Do LLMs Respect Concept Boundaries? A Bilingual Diagnostic Benchmark

Renjun Lai[1] **Yixuan Si**[2]
[1]Hangzhou Dianzi University, Hangzhou, China
[2]Westlake University, Hangzhou, China
hdulrj@hdu.edu.cn siyixuan@westlake.edu.cn

Abstract

Large language models are increasingly used as natural-language interfaces for structured information processing and agentic workflows, where user-specified constraints can determine downstream extraction, selection, tool use, and action planning. This paper studies whether models can execute natural-language concept-set constraints over candidate objects: given inclusion and exclusion conditions, the model must decide which candidates fall inside the intended concept boundary. We introduce a dual-resource diagnostic benchmark built from WordNet and BabelNet. The benchmark fixes backend constraints, candidate realizations, and gold answers before prompt rendering, and evaluates three task views: single-object judgment, exact candidate-set construction, and satisfiability judgment. The main evaluation set contains 202,500 rendered requests. Across 11 scored model runs, overall accuracy ranges from 43.70% to 56.36%, showing that the benchmark remains difficult for current strong models. Exact candidate-set construction yields consistently lower accuracy under stricter set-construction requirements, and recall diagnostics reveal asymmetric effects of constraint direction: increasing exclusion constraints lowers positive recall and raises negative recall, while increasing inclusion branches mainly erodes false-case rejection. These results suggest that concept-boundary execution is a measurable weakness under this diagnostic setup and that evaluating it requires factor-level diagnostics beyond aggregate accuracy. Code and data are available at https://github.com/RenjunLai/concept-boundary-benchmark.

1 Introduction

Large language models (LLMs) are increasingly used as natural-language interfaces for structured language-processing workflows: generative information extraction uses them to extract structured knowledge from natural-language text (Xu et al., 2024), while universal structure generation and grammar-constrained decoding cast structured NLP tasks as schema- or format-constrained generation (Lu et al., 2022; Geng et al., 2023). Schema-conditioned corpora and bilingual instruction datasets show that task schemas and instructions remain central to reliable information extraction (Gui et al., 2024b,a). As models are increasingly used in tool-using settings (Yao et al., 2025) and agentic workflows (Liu et al., 2024), whether early user intent and constraints are executed correctly can strongly affect downstream action selection and execution (OpenAI et al., 2025; GLM-4.5 Team et al., 2025). In such settings, users often express natural-language rules containing both inclusion and exclusion conditions; the challenge is whether models can apply these rules stably to candidate objects.

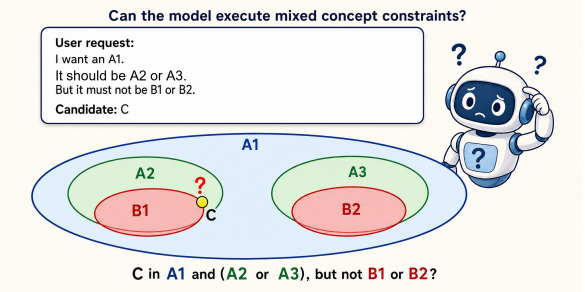


Figure 1: Abstract mixed-constraint setup. The model must decide whether candidate C belongs to A_1 and at least one of A_2 or A_3 while avoiding B_1 and B_2. The benchmark checks this candidate-level decision rather than the fluency of the model's verbal framing.

Prior work suggests two complementary directions. Negated prompts can substantially change model behavior (Jang et al., 2023), and large negation benchmarks show that negative knowledge can be tested with resource facts, templates, and distractors (García-Ferrero et al., 2023). At the same time, logic-form datasets bind natural-language reasoning examples to explicit formal

Findings of the Association for Computational Linguistics: IJCNLP-AACL 2026, pages 1797–1814
November 6-10, 2026 ©2026 Association for Computational Linguistics

representations (Han et al., 2024), and solver-augmented methods use symbolic forms and deterministic solvers for inference (Pan et al., 2023). These directions make it possible to turn user-style inclusion and exclusion rules into candidate-level judgments over resource objects.

Inclusion and exclusion rules define a target set: some candidates should remain inside the intended scope, while others should be removed from it. In our setting, the concept boundary is given jointly by a user-style rule and a backend resource; the model's task is not to regenerate this boundary, but to apply it to a single candidate or a candidate pool. Figure 1 illustrates this execution relationship. We define this ability as natural-language concept-set constraint execution and use it as an auditable diagnostic evaluation target. By using a resource backend to fix the object space, relation structure, and gold answers, we isolate this candidate-level operation from end-to-end factors such as planning, retrieval, and tool execution.

Each instance defines a positive concept scope and may include exclusion branches that remove auditable subregions from that scope. The gold answer is computed over backend resource objects and taxonomy relations rather than inferred from the surface prompt. More concretely, data construction first fixes hidden truth conditions and candidate realizations in the resource graph, and only then renders a natural-language prompt. This order makes model errors easier to localize across task form, constraint structure, and information exposure.

We instantiate this formulation in two resource families: a WordNet track (Miller, 1995) and a BabelNet track built from BabelNet synsets and BabelSememe filtering (Navigli and Ponzetto, 2012; Qi et al., 2020). The same hidden constraint is queried through three task views: single-object judgment, candidate filtering, and satisfiability judgment. The diagnostic matrix varies the number of positive concepts, the number of exclusion concepts, and the visible information level. Our contributions are:

- We propose and define natural-language concept-set constraint execution as candidate-level set-membership judgment over resource objects under inclusion and exclusion conditions.
- We construct and release three data products: a test set for quick validation, a main set for formal model evaluation, and a full set that preserves the complete generation frame. The main set covers two resource families, three task views, and three paired information levels, yielding 67,500 base samples and 202,500 formal evaluation requests.
- We evaluate eleven scored model runs on the same frozen request set, finding that exact candidate-set construction is substantially harder across models and that the numbers of included and excluded concepts produce directionally different effects on positive and negative recall.

2 Background

This section situates our benchmark along three lines of work: negation and exclusion research motivates testing whether models handle the direction of negative or exclusionary constraints (Jang et al., 2023; García-Ferrero et al., 2023); logic and satisfiability benchmarks motivate binding natural-language problems to checkable backend truth conditions (Han et al., 2024; Pan et al., 2023; Richardson and Sabharwal, 2022); and resource-backed semantic probing shows that lexical-semantic resources such as WordNet and BabelNet can provide stable object inventories, relation structures, and automatic labeling evidence (Miller, 1995; Navigli and Ponzetto, 2012; Qi et al., 2020). Our setting combines these requirements in a candidate-level task: the model receives natural-language inclusion and exclusion rules and, under a single resource-defined oracle, performs T1 single-object membership judgment, T2 exact candidate-set construction, and T3 candidate-pool satisfiability judgment. In contrast to AgentBench and τ-bench, which evaluate complete interaction, decision making, and tool use (Liu et al., 2024; Yao et al., 2025), our work measures concept-boundary execution at the candidate level.

2.1 Negation and Exclusion

Negation and exclusion are relevant because they can change what a model should accept or reject. Work on negated prompts shows that behavior under affirmative formulations does not reliably transfer to negated formulations (Jang et al., 2023). García-Ferrero et al. (2023) test negative knowledge with WordNet facts, affirmative and negative templates, and distractor substitutions. Mono-

tonicity work gives a related view of directionality: downward-entailing contexts change which replacements preserve entailment (Yanaka et al., 2019b). Neural models have also been shown to struggle with monotonicity reasoning (Yanaka et al., 2019a). These findings motivate evaluating inclusion and exclusion as directional constraints on candidate admissibility.

2.2 Checkable Truth Conditions

Logical-reasoning datasets provide construction patterns for linking natural language to explicit truth conditions. FOLIO annotates natural-language reasoning examples with first-order logic so that answers can be checked against formal representations (Han et al., 2024). Solver-augmented approaches such as Logic-LM translate language into symbolic forms and use deterministic solvers for inference (Pan et al., 2023). Natural-language satisfiability tasks likewise emphasize consistency and satisfiability as properties that can be validated outside the model response (Richardson and Sabharwal, 2022). In our setting, the gold answer is likewise fixed by a backend condition before it is rendered into the natural-language prompt seen by the model.

2.3 Resource-Backed Semantic Probing

Resource-backed semantic probing provides the object space for this kind of construction. García-Ferrero et al. (2023) instantiate WordNet relations with affirmative and negative templates and distractor substitutions, showing that lexical resources can support automatically labeled diagnostic data. In our WordNet track, noun synsets and hypernym-hyponym closure define the objects and concept sets used by the oracle (Miller, 1995). In the BabelNet track, BabelNet noun synsets provide a separate multilingual synset space (Navigli and Ponzetto, 2012), and BabelSememe supplies the agreement-filtered Chinese inventory used as a quality gate (Qi et al., 2020). In both tracks, glosses are visible information for the model, not sources of the gold answer. Because the two resource families differ in object identifiers, surface cleanliness, proper-name contamination, and gloss coverage, they should not be collapsed into a single translation-equivalent test set.

3 Benchmark

We introduce a diagnostic benchmark for evaluating how effectively LLMs execute natural-language concept-set constraints. The benchmark is instantiated in an English WordNet track and a Chinese BabelNet track. The two tracks share the same task abstraction, constraint matrix, information levels, and rendering policy, but use separate object identifiers, truth constructors, candidate generators, surface forms, and gloss fields.

3.1 Task Formulation

Each benchmark instance defines a constraint C over resource objects. The positive side contains a base concept A_1; when $a > 1$, $A_2, \ldots, A_a$ denote positive branches within the scope of A_1 and are interpreted disjunctively rather than as an additional taxonomic AND chain. We refer to a as the A-count. The negative side contains zero or more exclusion concepts $B_1, \ldots, B_b$, where b is the B-count, which remove subregions from the positive scope. In the main matrix, $a \in \{1, \ldots, 5\}$ and $b \in \{0, \ldots, 5\}$. Formally, the target set is $S(A_1)$ when $a = 1$, and $S(A_1) \cap (S(A_2) \cup \cdots \cup S(A_a))$ otherwise; a candidate satisfies C only if it is in this target set and outside all B sets. The recipe uses non-reflexive descendant closure, so $S(A)$ denotes descendants of A and does not automatically include A itself.

T1 asks whether a single candidate satisfies C. T2 asks the model to select all satisfying candidates from an explicit candidate pool. T3 asks whether at least one candidate in such a pool satisfies C. Thus T1, T2, and T3 are not separate benchmarks; they are three output views of the same resource-defined predicate. Table 1 summarizes the input, output, and oracle target for each view; Appendix D gives concrete rendered examples.

Each base sample is rendered under three paired information levels: I0 withholds concept and candidate glosses, I1 provides concept glosses, and I2 additionally provides candidate glosses. These levels change only the visible definition or gloss information, while the hidden constraint, candidate realization, and gold answer remain fixed.

3.2 Resource Foundations

The English track uses Princeton WordNet 3.0 and restricts the object space to noun synsets (Miller, 1995). Its internal truth unit is the synset identifier, and concept sets are derived from audited hypernym-hyponym closure. In our audit, WordNet contains 82,115 noun synsets, of which 57,717 remain usable after surface filtering, giv-

Task	Input	Output	Oracle
T1	single candidate	True/False	candidate in C
T2	candidate pool	index list	exact set
T3	candidate pool	True/False	any candidate in C

Table 1: The three task views query the same hidden predicate C with different candidate realizations and output protocols. Complete rendered examples are given in Appendix D.

Track	Objects	Usable	Truth unit
WordNet	82,115	57,717	WordNet noun synset
BabelNet	15,755	6,921	BabelNet noun synset

Table 2: Resource-family object spaces after task-specific filtering. Counts are reported separately because the two tracks are not translation-equivalent test sets.

ing 70.29% usable coverage. The Chinese track uses BabelNet noun synsets as the object space, but does not directly use all unfiltered BabelNet nodes (Navigli and Ponzetto, 2012). We adopt synsets retained after agreement filtering in the BabelSememe resource as a quality gate (Qi et al., 2020). We then apply task-specific filters over the retained BabelNet data, including noun-only filtering, proper-name exclusion, clean Chinese synonym availability, and Chinese gloss availability. The BabelNet-derived inventory contains 15,755 synsets, and the final usable Chinese object set after filtering contains 6,921 synsets, giving 43.93% usable coverage. Table 2 summarizes the resulting object-space sizes, usable-object counts, and truth units for the two resource tracks.

Both resource tracks are used only for non-commercial research purposes: WordNet through the NLTK interface under the Princeton WordNet license, and BabelNet-derived materials under BabelNet's non-commercial research-use terms. The benchmark uses lexical-semantic resource entries and taxonomy relations, and does not collect user information, private records, or personally identifiable information.

In both tracks, surface forms and glosses enter the prompt only after the hidden truth predicate has been computed. The gold answer is determined by object identifiers and audited taxonomy relations, not by surface forms or glosses. The two tracks are therefore reported separately rather than collapsed into a single aggregate, because their data sources, object scales, visible-material availability, language quality, and the tested models' English and Chinese capabilities differ.

3.3 Generation-Space Analysis and Sampling

A naive top-down route is not suitable as the formal generation entry point. If generation starts from broad concepts and enumerates possible lower-level concept combinations, the structural space explodes and natural-proportion sampling becomes dominated by a small number of broad or high-branching concepts. Our top-down audit estimates roughly 3.94×10^{11} structural constraint cores; expanding them across task views and information levels would produce roughly 3.54×10^{12} rendered requests. This scale rules out full expansion, and closure statistics further show why natural sampling would create structural concentration. WordNet usable objects have closure-size p50 of 0 and p99 of 30, but a maximum of 7,324; BabelNet usable objects have closure-size p50 of 0 and p99 of 389, but a maximum of 2,594.

We therefore use a seed-object-first route. It starts from candidate objects; for each seed cell, it chooses the nearest viable A_1 and constructs OR branches and B exclusions within that local scope. This disperses entry points over the object space, prevents samples from being dominated by large-A_1 combination counts, and keeps one default core per seed cell to avoid repeated expansion of the same seed under multiple broader A_1 choices. After audit, top-A_1 concentration remains visible but no longer dominates: the largest global shares are 3.90% for WordNet `person.n.01` and 3.36% for BabelNet `bn:00027546n`. The route still leaves sufficient capacity, with 1,078,687 feasible seed cells, 3,236,061 full-frame base samples, and about 9.71×10^6 full-frame rendered requests.

The evaluation set is designed for controlled stratified coverage. The main product first applies the positive-pool gate and then samples without replacement within strata defined by resource family, task, and constraint cell. Each task–constraint stratum draws 650 WordNet base samples and 100 BabelNet base samples. This ratio is chosen from the relative usable inventory sizes of the two resource families and from construction-control experience. It makes the sampling process stable to implement and transparent to verify, while supporting controlled comparison across English and Chinese tracks with different data scales. The audit shows no quota shortage under the mild positive-pool gate: the weakest eligible stratum still contains

Product	Base samples	Requests	Role
test	360	1,080	fast validation
main	67,500	202,500	formal evaluation
full	3,236,061	∼9.71M	release frame

Table 3: Dataset products. The main product is used for the model evaluation in this paper; the full product preserves the complete seed-object-first frame for release and future derivations.

23,114 WordNet cases and 2,779 BabelNet cases.

Across the paired information levels defined above, the hidden constraint, candidate realization, and gold answer remain fixed while only the visible definition or gloss information changes. Table 3 gives a main-text overview of the three dataset products, while Table 6 in Appendix A provides the exact per-family quotas and product-size definitions. The test product is used for fast validation, the main product is used for the formal model evaluation, and the full product preserves the complete seed-object-first frame. The release package includes the three dataset products, response data from the 11 scored runs, and the code needed for testing, validation, data generation, and analysis.

3.4 Construction Method

The construction method consists of resource audit, constraint construction, candidate realization, template rendering, and validation. Resource audit builds family-specific inventories of usable synsets, surfaces, glosses, and taxonomy edges, and records infeasible, shortage, budget-limited, and fallback states. Constraint construction starts from a seed object, searches upward for the lowest viable A_1, constructs positive OR branches and direct-depth B exclusion branches within it, and computes the A/B core and oracle truth. Infeasible seed cells are recorded in the audit rather than repaired by relaxing the truth rule.

Candidate realization occurs only after the constraint core and oracle truth have been fixed. T1 realizes the core as a single candidate object, while T2 and T3 realize it as an explicit candidate pool of fixed size 4. Pool size is a data-version setting; changing it would define a new data version or ablation. T1 and T3 are kept approximately balanced between true and false cases. T2 schedules gold set sizes 0, 1, and 2 and keeps the answer-size distribution as balanced as the feasible candidate pools allow. The prompt templates are manually designed to control task instructions, output format, and gloss layout, but the concepts, candidates, glosses, and gold answers in each request are filled automatically from frozen samples and resource fields.

Negative candidates are not sampled from an unconstrained random complement. For a constraint C, the system first computes positive pools and interpretable negative pools, and each negative candidate is assigned a backend truth-flag explanation for why it fails. This requires separating the provenance of exclusion constraints from the failure source of a candidate. In this recipe, exclusion constraints are mainly constructed from inside A_1 and inside the positive branch, while a negative candidate may fail because it hits an exclusion branch, fails the positive branch, comes from a near sibling, or falls outside A_1.

For each frozen sample, the oracle can recompute the gold answer from backend resource identifiers and the constraint core. The manifest records resource versions, renderer version, sampling weights, and necessary provenance. Coverage reports record product size, stratified coverage, answer distribution, top-A_1 concentration, seed and candidate reuse, positive and negative pool capacity, shortages, and fallbacks.

4 Experiments

4.1 Experimental Setup

All model evaluations included in the main results use the same frozen main rendered request set. The main product contains 67,500 base samples rendered into 202,500 requests under the three information levels. The three renderings of a base sample share the same hidden constraint, candidate realization, and gold answer, and differ only in the amount of visible definition or gloss information. T1 and T3 require a final True or False answer, while T2 requires a candidate-number list. The scoring protocol is answer-only: explanations, reasoning traces, and intermediate text are ignored by the formal parser. If the final answer is missing, unparseable, or incompatible with the task output protocol, the sample remains in the denominator and is scored as incorrect.

4.2 Models

We evaluate a fixed set of LLM runs spanning different model families, scales, and inference modes. The model set includes DeepSeek V4 Pro and Flash (DeepSeek-AI, 2026), Qwen3.5 27B

and 122B (Qwen Team, 2026), GLM 5.1 (GLM-5-Team et al., 2026), GLM 4.7 (GLM-4.5 Team et al., 2025), Gemma 4 31B (Google DeepMind, 2026), GPT-OSS 120B (OpenAI et al., 2025), and MiniMax M2.7 (MiniMax, 2026). Table 4 marks GLM 5.1 with "–" because the run was partially tested but API serving limitations prevented us from completing the full test.

Direct-answer and reasoning-enabled runs use the same requests, task prompts, output protocol, and parser. Within each inference mode, we keep the calling constraints as consistent as the model interfaces allow, and each model family uses its recommended decoding settings for the corresponding mode. Table 7 in Appendix G reports the concrete settings. When the same model has both direct-answer and reasoning-enabled results, we use the pair as a descriptive comparison; cross-family ordering is reported as performance comparison only.

4.3 Metrics

Let D be a reported slice of the main request set, y_i the oracle answer for request i, and $\hat{y}_i$ the parsed final answer. Accuracy is computed as:

$$\mathrm{Acc}(D) = \frac{|\{i \in D : \hat{y}_i = y_i\}|}{|D|}.$$

Overall accuracy uses all rendered requests in the main product. Task-level accuracy restricts D to T1, T2, or T3, and resource-family accuracy restricts D to WordNet or BabelNet. For T2, $\hat{y}_i$ and y_i are sets of candidate indices, and a prediction is correct only when the parsed set exactly matches the gold set; partial overlap receives no credit.

For the binary task views T1 and T3, we additionally report positive and negative recall. Positive recall measures acceptance of gold-true cases, and negative recall measures rejection of gold-false cases:

$$\mathrm{Recall}_+ = \frac{TP}{TP+FN},$$
$$\mathrm{Recall}_- = \frac{TN}{TN+FP}.$$

These recall values are computed within A/B constraint cells when we analyze directional effects. T2 is diagnosed separately with exact-set accuracy by gold set size and with predicted-set-size distributions.

T1 and T3 each contain 33,750 gold-true and 33,750 gold-false requests, so random guessing under a balanced binary prior and the majority-class anchor are both 50%. T2 contains 22,590, 22,500, and 22,410 requests with gold-set sizes 0, 1, and 2, respectively. T2 predicts a subset of four candidates; always returning the empty set, a fixed singleton, or a uniformly random subset induces different output priors, so there is no single random baseline independent of the chosen strategy. We report the frozen gold-set-size distribution as the calibration reference for interpreting exact-set accuracy.

4.4 Main Results

Table 4 reports the evaluated runs, grouped by model family. Overall accuracy ranges from 43.70% to 56.36%, with DeepSeek V4 Pro Think High obtaining the highest overall score. The leading scored rows mostly come from reasoning-enabled settings. A more controlled descriptive comparison comes from Qwen3.5: the 27B model improves from 47.28% in the direct-answer setting to 54.42% in the reasoning-enabled setting, and the 122B model improves from 48.73% to 54.55%. Model scaling is also visible within this family, especially in the direct-answer setting where Qwen3.5 122B outperforms Qwen3.5 27B; under the reasoning-enabled setting, however, the two overall scores are very close.

The task columns show a stable structural pattern: T2 yields consistently lower exact-set accuracy under stricter set-construction requirements. Protocol failures do not explain these overall low accuracies: median invalid-format rates are 0.023% on WordNet and 0.007% on BabelNet, and API errors affect 0.00139% of all formal evaluation records; Appendix J reports the run-level audit and valid-only sensitivity. Subsequent quantitative analyses use the same scored runs in Table 4. Appendix I gives a representative detailed report for Qwen3.5 27B Think High; the same report schema is available for all 11 scored runs.

4.5 Recall and Constraint Direction

The main results show that this benchmark is not easily solved even by current strong models. We next analyze how these errors vary with the number of inclusion and exclusion constraints. We use Qwen3.5 27B Think High as a representative run for A/B constraint-matrix visualization and verify the same direction against all scored runs. Figure 2 shows T1 positive and negative recall under different (a, b) conditions for this representative model.

The ordered B-count trend is stable across the 11 scored runs in both T1 and T3. After averaging

Model	Mode	T1	T2	T3	WordNet	BabelNet	Overall
DeepSeek V4 Pro	Think High	68.45	37.83	62.81	56.11	58.00	56.36
DeepSeek V4 Flash	Non-think	58.21	19.69	53.20	43.34	46.01	43.70
Qwen3.5 27B	Think High	67.19	34.12	61.95	53.89	57.82	54.42
Qwen3.5 27B	Non-think	59.57	24.90	57.35	46.23	54.09	47.28
Qwen3.5 122B	Think High	67.31	34.45	61.89	54.01	58.09	54.55
Qwen3.5 122B	Non-think	61.50	27.62	57.07	47.86	54.42	48.73
GLM 5.1	Thinking	–	–	–	–	–	–
GLM 4.7	Non-think	53.95	28.65	54.50	45.32	48.14	45.70
Gemma 4 31B	Thinking	67.25	33.67	58.56	52.42	58.01	53.16
Gemma 4 31B	Non-think	64.75	33.14	60.24	52.11	56.61	52.71
GPT-OSS 120B	Think High	66.55	33.15	61.27	53.08	57.37	53.66
MiniMax M2.7	Thinking	67.17	32.32	60.97	52.82	57.81	53.49

Table 4: Main answer-only accuracy results in percent. T1, T2, and T3 report task-level accuracy over both resource families; WordNet and BabelNet report resource-family accuracy. Best values in each numeric column are underlined. "–" indicates that GLM 5.1 was partially tested but not scored because API serving limitations prevented completion of the full run.

over A-count, all 22 run–task pairs have a negative linear slope for positive recall and a positive linear slope for negative recall; among 220 adjacent step comparisons, only two local reversals occur. Appendix J reports the run-level slopes and reversal counts. For the representative T1 run in Figure 2, positive recall drops from 63.4% at $b = 0$ to 48.3% at $b = 5$, while negative recall rises from 67.6% to 84.6%. The key failure is the drop in positive recall; the increase in negative recall shows that the drop is not generic degradation, but a shift toward rejection under stronger exclusion constraints.

A-count shows a different asymmetry. When we average over B-count and audit the ordered A-count sequence, the negative-recall slope is negative in all 22 run–task pairs, whereas the positive-recall slope is positive in only 17 pairs. A-count is therefore better described as rejection erosion: more inclusion branches consistently weaken the model's ability to reject false cases, but do not consistently improve recall on true cases. Because A_1 defines the outer scope while $A_2, \ldots, A_a$ are OR branches inside A_1, A-count is structurally non-homogeneous and should not be interpreted as a difficulty axis symmetric to B-count.

4.6 T2 Set-Size Diagnostics

The gold answer in T2 is a set of candidate indices, so the task should not be reduced to a binary label indicating whether the gold set is empty. The gold set size is a generation-control and diagnostic field that helps identify exact-set failures in empty-set, one-positive, and two-positive candidate pools; it does not replace the formal a/b/information axes.

Table 5 reports T2 exact-set accuracy by gold set size for six reasoning-enabled or thinking runs. All six runs show low accuracy on two-positive pools, and the pattern differs across empty and one-positive pools. These results show which set types are difficult, but do not by themselves explain why the model gives the wrong set.

Model	$\|G\| = 0$	$\|G\| = 1$	$\|G\| = 2$
DeepSeek V4 Pro	50.64	35.24	27.53
Qwen3.5 122B	35.35	43.31	24.63
Qwen3.5 27B	35.10	43.65	23.57
Gemma 4 31B	37.87	38.54	24.56
GPT-OSS 120B	40.28	33.47	25.64
MiniMax M2.7	37.50	33.50	25.92

Table 5: T2 exact-set accuracy by gold set size $|G|$ for six reasoning-enabled or thinking runs. Values are percentages.

Figure 3 therefore reports predicted set size conditioned on gold set size for the representative run. The model predicts a one-candidate set for 47.4% of empty-set cases, 64.2% of one-positive cases, and 48.6% of two-positive cases. This output-cardinality bias helps explain why T2 remains difficult even for strong reasoning-enabled runs. The main conclusion is narrow: T2 primarily exposes difficulty in complete set construction and output-cardinality control.

4.7 Robustness and Error Audit

As a boundary-sensitivity check, removing samples in which a candidate directly overlaps a constraint node improves all three tasks for all 11 runs in both resource families; when the WordNet slice is further restricted to monosemous candidates, the cross-run median T2 accuracy remains

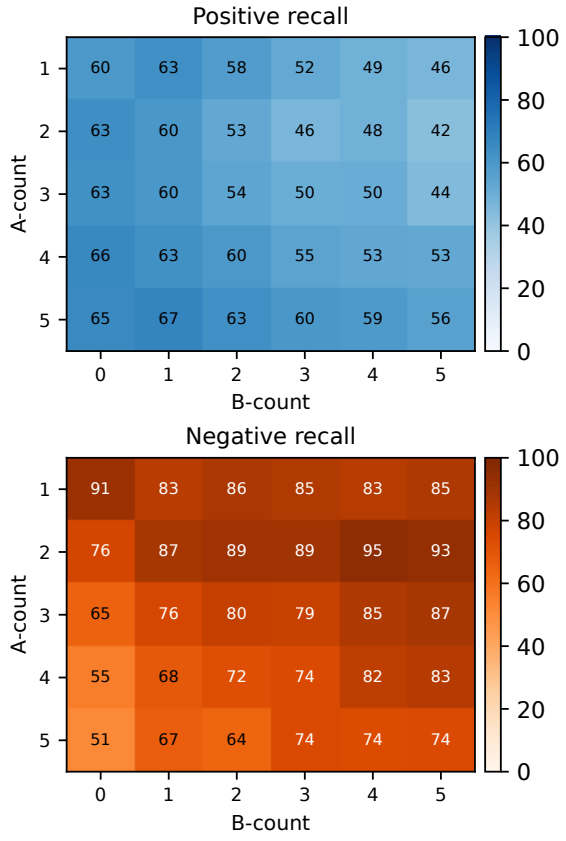


Figure 2: Representative T1 recall matrices for Qwen3.5 27B Think High. The top panel shows positive recall and the bottom panel shows negative recall. Positive recall decreases as B-count increases, while negative recall generally increases. Each cell prints the percentage value, so the trend does not depend on color alone.

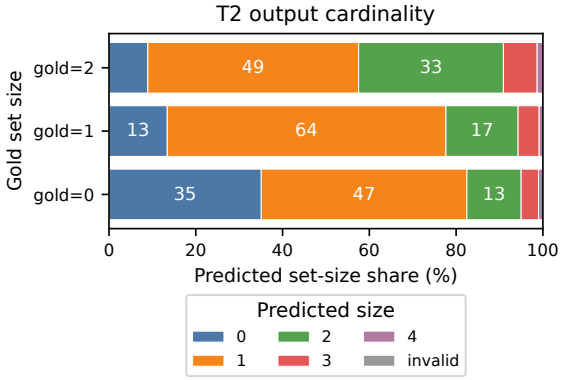


Figure 3: Predicted output set-size distribution for T2 in Qwen3.5 27B Think High. The model frequently predicts a single candidate, even when the gold set is empty or has size two.

48.50% (range: 27.60%–52.21%). Node boundaries and candidate polysemy affect difficulty but do not account for all constraint-execution errors; Appendix J reports the full results.

The cross-run error decomposition shows that T1 frequently accepts candidates that fail a required positive branch, with median false-positive rates of 29.39% on WordNet and 25.16% on BabelNet; T2 is dominated by over-selection in WordNet (a median 49.36% of errors) and omission in BabelNet (46.97%); and T3 contains both false positives and false negatives. Appendix J reports the full denominators, run ranges, and three cases.

5 Conclusion

This paper proposed and formalized natural-language concept-set constraint execution as a diagnostic target: the model must apply user-specified inclusion and exclusion constraints to concrete candidate objects. We built a dual-resource-family benchmark from WordNet and BabelNet, instantiated the same task abstraction in both resource tracks, and queried the same hidden constraint through T1 single-object judgment, T2 exact candidate-set construction, and T3 satisfiability judgment. The benchmark fixes the constraint core, candidate realization, and gold answer before rendering the prompt, and uses answer-only scoring so that model outputs can be audited against the resource backend.

The experiments show that the benchmark remains difficult for current strong models. Across the 11 scored runs, overall accuracy ranges from 43.70% to 56.36%; in every scored run, T2 exact-set accuracy is lower than T1 and T3, showing that complete candidate-set construction is harder than binary judgment. The T1 and T3 recall matrices further show that model errors are not only reflected in aggregate accuracy, but also in how inclusion and exclusion constraints affect acceptance and rejection behavior: B-count shows a stable ordered direction, while A-count mainly appears as erosion of false-case rejection.

The main contribution of this study is a reproducible and auditable evaluation framework, together with empirical findings about concept-set constraint execution. The framework keeps resource family, task view, information level, and constraint factors visible as analysis dimensions, allowing results to be localized to concrete generation conditions and error types. More importantly, it highlights the need to evaluate early constraint execution in structured information processing and agentic workflows: if a model cannot

reliably execute user-specified inclusion and exclusion conditions, downstream information extraction, candidate selection, tool use, or action planning may operate over an incorrect boundary. Future work can extend the framework to larger candidate pools, more natural prompt surfaces, longer contexts, multi-turn interactions, and verifier-assisted error analyses.

Limitations

The two resource tracks in this paper should not be interpreted as English and Chinese versions of the same test set. The WordNet track is grounded in English noun synsets and hypernym-hyponym closure, whereas the BabelNet track is grounded in filtered BabelNet noun synsets, Chinese surfaces, and Chinese glosses. They differ in object identifiers, taxonomy relation coverage, surface cleanliness, proper-name contamination, and gloss coverage. Family-first reporting is therefore an interpretive constraint of the benchmark rather than a presentation preference.

The benchmark covers a controlled single-turn, single-instance setting: a model executes a concept-set constraint governed by inclusion, exclusion, and information level. It does not cover long-context interaction, multi-turn dialogue, stacked demonstrations, complex coreference, free-form question answering, or all forms of logical and instruction-following behavior. The formal matrix also excludes pure exclusion, non-trivial overlap, depth, scope, and more natural surface rewrite variants. Relation types are also deliberately limited: although both resources provide other usable relations, this paper uses only hypernymy/hyponymy relations to keep the structural definition stable and avoid imposing additional external-knowledge requirements on the model. These restrictions reduce task coverage, but they provide stronger oracle auditability, candidate control, and factor-level interpretability.

Answer-only accuracy makes the main score depend on the final answer rather than on generated explanations or reasoning traces. This improves auditability, but it also limits interpretation: semantic errors, formatting errors, refusals, empty outputs, and failures to follow the answer protocol can all enter the error denominator. The positive and negative recall matrices for T1 and T3 help explain whether models are more likely to accept true cases or reject false cases, and how this tendency changes with A-count and B-count; however, these metrics do not explain why a model made a particular judgment. T2 exact-set evaluation is stricter because the model must return the complete correct candidate set; gold-set-size and predicted-set-size diagnostics can explain output-cardinality biases, but they do not provide a complete causal account.

The evaluated model set covers several current model families, scales, and inference modes, but it is not an exhaustive sample of the LLM space. Reasoning-enabled and direct-answer results can be used as descriptive comparisons, especially when they involve the same model and the same frozen requests; however, cross-family or cross-setting rankings should not be interpreted as pure causal effects of reasoning. All final claims about model performance must therefore be tied to the reported-result table rather than extrapolated from isolated cases to model families or reasoning mechanisms.

Ethical Considerations

This paper uses lexical-semantic resources as the oracle substrate, which makes gold answers auditable but does not make the resources unbiased or complete representations of world knowledge. WordNet, BabelNet, and related resources inherit biases from language coverage, cultural perspective, lexical curation, and maintenance history; their glosses may also encode cultural, definitional, and curation biases, and these glosses become visible prompt material at higher information levels. The BabelNet track is constructed from BabelNet synsets and existing resource-annotation filters, and does not rely on newly collected user text or personal data. However, because BabelNet was originally built with substantial automatic construction and encyclopedic sources, Chinese surfaces, gloss coverage, proper-name contamination, and cross-resource linking may introduce noise; gloss noise is a particularly important concern. Filtering, auditing, and family-first reporting reduce these risks, but they cannot eliminate resource noise or coverage bias. Gold answers in this benchmark should therefore be understood as internally auditable labels, not as final judgments about social meaning or real-world category boundaries. The release documentation includes provenance, filtering rules, and resource-family identifiers so that users can audit sample

origin and scope.

Model-evaluation results can be affected by model version, decoding settings, answer budget, serving conditions, failure handling, retry policy, and parsing protocol. Even when the same frozen requests are used, different run settings can affect scores and reproducibility. This paper and the release documentation transparently report model versions, main inference settings, failure handling, and the output-parsing protocol. Discussions of model failures are tied to the evaluated versions, settings, and benchmark conditions rather than generalized into stable properties of entire model families.

The public release entry covers the benchmark data, response data from the 11 scored runs, the generation rules, and the related code to support auditability, reproducibility, and follow-up error analysis. Fixed requests may enter subsequent training data or encourage optimization for the benchmark's answer format and templates; subsequent comparisons should therefore disclose possible data exposure and generate new versioned splits from the public recipe when needed.

Acknowledgments

This work was supported by the 2023 National College Students' Innovation and Entrepreneurship Training Program (Project No. 202310336009). We thank Professor Xiaoliang Xu for his guidance and support during the early exploratory stage of this work. Some of the computational resources used in this work were provided by Westlake University.

References

DeepSeek-AI. 2026. DeepSeek-V4: Towards Highly Efficient Million-Token Context Intelligence.

Iker García-Ferrero, Begoña Altuna, Javier Alvez, Itziar Gonzalez-Dios, and German Rigau. 2023. This is not a Dataset: A Large Negation Benchmark to Challenge Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8596–8615, Singapore. Association for Computational Linguistics.

Saibo Geng, Martin Josifoski, Maxime Peyrard, and Robert West. 2023. Grammar-Constrained Decoding for Structured NLP Tasks without Finetuning. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 10932–10952, Singapore. Association for Computational Linguistics.

GLM-4.5 Team, Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, Jiajie Zhang, Kedong Wang, Lucen Zhong, Mingdao Liu, Rui Lu, Shulin Cao, Xiaohan Zhang, Xuancheng Huang, Yao Wei, and 152 others. 2025. GLM-4.5: Agentic, Reasoning, and Coding (ARC) Foundation Models. *Preprint*, arXiv:2508.06471.

GLM-5-Team, Aohan Zeng, Xin Lv, Zhenyu Hou, Zhengxiao Du, Qinkai Zheng, Bin Chen, Da Yin, Chendi Ge, Chenghua Huang, Chengxing Xie, Chenzheng Zhu, Congfeng Yin, Cunxiang Wang, Gengzheng Pan, Hao Zeng, Haoke Zhang, Haoran Wang, Huilong Chen, and 167 others. 2026. GLM-5: From Vibe Coding to Agentic Engineering. *Preprint*, arXiv:2602.15763.

Google DeepMind. 2026. Gemma 4. https://deepmind.google/models/gemma/gemma-4/.

Honghao Gui, Shuofei Qiao, Jintian Zhang, Hongbin Ye, Mengshu Sun, Lei Liang, Jeff Z. Pan, Huajun Chen, and Ningyu Zhang. 2024a. InstructIE: A Bilingual Instruction-based Information Extraction Dataset. In *The Semantic Web – ISWC 2024*, pages 59–79, Cham. Springer Nature Switzerland.

Honghao Gui, Lin Yuan, Hongbin Ye, Ningyu Zhang, Mengshu Sun, Lei Liang, and Huajun Chen. 2024b. IEPile: Unearthing Large Scale Schema-Conditioned Information Extraction Corpus. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 127–146, Bangkok, Thailand. Association for Computational Linguistics.

Simeng Han, Hailey Schoelkopf, Yilun Zhao, Zhenting Qi, Martin Riddell, Wenfei Zhou, James Coady, David Peng, Yujie Qiao, Luke Benson, Lucy Sun, Alexander Wardle-Solano, Hannah Szabó, Ekaterina Zubova, Matthew Burtell, Jonathan Fan, Yixin Liu, Brian Wong, Malcolm Sailor, and 16 others. 2024. FOLIO: Natural Language Reasoning with First-Order Logic. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22017–22031, Miami, Florida, USA. Association for Computational Linguistics.

Joel Jang, Seonghyeon Ye, and Minjoon Seo. 2023. Can Large Language Models Truly Understand Prompts? A Case Study with Negated Prompts. In *Proceedings of The 1st Transfer Learning for Natural Language Processing Workshop*, pages 52–62. PMLR.

Xiao Liu, Hao Yu, Hanchen Zhang, Yifan Xu, Xuanyu Lei, Hanyu Lai, Yu Gu, Hangliang Ding, Kaiwen Men, Kejuan Yang, Shudan Zhang, Xiang Deng, Aohan Zeng, Zhengxiao Du, Chenhui Zhang, Sheng Shen, Tianjun Zhang, Yu Su, Huan Sun, and 3 others. 2024. AgentBench: Evaluating LLMs as Agents.

International Conference on Learning Representations, 2024:52989–53046.

Yaojie Lu, Qing Liu, Dai Dai, Xinyan Xiao, Hongyu Lin, Xianpei Han, Le Sun, and Hua Wu. 2022. Unified Structure Generation for Universal Information Extraction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5755–5772, Dublin, Ireland. Association for Computational Linguistics.

George A. Miller. 1995. WordNet: A lexical database for English. *Commun. ACM*, 38(11):39–41.

MiniMax. 2026. MiniMax M2.7: Early Echoes of Self-Evolution. https://www.minimax.io/news/minimax-m27-en.

Roberto Navigli and Simone Paolo Ponzetto. 2012. BabelNet: The automatic construction, evaluation and application of a wide-coverage multilingual semantic network. *Artificial Intelligence*, 193:217–250.

OpenAI, Sandhini Agarwal, Lama Ahmad, Jason Ai, Sam Altman, Andy Applebaum, Edwin Arbus, Rahul K. Arora, Yu Bai, Bowen Baker, Haiming Bao, Boaz Barak, Ally Bennett, Tyler Bertao, Nivedita Brett, Eugene Brevdo, Greg Brockman, Sebastien Bubeck, Che Chang, and 107 others. 2025. Gpt-oss-120b & gpt-oss-20b Model Card. *Preprint*, arXiv:2508.10925.

Liangming Pan, Alon Albalak, Xinyi Wang, and William Wang. 2023. Logic-LM: Empowering Large Language Models with Symbolic Solvers for Faithful Logical Reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 3806–3824, Singapore. Association for Computational Linguistics.

Fanchao Qi, Liang Chang, Maosong Sun, Sicong Ouyang, and Zhiyuan Liu. 2020. Towards Building a Multilingual Sememe Knowledge Base: Predicting Sememes for BabelNet Synsets. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8624–8631.

Qwen Team. 2026. Qwen3.5: Accelerating Productivity with Native Multimodal Agents. https://qwen.ai/blog?id=qwen3.5.

Kyle Richardson and Ashish Sabharwal. 2022. Pushing the Limits of Rule Reasoning in Transformers through Natural Language Satisfiability. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(10):11209–11219.

Derong Xu, Wei Chen, Wenjun Peng, Chao Zhang, Tong Xu, Xiangyu Zhao, Xian Wu, Yefeng Zheng, Yang Wang, and Enhong Chen. 2024. Large language models for generative information extraction: A survey. *Frontiers of Computer Science*, 18(6):186357.

Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, Kentaro Inui, Satoshi Sekine, Lasha Abzianidze, and Johan Bos. 2019a. Can Neural Networks Understand Monotonicity Reasoning? In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 31–40, Florence, Italy. Association for Computational Linguistics.

Hitomi Yanaka, Koji Mineshima, Daisuke Bekki, Kentaro Inui, Satoshi Sekine, Lasha Abzianidze, and Johan Bos. 2019b. HELP: A Dataset for Identifying Shortcomings of Neural Models in Monotonicity Reasoning. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 250–255, Minneapolis, Minnesota. Association for Computational Linguistics.

Shunyu Yao, Noah Shinn, Pedram Razavi, and Karthik Narasimhan. 2025. τ-bench: A Benchmark for Tool-Agent-User Interaction in Real-World Domains. *International Conference on Learning Representations*, 2025:9965–10017.

A Dataset Product Definitions

The benchmark has three frozen data products. The product names are not scale labels chosen after running models; they define distinct roles in the construction contract. The `test` product is a small but complete validation product, the `main` product is the formal model-evaluation product used in this paper, and the `full` product is the full seed-object-first frame retained for release, audit, and future derivations.

For sampled products, the scale is determined by task count, constraint-cell count, family quota, and information-level count. There are three task views, 30 constraint cells from the 5×6 A/B matrix, and three information levels. The sampled-product formulas are:

```
base samples = task count × constraint cells × sum(family quotas)
rendered requests = base samples × information levels
```

The family quotas are summed inside each task–constraint stratum; the number of resource families is not itself a multiplier. This distinction matters because the WordNet and BabelNet tracks have unequal object spaces and unequal quotas.

The `test` product follows the same stratification as `main`: all three tasks, all 30 A/B cells, both resource families, and all three information levels are present. Its scale is $3 \times 30 \times (3 + 1) = 360$ base samples and $360 \times 3 = 1{,}080$ rendered requests. It is intended to validate generation, rendering, parsing, and reporting before larger calls are

Product	WordNet quota	BabelNet quota	Base samples	Requests	Role
test	3	1	360	1,080	validation and sanity check
main	650	100	67,500	202,500	formal model evaluation
full	–	–	3,236,061	9,708,183	complete release frame

Table 6: Exact product definitions used by the benchmark. The sampled products use per-family quotas per task–constraint stratum; the full product is not quota-sampled.

made. The `main` product uses the same formula with quotas 650 and 100, yielding $3 \times 30 \times (650 + 100) = 67{,}500$ base samples and 202,500 rendered requests. The `full` product is not a model-testing product in this paper. It materializes the complete seed-object-first frame under the current feasibility and positive-pool gates, and can be used to derive future samples without changing the core oracle.

B Resource Filtering and Track Boundaries

The WordNet track and the BabelNet track are separate resource-family instantiations of the same task abstraction. They should not be read as translation-equivalent test sets. WordNet supplies English noun synsets, surface forms, definitions, and hypernym-hyponym closure. The usable WordNet object set contains 57,717 noun synsets after the benchmark's surface and feasibility filters.

The BabelNet track uses BabelNet noun synsets, surfaces, glosses, and taxonomy information. A prior BabelSememe resource provides a quality gate by identifying BabelNet synsets retained after agreement filtering, and we then apply benchmark-specific filters over the retained BabelNet data. These include noun filtering, proper-name exclusion, availability of a clean Chinese surface, and availability of a Chinese gloss. This process does not add HowNet definitions or HowNet surfaces to the prompt material; the surface forms and glosses used by the benchmark come from the retained BabelNet data. The final usable BabelNet object set contains 6,921 synsets.

The gold answer is never derived from the natural-language surface alone. In both tracks, the oracle is computed over backend resource identifiers and audited relations, and the prompt surface is rendered only after the hidden truth condition and candidate realization have been frozen. This is why the benchmark can expose disagreement between fluent verbal framing and candidate-level execution: the model does not get to determine the gold boundary by paraphrasing the prompt.

The two tracks differ in object scale, taxonomy density, surface cleanliness, proper-name contamination, gloss availability, and language quality. They also interact differently with model capabilities because the visible prompt material is English in the WordNet track and Chinese in the BabelNet track. Reporting therefore keeps resource-family accuracy separate from overall accuracy.

C Constraint Cells and Prompt Views

The constraint matrix has five A-count values and six B-count values. A_1 defines the outer positive scope. When the A-count is greater than one, $A_2, \ldots, A_a$ are positive branches inside A_1 and are interpreted as an OR condition. B concepts are exclusion branches that remove regions from the positive scope. The target predicate is therefore a set-membership predicate rather than a surface-language preference:

> Accept a candidate only if it belongs to the positive target set and does not belong to any excluded B region.

The three task views ask the same hidden predicate in different output forms. T1 presents a single candidate and requires a True/False answer. T2 presents a pool of four candidates and requires the exact set of satisfying candidate indices. T3 presents a pool of four candidates and requires a True/False answer indicating whether the pool contains at least one satisfying candidate.

Each base sample is rendered under the same three information levels used in the main product. I0 withholds concept and candidate glosses, forcing the model to rely on surface forms and its internal knowledge. I1 provides gloss information for the constraint concepts, approximating a normal prompt-engineering setting in which the user defines key concepts. I2 additionally provides candidate gloss information, approximating a more agentic setting in which an external retrieval step supplies more candidate-side information. These

levels are paired by `base_sample_id`: the three renderings are different views of the same hidden constraint and candidate realization, not independent backend samples.

The rendered templates enforce short answer-only protocols. A schematic T1 prompt can be represented as:

> **Task.** Decide whether the candidate satisfies the concept constraint.
>
> **Constraint.** The candidate must be inside A and outside B1, ..., Bb.
>
> **Candidate.** C.
>
> **Answer format.** Return only True or False.

T2 and T3 use the same hidden constraint but replace the single candidate with a numbered candidate pool. T2 requires a list of all satisfying candidate numbers, while T3 requires only whether at least one satisfying candidate exists. The formal scoring code parses only the final answer and does not score explanations.

D Representative Task Instances

This section shows the complete rendered form of one I2 request for each task view. All three examples are taken from the WordNet main product and use the same answer-only protocols as the evaluated requests. The prompt text is shown as the model receives it, with the gold answer reported after each prompt.

D.1 T1: Single-Candidate Judgment

Request. `request_baa3a3fee6690268`. **Gold answer.** True.

> You are given a concept-boundary question.
>
> **Target condition**
>
> - A1 doctor is the outer required concept.
> - The target object must also be a kind of at least one concept below.
> - A2 specialist.
> - Exclude objects that are a kind of any concept below.
> - B1 surgeon.
>
> **Candidate**
>
> - rheumatologist.
>
> **Concept notes**
>
> - A1 doctor: a licensed medical practitioner.
> - A2 specialist: practices one branch of medicine.
> - B1 surgeon: a physician who specializes in surgery.
>
> **Candidate notes**
>
> - Candidate rheumatologist: a physician specializing in rheumatic diseases.
>
> **Question.** Does the candidate satisfy the target condition?
>
> **Required answer.** Begin with True or False.

D.2 T2: Exact Candidate-Set Construction

Request. `request_160f7d009f847d04`. **Gold answer.** `[1]`.

> You are given a concept-boundary question.
>
> **Target condition**
>
> - A1 creator is the outer required concept.
> - The target object must also be a kind of at least one concept below.
> - A2 maker.
> - Exclude objects that are a kind of any concept below.
> - B1 artist.
>
> **Candidates**
>
> - 1. generator.
> - 2. songwriter.
> - 3. constructivist.
> - 4. colorist.
>
> **Concept notes**
>
> - A1 creator: a person who grows or makes or invents things.
> - A2 maker: a person who makes things.
> - B1 artist: a person whose creative work shows sensitivity and imagination.
>
> **Candidate notes**
>
> - 1. generator: someone who originates or causes or initiates something.
> - 2. songwriter: a composer of words or music for popular songs.
> - 3. constructivist: an artist of the school of constructivism.
> - 4. colorist: a painter able to achieve special effects with color.
>
> **Required answer.** Return only the list of candidate numbers that satisfy the target condition, for example `[1, 3]` or `[]`.

D.3 T3: Pool Satisfiability Judgment

Request. `request_d3a7ec14e1f7f178`. **Gold answer.** True.

> You are given a concept-boundary question.
>
> **Target condition**
>
> - A1 substance is the outer required concept.
> - The target object must also be a kind of at least one concept below.
> - A2 antigen.
> - Exclude objects that are a kind of any concept below.
> - B1 food.
>
> **Candidates**

- 1. Rh.
- 2. cordwood.
- 3. foodstuff.
- 4. frittata.

Concept notes

- A1 substance: a particular kind or species of matter with uniform properties.
- A2 antigen: any substance (as a toxin or enzyme) that stimulates an immune response in the body (especially the production of antibodies).
- B1 food: any substance that can be metabolized by an animal to give energy and build tissue.

Candidate notes

- 1. Rh: a blood group antigen possessed by Rh-positive people; if an Rh-negative person receives a blood transfusion from an Rh-positive person it can result in hemolysis and anemia.
- 2. cordwood: firewood cut and stacked in cords; wood sold by the cord.
- 3. foodstuff: a substance that can be used or prepared for use as food.
- 4. frittata: Italian omelet with diced vegetables and meats; cooked until bottom is set then inverted into another pan to cook the top.

Question. Does at least one candidate satisfy the target condition?

Required answer. Begin with True or False. True means at least one candidate satisfies the target condition.

E Candidate Realization and Answer Balance

Candidate realization occurs after the constraint core has been fixed. This ordering prevents candidate selection from being driven by attractive surface strings or by model-generated rationales. For T1, the realization step selects one candidate whose truth value is determined by the backend oracle. For T2 and T3, the realization step constructs a fixed pool of four candidates. The pool size is fixed in the current product definition; changing it would constitute a new benchmark version or a separate ablation.

T2 controls the number of satisfying candidates in the pool. Gold set sizes 0, 1, and 2 are scheduled as diagnostic conditions because exact-set construction has different failure modes when the correct output is empty, singleton, or multi-item. T3 is binary, but it uses the same candidate-pool machinery and asks whether any satisfying candidate is present. T1 and T3 true/false cases are controlled by the generation recipe and are not left to accidental prevalence in the resource.

Negative candidates are not sampled from an unconstrained complement. For each constraint C, the system computes positive pools and interpretable negative pools, and each negative candidate has a backend explanation of why it fails. A negative candidate may hit an exclusion branch, fail the positive branch requirement, come from a near sibling, or fall outside A_1. This distinction is important because B concepts are exclusion constraints, whereas a false candidate can fail for several resource-structural reasons.

The construction audit includes answer balance, candidate reuse, seed reuse, positive-pool capacity, negative-pool capacity, shortages, fallbacks, and top-A_1 concentration. In the scheduler probe reported during construction, 668 of 800 representative core rows passed the positive-pool gate and expanded to 2,004 base task views. The tested candidate-scheduling strategies validated without skipped samples or fallbacks, and T1/T3 answer balance and T2 gold-size balance were within one sample per cell. These audits support the use of controlled main-set comparisons rather than uncontrolled natural-prevalence sampling.

F Sampling Weights and Coverage Diagnostics

The full seed-object-first frame is not sampled by a proportional top-down process. A top-down audit estimates roughly 3.94×10^{11} structural constraint cores and roughly 3.54×10^{12} rendered requests after task and information expansion. This space is too large for direct publication or API evaluation, and natural-proportion sampling would overweight broad concept nodes and high-branching subtrees.

The sampled products therefore use post-gate stratified sampling. Each stratum is defined by resource family, task view, and A/B constraint cell. The main product applies a mild positive-pool gate of `final_positive_count >= 5` before sampling. Within each eligible stratum h, the inclusion probability and sampling weight are:

$$\pi_i = n_h/N_h, \quad w_i = N_h/n_h.$$

Here N_h is the post-gate eligible population for stratum h, and n_h is the frozen quota. The main product therefore estimates the post-gate sampled target population, not the entire full frame that also includes smaller positive pools. The full product keeps smaller positive-pool cases with

Model family	Mode	Temp./top-p
DeepSeek V4 Flash	Non-think	1.0 / 1.0
DeepSeek V4 Pro	Think High	1.0 / 1.0
Qwen3.5	Non-think	0.7 / 0.8
Qwen3.5	Think High	1.0 / 0.95
GLM 4.7	Non-think	1.0 / 0.95
GLM 5.1	Thinking	1.0 / 0.95
Gemma 4 31B	Non-think	1.0 / 0.95
Gemma 4 31B	Thinking	1.0 / 0.95
GPT-OSS 120B	Think High	1.0 / 1.0
MiniMax M2.7	Thinking	1.0 / 0.95

Setting	Answer	Reasoning	Control
DeepSeek V4 Flash / Non-think	128	–	disabled
DeepSeek V4 Pro / Think High	32768	32768	high effort
Qwen3.5 / Non-think	128	–	disabled
Qwen3.5 / Think High	32768	32768	enabled
GLM 4.7 / Non-think	128	–	disabled
GLM 5.1 / Thinking	32768	32768	enabled
Gemma 4 31B / Non-think	128	–	disabled
Gemma 4 31B / Thinking	32768	32768	enabled
GPT-OSS 120B / Think High	32768	32768	high effort
MiniMax M2.7 / Thinking	32768	32768	uncontrolled

Table 7: Model calling settings used for the evaluated modes. Budgets are token limits when exposed by the interface, except that saved reasoning traces are additionally truncated for reporting when required by the run configuration. MiniMax is listed as thinking because the model itself is a hybrid-thinking model and does not support active control over whether reasoning is used.

`final_positive_count >= 1` as part of the release frame, but these cases are not silently mixed into the formal main-set model score.

Coverage diagnostics are intended to make concentration visible rather than invisible. The construction audit reports the weakest eligible main-product strata, top-A_1 shares, closure-size summaries, and quota shortages. The main product has no quota shortage under the stated gates: the weakest eligible stratum still has 23,114 WordNet cases and 2,779 BabelNet cases. The largest post-route top-A_1 shares remain visible but do not dominate the product, at 3.90% for WordNet `person.n.01` and 3.36% for BabelNet `bn:00027546n`.

G Model Calling Settings

All runs use the same frozen request set and the same answer parser. Within each inference mode, we keep the main limits aligned where the interfaces allow. Direct-answer runs use short final-answer budgets; reasoning-enabled runs use larger budgets and high-effort reasoning controls when the model interface exposes them. Decoding parameters follow the recommended settings for each model family and mode. Table 7 reports the concrete calling settings used for these evaluated modes.

Slice	Total	Correct	Accuracy
Overall	202,500	110,196	54.42
T1	67,500	45,350	67.19
T2	67,500	23,032	34.12
T3	67,500	41,814	61.95
WordNet	175,500	94,585	53.89
BabelNet	27,000	15,611	57.82
I0	67,500	35,609	52.75
I1	67,500	35,942	53.25
I2	67,500	38,645	57.25

Table 8: Representative detailed evaluation summary for Qwen3.5 27B Think High. Accuracy values are percentages.

H Evaluation Protocol

The evaluation protocol is answer-only. For T1 and T3, the parser expects a final True or False answer. For T2, it expects a list of candidate indices. If the final answer is missing, empty, malformed, incompatible with the task, or a refusal rather than a valid answer, the request remains in the denominator and is counted as incorrect. Reasoning traces and explanations are not scored.

I Representative Detailed Evaluation Report

Table 8 reports the main fields from the detailed report for Qwen3.5 27B Think High. The same report schema is available for all 11 scored runs. Figures 4 and 5 show the T1 and T3 binary diagnostic matrices.

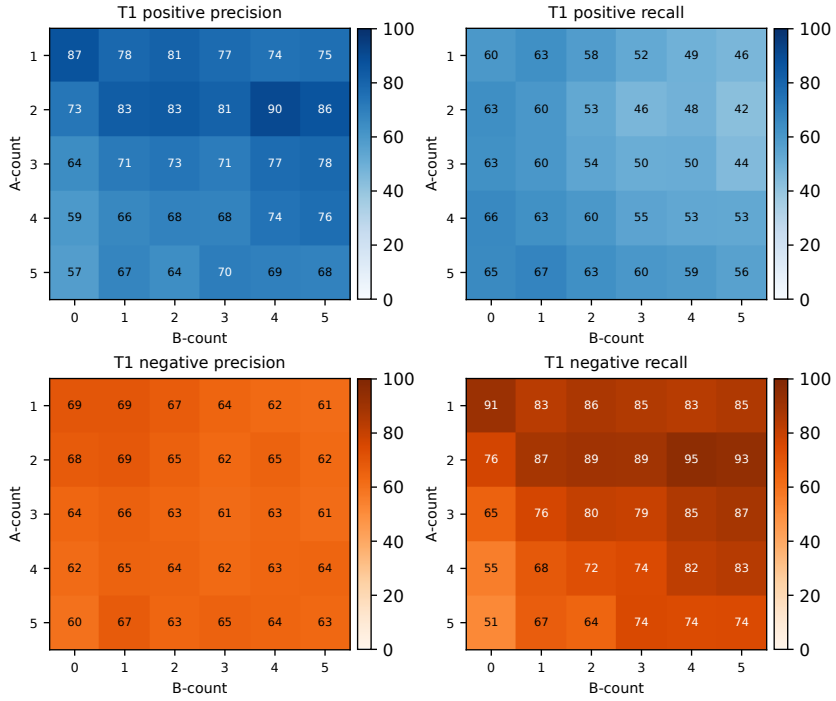


Figure 4: Representative T1 binary diagnostics for Qwen3.5 27B Think High. The four panels report positive precision, positive recall, negative precision, and negative recall.

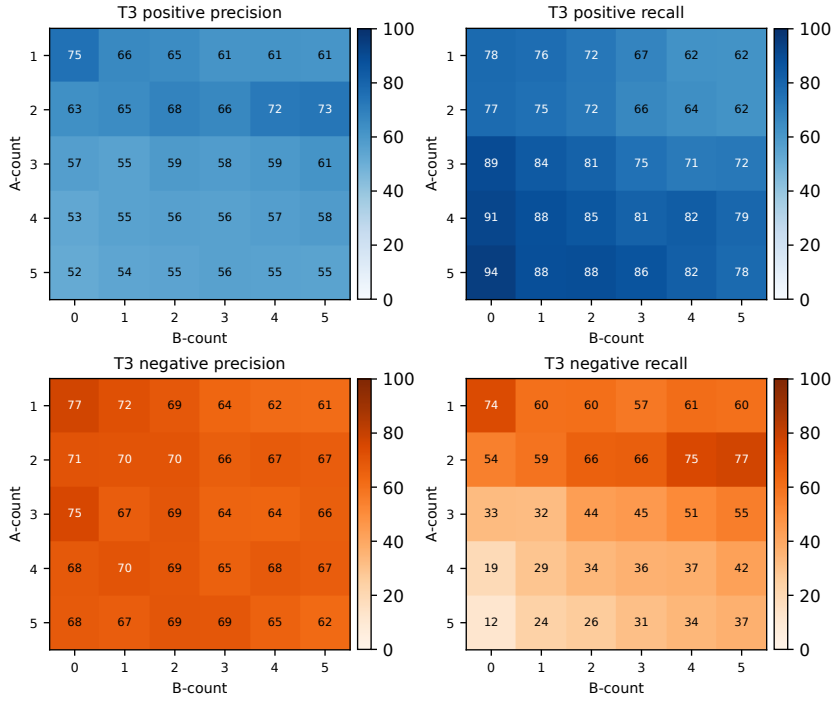


Figure 5: Representative T3 binary diagnostics for Qwen3.5 27B Think High. The four panels report positive precision, positive recall, negative precision, and negative recall.

J Cross-Run Robustness and Error Audits

J.1 Protocol and Boundary Robustness

This section provides the run-level and aggregate evidence for the robustness and error audits summarized in the main text. Invalid-format rates in Table 9 are computed over all requests in the corresponding resource; the no-overlap slice removes base samples in which a candidate directly overlaps a constraint node; and the monosemous condition is applied only to WordNet and requires every candidate in a sample to have a single noun sense.

After invalid-format outputs are excluded, the WordNet accuracy increase is at most 0.084 percentage points for ten runs; the exception, Gemma 4 31B Thinking, increases by about 1.44 points. Across the 2,227,500 formal evaluation records from all 11 runs, 31 (0.00139%) end in API errors before a scorable answer is produced and remain incorrect under the common protocol. Table 10 reports the corresponding cross-run results on the full data and boundary-robustness slices.

Run	WordNet invalid	BabelNet invalid
DeepSeek V4 Flash / Direct	0.011%	0.067%
DeepSeek V4 Pro / Thinking	0.098%	0.019%
GLM 4.7 / Direct	0.185%	0.133%
GPT-OSS 120B / Thinking	0.007%	0.007%
Gemma 4 31B / Direct	0.045%	0.107%
Gemma 4 31B / Thinking	2.669%	0.007%
MiniMax M2.7 / Hybrid	0.047%	0.000%
Qwen3.5 122B / Direct	0.001%	0.044%
Qwen3.5 122B / Thinking	0.001%	0.007%
Qwen3.5 27B / Direct	0.001%	0.000%
Qwen3.5 27B / Thinking	0.023%	0.000%

Table 9: Invalid-format rates by resource family for the 11 scored runs. Final answers that are unparseable or incompatible with the task output protocol remain incorrect in the formal accuracy.

Slice	Stat.	T1	T2	T3
WN full	Median	66.49	32.04	59.65
	Range	54.14–68.65	19.24–37.28	52.98–62.40
WN no-overlap	Median	68.95	36.32	61.90
	Range	55.01–71.07	21.19–42.55	54.03–65.20
WN no-overlap + mono.	Median	72.19	48.50	70.04
	Range	56.22–73.81	27.60–52.21	57.03–72.60
BN full	Median	66.33	39.52	65.46
	Range	52.72–68.16	22.63–41.43	54.59–66.23
BN no-overlap	Median	68.95	45.13	68.47
	Range	53.55–71.01	25.01–46.95	55.76–69.53

Table 10: Cross-run median accuracy and run ranges on the full data and boundary-robustness slices (all values are percentages). WN and BN denote WordNet and BabelNet. The WordNet no-overlap + monosemous slice retains 7,405, 1,081, and 989 base samples for T1, T2, and T3, respectively; the WordNet monosemy definition is not applied to BabelNet.

J.2 Observable Error Types

Table 11 reports cross-run medians and run ranges for observable error types by task and resource. T1 rates use gold-false requests of the corresponding source as the denominator, T2 shares use all T2 errors in each run as the denominator, and T3 rates use all requests of the corresponding task as the denominator.

J.3 Recall Trends Across Runs

For each B-count, we first aggregate over A-count and information levels and then fit the linear slope of recall over B-count. Positive-recall slopes are negative and negative-recall slopes are positive in all 22 run–task pairs; among 220 adjacent B-count steps, only two local changes move against the overall ordered direction. The slopes reported in Table 12 are percentage-point changes per additional B constraint.

J.4 Qualitative Error Cases

Table 13 summarizes the key fields and cross-run outputs of three frozen I2 requests. All final outputs are format-valid; error types are determined only from the resource oracle and parsed final answers.

Task	Resource	Error type	Median	Run range
T1	WordNet	fails positive branch, false-positive rate	29.39%	8.89–48.92%
T1	WordNet	hits B, false-positive rate	13.49%	4.13–38.33%
T1	WordNet	near sibling, false-positive rate	19.63%	3.15–30.85%
T1	BabelNet	fails positive branch, false-positive rate	25.16%	5.51–38.65%
T1	BabelNet	hits B, false-positive rate	7.99%	0.54–25.07%
T1	BabelNet	near sibling, false-positive rate	5.52%	0.76–19.81%
T2	WordNet	over-selection only, share of errors	49.36%	43.97–58.71%
T2	WordNet	mixed FP/FN, share of errors	24.72%	17.62–31.72%
T2	WordNet	omission only, share of errors	26.84%	11.80–34.16%
T2	BabelNet	over-selection only, share of errors	35.62%	29.63–51.47%
T2	BabelNet	mixed FP/FN, share of errors	17.21%	14.57–31.32%
T2	BabelNet	omission only, share of errors	46.97%	16.99–52.72%
T3	WordNet	false-positive request rate	26.53%	12.01–30.01%
T3	WordNet	false-negative request rate	10.97%	9.20–33.46%
T3	BabelNet	false-positive request rate	17.88%	6.20–23.71%
T3	BabelNet	false-negative request rate	16.99%	13.93–39.14%

Table 11: Observable error types for the three tasks in the two resource families. Values are medians and ranges over the 11 scored runs; the denominators for the tasks are defined in the accompanying text.

Run	T1 positive	T1 negative	T3 positive	T3 negative	Reversals
DeepSeek V4 Flash / Direct	-6.282	+6.634	-5.658	+5.462	0
DeepSeek V4 Pro / Thinking	-3.848	+3.544	-4.619	+4.590	0
GLM 4.7 / Direct	-2.712	+1.968	-6.161	+4.995	0
GPT-OSS 120B / Thinking	-4.572	+4.232	-3.757	+3.840	0
Gemma 4 31B / Direct	-2.918	+3.698	-4.099	+4.535	1
Gemma 4 31B / Thinking	-5.053	+4.393	-5.438	+3.771	0
MiniMax M2.7 / Hybrid	-4.880	+4.505	-4.763	+4.786	0
Qwen3.5 122B / Direct	-2.933	+3.193	-7.505	+7.314	1
Qwen3.5 122B / Thinking	-3.158	+3.084	-2.973	+2.958	0
Qwen3.5 27B / Direct	-4.522	+3.871	-6.150	+5.873	0
Qwen3.5 27B / Thinking	-3.214	+3.126	-3.196	+3.218	0

Table 12: Linear slopes of recall over B-count after aggregation over A-count and information levels. Reversals count adjacent B-count steps that locally move against the corresponding ordered direction.

Task / request	Frozen condition and candidates	Oracle	Cross-run output and observable error
T1 / `request_e395af793d68dadc`	A1 belief; positive branches doctrine, opinion, and religion; no exclusions; candidate meliorism, glossed as "the belief that the world can be made better by human effort"	False	11/11 True; acceptance within the outer scope without satisfying a required positive branch
T2 / `request_6a2c90c44fda8c89`	A1 conductor; no exclusions; candidates frankincense, lignite, silver, and semiconductor, with their glosses provided	[3,4]	11/11 [3]; incomplete-set omission
T3 / `request_fad438ed68a27885`	A1 improvement; B1 cleaning; candidates scrub, shave, sweeping, and electrolysis, with their glosses provided	False	10/11 True and 1/11 False; candidate-pool satisfiability false positive

Table 13: Observable errors in three frozen I2 requests. Conditions and candidates come from the stored requests; oracle labels and cross-run outputs come from the frozen evaluation records.