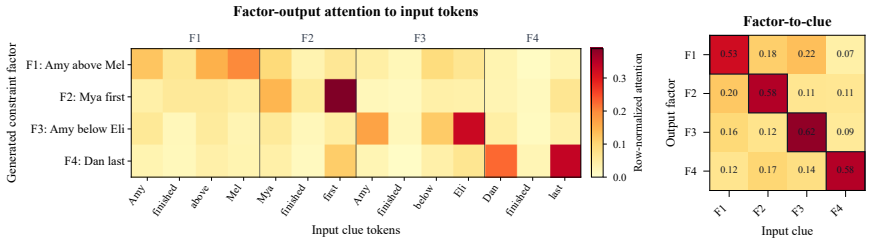


Figure 6: Attention probe for the text-only 7B extractor. We average decoder self-attention over the last four layers while teacher-forcing the generated JSON. Across 64 logical-deduction examples, factor-to-clue top-1 agreement is 0.863 and mean diagonal attention share is 0.600. This is an extractor diagnostic; projection uses the emitted factors.

The attention probe describes where the extractor attends. The factor-deletion experiment below measures how individual grounded constraints affect candidate energy.

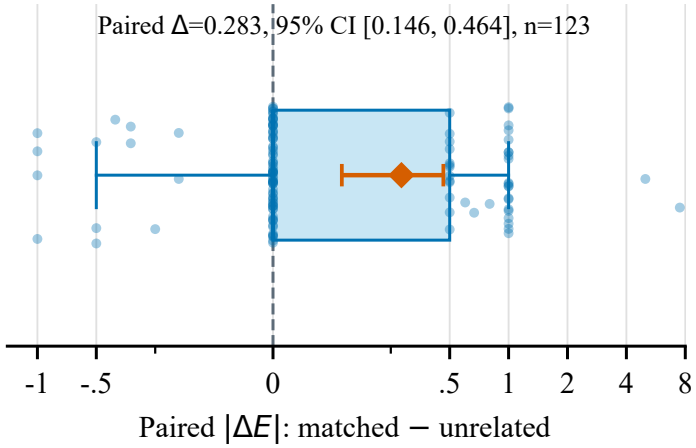


Figure 7: Grounded-span intervention on 123 learned logical graphs. One-factor deletion produces a larger hard-negative energy response for candidate-matched factors than for unrelated factors in the same graph. Points are examples, the box shows the interquartile range, and the orange diamond/error bar is the mean with its 95% bootstrap confidence interval.

**Causal factor-deletion protocol.** We intervene on the extracted graph and keep the extractor output fixed. For each logical example, we select the lowest-energy incorrect candidate and delete one factor at a time. The matched response averages absolute energy changes over factors whose grounded entities occur in that candidate. The unrelated response averages the remaining factors in the same graph. Their paired difference measures candidate-specific sensitivity to a single-factor deletion.

### C.2 VL Weak Extractor

| Group | CoT-SC | Parser proj. | LLM proj. | Gain | JSON | Graph |
|---|---|---|---|---|---|---|
| Logical | 0.542 | 1.000 | 0.734 | 0.193 | 1.000 | 0.979 |
| Tracking | 0.318 | 1.000 | 0.646 | 0.328 | 1.000 | 0.833 |
| All | 0.430 | 1.000 | 0.690 | 0.260 | 1.000 | 0.906 |

Table 37: Qwen2.5-VL-7B used as a text-conditioned JSON extractor on BBH. The shared projector selects among the answer options. Table 33 gives the text-only extractor comparison.

Table 37 evaluates Qwen2.5-VL-7B as a text-conditioned graph extractor. It reaches 0.690 across O/T and provides an additional comparison with the text-only Qwen2.5-Instruct models.

### C.3 Strong BBH Listwise Verifier

| Group | Method | $n$ | CoT-SC | Verifier | GLaCS | GLaCS-verifier |
|---|---|---|---|---|---|---|
| Logical | Qwen14B listwise | 192 | 0.542 | 0.740 | 0.776 | 0.036 |
| Logical | Qwen14B graph-listwise | 192 | 0.542 | 0.729 | 0.776 | 0.047 |
| Tracking | Qwen14B listwise | 192 | 0.318 | 0.208 | 0.938 | 0.729 |
| Tracking | Qwen14B graph-listwise | 192 | 0.318 | 0.260 | 0.938 | 0.677 |
| O/T All | Qwen14B listwise | 384 | 0.430 | 0.474 | 0.857 | 0.383 |
| O/T All | Qwen14B graph-listwise | 384 | 0.430 | 0.495 | 0.857 | 0.362 |
| Temporal | Qwen14B listwise | 64 | 0.297 | 0.859 | 1.000 | 0.141 |
| Temporal | Qwen14B graph-listwise | 64 | 0.297 | 0.922 | 1.000 | 0.078 |
| Navigate | Qwen14B listwise | 64 | 0.406 | 0.609 | 0.875 | 0.266 |
| Navigate | Qwen14B graph-listwise | 64 | 0.406 | 0.609 | 0.875 | 0.266 |
| Web Lies | Qwen14B listwise | 64 | 0.266 | 0.500 | 0.891 | 0.391 |
| Web Lies | Qwen14B graph-listwise | 64 | 0.266 | 0.531 | 0.891 | 0.359 |
| Boolean | Qwen14B listwise | 64 | 0.812 | 0.781 | 1.000 | 0.219 |
| Boolean | Qwen14B graph-listwise | 64 | 0.812 | 0.781 | 1.000 | 0.219 |
| All | Qwen14B listwise | 256 | 0.445 | 0.688 | 0.941 | 0.254 |
| All | Qwen14B graph-listwise | 256 | 0.445 | 0.711 | 0.941 | 0.230 |

Table 38: Qwen2.5-14B listwise selection on shared BBH options. Graph-listwise selection also receives the extracted JSON graph. Its generated choice is compared with the projected-energy decision.

Table 38 compares the large generic verifier with factorized projection. The graph-listwise selector receives the same extracted JSON graph as GLaCS and generates its selected answer. It reaches 0.495 on O/T and 0.711 on Extended, compared with 0.857 and 0.941 for projection. The Extended gap is 23.0 points. These comparisons measure the contribution of explicit state evaluation under shared extracted constraints.

### C.4 BBH Extractor Confidence and Leakage Audits

| Group | $n$ | CoT-SC | Projection | Gain | Win/loss |
|---|---|---|---|---|---|
| Logical | 192 | 0.542 [0.469, 0.615] | 0.776 [0.714, 0.833] | 0.234 [0.161, 0.307] | 53/8 |
| Tracking | 192 | 0.318 [0.255, 0.385] | 0.938 [0.901, 0.969] | 0.620 [0.552, 0.688] | 119/0 |
| All | 384 | 0.430 [0.380, 0.479] | 0.857 [0.820, 0.891] | 0.427 [0.372, 0.479] | 172/8 |

Table 39: Row-level confidence audit for the text-only Qwen2.5-14B extractor on BBH logical-deduction and tracking tasks. Intervals are bootstrap 95% CIs over examples; win/loss counts compare projection against CoT-SC.

These audits report bootstrap confidence intervals and paired win/loss counts for the text-only Qwen2.5-14B extractor. On the original order/tracking slice, the aggregate projection gain over CoT-SC is $0.427$ with a bootstrap $95\%$ CI of $[0.372, 0.479]$. On the extended BBH slice, the aggregate gain

| Group | $n$ | CoT-SC | Projection | Gain | Win/loss |
|---|---|---|---|---|---|
| Temporal | 64 | 0.297 [0.188, 0.406] | 1.000 [1.000, 1.000] | 0.703 [0.594, 0.812] | 45/0 |
| Navigate | 64 | 0.406 [0.281, 0.531] | 0.875 [0.797, 0.953] | 0.469 [0.344, 0.594] | 31/1 |
| Web Lies | 64 | 0.266 [0.156, 0.375] | 0.891 [0.812, 0.953] | 0.625 [0.500, 0.750] | 41/1 |
| Boolean | 64 | 0.812 [0.719, 0.906] | 1.000 [1.000, 1.000] | 0.188 [0.094, 0.281] | 12/0 |
| All | 256 | 0.445 [0.387, 0.508] | 0.941 [0.910, 0.969] | 0.496 [0.434, 0.559] | 129/2 |

Table 40: Row-level confidence audit for the text-only Qwen2.5-14B extractor on extended BBH global-consistency tasks. Intervals are bootstrap 95% CIs over examples; win/loss counts compare projection against CoT-SC.

| Group | $n$ | CoT-SC | Fixed graph | Stem-only graph | Drop | Stem graph |
|---|---|---|---|---|---|---|
| Temporal | 64 | 0.297 | 1.000 | 1.000 | 0.000 | 1.000 |
| Navigate | 64 | 0.406 | 0.875 | 0.875 | 0.000 | 1.000 |
| Web Lies | 64 | 0.266 | 0.891 | 0.891 | 0.000 | 0.984 |
| Boolean | 64 | 0.812 | 1.000 | 1.000 | 0.000 | 1.000 |
| All | 256 | 0.445 | 0.941 | 0.941 | 0.000 | 0.996 |

Table 41: Option-hidden extraction audit for extended BBH tasks. The stem-only extractor sees no answer options or option labels; projection is still evaluated over the original fixed options to isolate extraction leakage.

| Group | Seeds | $n$/seed | Parser | LLM projection | Graph |
|---|---|---|---|---|---|
| Temporal | 3 | 64 | 1.000±0.000 | 0.990±0.009 | 1.000±0.000 |
| Navigate | 3 | 64 | 1.000±0.000 | 0.844±0.031 | 1.000±0.000 |
| Web Lies | 3 | 64 | 1.000±0.000 | 0.885±0.024 | 0.979±0.009 |
| Boolean | 3 | 64 | 1.000±0.000 | 1.000±0.000 | 1.000±0.000 |
| All | 3 | 256 | 1.000±0.000 | 0.930±0.010 | 0.995±0.002 |

Table 42: Extended BBH extractor stability across sampled seeds. Each seed samples 64 examples per family directly from the public BBH task files. The extractor sees no answer options or option labels; projection is evaluated over the original fixed options.

is $0.496$ with CI $[0.434, 0.559]$. The option-hidden extended audit removes answer options and option labels from the extractor prompt while still projecting over the original fixed options; the aggregate result remains $0.941$, supporting extraction with the option labels hidden. The multi-seed stability audit resamples each extended family with three seeds. Projection remains high at $0.930 \pm 0.010$ aggregate accuracy, with JSON validity $1.000$ and graph success $0.995 \pm 0.002$, showing stable results across the sampled 64-example slices.

# D Diagnostic and Boundary Analyses

## D.1 Broad Public Benchmark Reranking

Table 43 evaluates a conservative adapter on broader reasoning benchmarks with fixed Qwen-generated candidates. Gains are small on several arithmetic word-problem rows, and most rows are unchanged. This behavior reflects the limited availability of reliable grounded factors outside the structured tasks.

| Task | Run | Prompted | GLaCS | Gain | Parse | Oracle | No projection |
|---|---|---|---|---|---|---|---|
| ASDiv-A | CoT-SC | 1.000 | 1.000 | 0.000 | 1.000 | 1.000 | 0.984 |
| ASDiv-A | Direct-SC | 0.922 | 0.922 | 0.000 | 1.000 | 1.000 | 0.922 |
| BBH Logic-3 | CoT-SC | 0.750 | 0.750 | 0.000 | 1.000 | 0.969 | 0.703 |
| BBH Logic-3 | Direct-SC | 0.578 | 0.578 | 0.000 | 1.000 | 0.969 | 0.578 |
| BBH Track-3 | CoT-SC | 0.422 | 0.422 | 0.000 | 1.000 | 0.828 | 0.500 |
| BBH Track-3 | Direct-SC | 0.297 | 0.297 | 0.000 | 1.000 | 0.859 | 0.297 |
| DROP | CoT-SC | 0.609 | 0.609 | 0.000 | 0.984 | 0.750 | 0.672 |
| DROP | Direct-SC | 0.562 | 0.562 | 0.000 | 1.000 | 0.703 | 0.531 |
| GSM8K | CoT-SC | 0.797 | 0.828 | 0.031 | 1.000 | 0.891 | 0.797 |
| GSM8K | Direct-SC | 0.156 | 0.156 | 0.000 | 1.000 | 0.266 | 0.125 |
| MATH-500 | CoT-SC | 0.469 | 0.469 | 0.000 | 0.750 | 0.531 | 0.438 |
| MATH-500 | Direct-SC | 0.188 | 0.188 | 0.000 | 1.000 | 0.453 | 0.188 |
| MultiArith | CoT-SC | 1.000 | 1.000 | 0.000 | 1.000 | 1.000 | 1.000 |
| MultiArith | Direct-SC | 0.281 | 0.297 | 0.016 | 1.000 | 0.547 | 0.297 |
| ProofWriter | CoT-SC | 0.641 | 0.641 | 0.000 | 0.938 | 0.828 | 0.641 |
| ProofWriter | Direct-SC | 0.656 | 0.656 | 0.000 | 1.000 | 0.984 | 0.672 |
| SVAMP | CoT-SC | 0.953 | 0.984 | 0.031 | 1.000 | 0.984 | 0.969 |
| SVAMP | Direct-SC | 0.516 | 0.531 | 0.016 | 1.000 | 0.656 | 0.531 |

Table 43: Public reasoning benchmark reranking with Qwen-generated candidates. GLaCS uses the same generated candidates and reranks them with factorized energy.

### D.2 Imperfect Constraint Extraction

| Setting | Logical | Tracking | All | Gain vs. CoT-SC |
|---|---|---|---|---|
| CoT-SC | 0.542 | 0.318 | 0.430 | – |
| Projection, clean | 1.000 | 1.000 | 1.000 | 0.570 |
| Missing factors, $q = 0.1$ | 0.830 | 0.808 | 0.819 | 0.390 |
| Missing factors, $q = 0.2$ | 0.699 | 0.648 | 0.673 | 0.244 |
| Spurious factors, $q = 0.1$ | 0.872 | 0.801 | 0.836 | 0.407 |
| Spurious factors, $q = 0.2$ | 0.778 | 0.653 | 0.716 | 0.286 |
| Mixed noise, $q = 0.1$ | 0.734 | 0.658 | 0.696 | 0.267 |
| Mixed noise, $q = 0.2$ | 0.570 | 0.464 | 0.517 | 0.087 |

Table 44: Robustness to imperfect constraint extraction on public BBH global-consistency tasks. Missing noise drops true factors; spurious noise injects corrupted factors; mixed noise does both with the same rate $q$. Results average over five seeds.

| Setting | O/T exact | O/T valid | Extended exact | Extended valid |
|---|---|---|---|---|
| Clean | $0.857 \pm 0.000$ | $0.997 \pm 0.000$ | $0.941 \pm 0.000$ | $0.996 \pm 0.000$ |
| Missing 10% | $0.760 \pm 0.005$ | $0.995 \pm 0.000$ | $0.823 \pm 0.020$ | $0.878 \pm 0.023$ |
| Missing 20% | $0.654 \pm 0.008$ | $0.990 \pm 0.005$ | $0.708 \pm 0.002$ | $0.771 \pm 0.016$ |
| Spurious 10% | $0.766 \pm 0.008$ | $0.997 \pm 0.000$ | $0.829 \pm 0.008$ | $0.957 \pm 0.007$ |
| Spurious 20% | $0.663 \pm 0.019$ | $0.998 \pm 0.002$ | $0.743 \pm 0.026$ | $0.914 \pm 0.021$ |
| Mixed 10% | $0.685 \pm 0.009$ | $0.995 \pm 0.000$ | $0.762 \pm 0.018$ | $0.858 \pm 0.012$ |
| Mixed 20% | $0.546 \pm 0.008$ | $0.990 \pm 0.006$ | $0.659 \pm 0.018$ | $0.745 \pm 0.015$ |

Table 45: Noise stress test on actual Qwen2.5-14B answer-free extracted graphs. Exact match and graph validity are mean $\pm$ sample standard deviation over three deterministic factor-perturbation seeds; clean rows are deterministic.

Table 44 perturbs deterministic BBH graphs. Dropping $10\%$ of true factors or injecting corrupted factors at a $10\%$ rate leaves aggregate accuracy above $0.81$, compared with $0.430$ for CoT-SC. At $20\%$, either noise type retains a gain of 24–29 percentage points. Applying both types at $20\%$ reduces the gain to $8.7$ points.

Table 45 applies the same perturbations to Qwen2.5-14B graphs. Accuracy decreases with noise in every family. Extended graph validity is particularly sensitive: removing a required interval, move, truth link, or Boolean operator can make execution fail. These rows remain in the evaluation and use

| Graph condition | O/T exact | Extended exact | Graph valid |
|---|---|---|---|
| Matched graph | $0.857 \pm 0.000$ | $0.941 \pm 0.000$ | 0.997/0.996 |
| Within-task shuffled | $0.408 \pm 0.012$ | $0.452 \pm 0.034$ | 0.997/0.996 |
| Empty graph | $0.432 \pm 0.000$ | $0.543 \pm 0.000$ | 0.000/0.000 |

Table 46: Graph-semantic negative controls over 20 within-task derangements. Shuffling preserves JSON shape and task-core validity but assigns each example another example's executable graph.

the canonical tie rule. In Table 46, within-task shuffling preserves graph validity but lowers accuracy to 0.408/0.452, near the empty-graph controls. The useful signal therefore comes from constraints matched to each input.

### D.3 Generated-Candidate Audit Details

| Group | $n$ | Candidate oracle | Direct graph-induced | Fixed options |
|---|---|---|---|---|
| Logical | 192 | 1.000 | 0.755 | 0.776 |
| Tracking | 192 | 1.000 | 0.885 | 0.938 |
| All | 384 | 1.000 | 0.820 | 0.857 |

Table 47: Direct execution on O/T with Qwen2.5-14B graphs and graph-induced answer candidates. Candidate oracle measures gold coverage of the induced set. Fixed-option projection provides the comparison.

Direct graph-induced execution reaches 0.820 on O/T with the original option labels hidden, compared with 0.857 for candidate-preserving fixed-option projection. This audit evaluates the extracted constraints using a graph-derived answer space.

| Family | $n$ | Exact | In-candidate | Invalid/ambig. |
|---|---|---|---|---|
| Temporal | 64 | 0.000 | 0.000 | 1.000 |
| Navigate | 64 | 0.875 | 1.000 | 0.000 |
| Web of Lies | 64 | 0.891 | 0.984 | 0.016 |
| Boolean | 64 | 1.000 | 1.000 | 0.000 |
| All | 256 | 0.691 | 0.746 | 0.254 |

Table 48: Option-free direct execution on Extended BBH. Closed-domain graphs emit answers without consulting the original options. Temporal schedule graphs define a set of feasible free intervals but no unique answer-denoting interval, so the direct executor abstains. Abstentions count as errors in "Exact."

Direct execution produces a single answer for the closed-domain tasks. Temporal scheduling instead yields a set of feasible intervals. Candidate-conditioned evaluation resolves the queried choice across all four families and reaches 0.941 overall.

The stem-only audit induces a median of five candidates, with gold coverage 0.997 and zero empty or duplicate sets after canonicalization. Relative to fixed-option projection with the same stem-only graph, induced candidates produce 1 win, 9 losses, and 374 ties across 384 examples. Both extraction and candidate construction use the problem stem with the original options hidden.

| Group | $n$ | Avg. cand. | Median | Range |
|---|---|---|---|---|
| Logical | 192 | 5.000 | 5 | 3–7 |
| Tracking | 192 | 4.990 | 5 | 3–7 |
| All | 384 | 4.995 | 5 | 3–7 |

| Group | Gold cov. | Empty | Duplicate | W/L/T |
|---|---|---|---|---|
| Logical | 1.000 | 0.000 | 0.000 | 1/0/191 |
| Tracking | 0.995 | 0.000 | 0.000 | 0/9/183 |
| All | 0.997 | 0.000 | 0.000 | 1/9/374 |

Table 49: Candidate-set diagnostics for the stem-only BBH generated-candidate audit. Both the extractor and candidate-construction prompts hide the original answer options. Candidates are induced from the Qwen2.5-14B extracted graph and canonicalized before projection. Empty and duplicate rates are measured after canonicalization. W/L/T compares graph-induced LLM projection with fixed-option LLM projection using the same stem-only extracted graph.