SEMBENCH-HARD: EVALUATING QUERY TRANSLATION AND OPERATOR GROUNDING IN LLM-BACKED SEMANTIC QUERY PROCESSING

Anonymous authors
Paper under double-blind review

ABSTRACT

Structured LLM workflows must turn a request into steps, and each intermediate step must return not only a decision but a grounded artifact that later steps can check and consume. Semantic query-processing engines (SQPEs) expose this need: they answer analytics questions over text-rich records with LLM-backed FILTER, JOIN, MAP, and RANK operators. An NL-facing SQPE can fail in two places: when it compiles a question into an operator program, and when an operator returns a decision without the grounding, such as evidence sentences or attributes, that its output contract requires. SemBench-Hard makes both harder and evaluates them separately. Its translation axis withholds the step-by-step plan from the question and scores generated programs by their final output; its execution axis runs reference programs through three system adapters with three backbone LLMs on paired Base and Hard operator tasks over four public datasets, where Hard adds distractors that make the right grounding harder to find, explicit constraints, output schemas, and deterministic grounding checks. On the translation axis, many accepted programs differ from the reference, so exact-match program scoring is too strict, and goal-only queries lead two of four translators to treat MAP's string output as a structured object. On the execution axis, Hard tasks lower strict (reference-exact) acceptance from 86.0% to 60.1% (-25.9 points); crediting grounding-only rejections at the rate human annotators judge broadly valid leaves -16.1, and accepting every grounding-only rejection leaves -6.3. The main finding is that operators mostly keep the decision but often return grounding that fails the contract's check; annotators judge about a third of these rejected outputs insufficiently grounded or wrong, three times the rate among accepted outputs. The loss persists: chains (LOTUS $\times$ Qwen3) stay 25–31 points lower under Hard at every length, and bounded task-local tools add 1.0 point at 2.24 times the API cost (4.0 with a SWE-bench helper). SQPEs should therefore schema-check generated programs and check each operator's grounding.

1 INTRODUCTION

Many practical analytics questions need more than a final label. A scientist needs the sentence that makes a claim's support checkable, a developer the patch and tests that resolve an issue, a contract analyst the clause that decides whether a policy applies, and a product analyst the attributes that separate an item from its variants and bundles. These tasks have relational structure (records, joins, candidate sets, rankings), but their key predicates are semantic and must be interpreted from text-rich records: they sit between relational queries and free-form generation.

Semantic query-processing engines (SQPEs) (Jo & Trummer, 2024; Liu et al., 2025a; Patel et al., 2025; Shankar et al., 2025) target this middle ground by extending relational query processing with LLM-backed operators such as FILTER, JOIN, MAP, and RANK. The engine keeps the dataflow inspectable and confines LLM calls to operator predicates and transformations instead of one table-level prompt for the whole query. This modularity is the appeal of SQPEs, and it creates two places where an LLM can fail, both at the output contract between operators: a program must bind to it, and an operator must fill it. An NL-facing SQPE must first translate the user's question into an operator program over concrete relations (Liu et al., 2025b; Dong et al., 2026). Each LLM-backed

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.

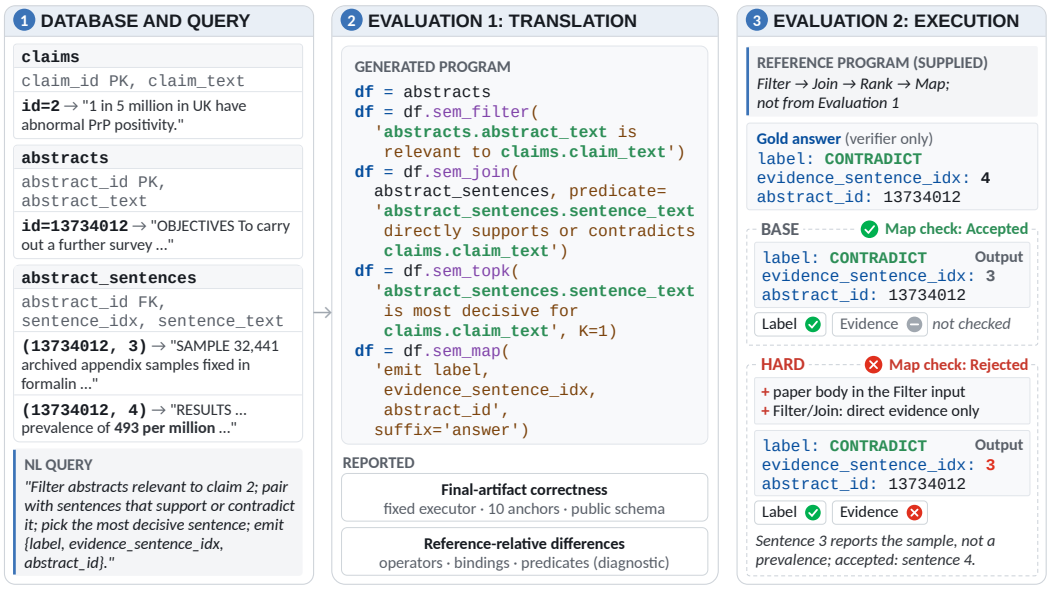


Figure 1: SemBench-Hard on a SciFact example. Evaluation 1 (translation) runs a program generated from the query and the public schema on a fixed executor and checks its final output. Evaluation 2 (execution) runs the reference program under Base and Hard; under Hard, FILTER and JOIN run Hard tasks and the terminal MAP output must also cite the annotated sentence.

operator must then return the output its program asks for, such as a label with its evidence, because the next operator or the analyst consumes that output and cannot check a decision whose support is missing. Can an LLM-backed operator return not only the right decision, but the grounded artifact that downstream computation needs? This is the operator-level analogue of FEVER, where requiring evidence cut a baseline's accuracy from 50.91% to 31.87% (Thorne et al., 2018).

Existing evaluations hold both failure points fixed. SemBench (Lao et al., 2026) evaluates how systems execute supplied semantic queries, on quality and cost; it does not measure whether a model can compile a question into the intended operator dataflow, and it does not vary what an operator must return for the same input. Text-to-SQL benchmarks such as Spider (Yu et al., 2018) and BIRD (Li et al., 2023) evaluate translation, but their programs run on deterministic databases, where a correct program returns a correct result; semantic programs pass free-text predicates and untyped LLM outputs between operators, and even a correct program can return an unusable output.

SemBench-Hard hardens both sides with two evaluation axes over four public datasets turned into relational views (Figure 1). The translation axis gives translators step-by-step queries or goal-only ones that withhold the plan, fixes the executor and the verifier, and scores only a program's final output. The execution axis fixes the program and hardens the operator task: three system adapters run the reference program on paired Base and Hard tasks with the same source item and operator family. A Hard task adds distractors, explicit constraints, an output schema, and deterministic checks of *grounding*, the evidence or attributes cited for a decision; *acceptance* is the share of outputs the verifier accepts, and the *gap* is Hard minus Base acceptance on matched tasks.

This separation localizes failures that one end-to-end score would conflate, and chains and bounded tools test whether they persist and whether local help repairs them. The Hard condition asks more than difficulty: it asks whether an operator still returns a usable output when its decision must come with a checkable grounding artifact. Distractors make the right grounding harder to find; the checks make its absence visible. Grounding checks score agreement with a reference, so we report strict, grounding-relaxed, and human-calibrated (sufficiency) acceptance.

We report four findings:

1. **Exact-match program scoring is too strict, and goal-only queries expose schema-binding failures.** About 40% of the generated programs accepted on a non-empty target differ from their

reference; where the reference program is accepted, they pass on 26.9–56.7% (22.9–62.8% on 70 anchors in an extension). Goal-only queries lower Correct by 4.9 points (6.8 with a second executor) and lead two translators to read MAP's string output as an object (`answer.label`).

2. **Hard tasks lower acceptance by 16.1 (calibrated) to 25.9 points (strict).** Strict acceptance falls from 86.0% to 60.1% (-25.9 points $[-26.4, -25.3]$). Under strict scoring, every scenario, operator family, system adapter, and backbone loses at least 12 points, as do a fifth scenario, HotpotQA (-17.2), and a fourth backbone, Claude Opus 5.5 (-21.5 to -24.2).
3. **Most Hard rejections keep the decision but fail the grounding check.** Accepting outputs whose only failed check is grounding leaves -6.3. Three annotators ($\kappa = 0.94$) judge 32% of grounding-only rejections invalid, against 10% of accepted outputs. Crediting grounding-only rejections at broadly-valid rates leaves -16.1.
4. **The gap persists along chains and with bounded tools.** On LOTUS $\times$ Qwen3, the share of anchors whose every step of a one- to four-operator prefix is accepted is 25–31 points lower under Hard, averaged over scenarios. Outside the nine SWE-bench MAP cells, where a repair-specification helper proposes the edited files, up to three task-local tool calls raise Hard acceptance by only 1.0 point at 2.24 times the API cost; including them, it is 4.0 over 115 cells.

2 SEMBENCH-HARD

Three design choices keep the comparison controlled. All scenarios share the four operator families, so differences across domains come from records, constraints, and verifiers rather than from a new operator language. Multi-operator execution tasks are prefixes of a canonical four-step chain, which varies composition depth without changing what a scenario asks. Hardening is paired by anchor and operator family, so the gap compares matched tasks rather than unrelated easy and hard examples.

Terms. A *reference program* is our program for a query or task. A *translator* LLM writes a *generated program*, a fixed *executor* runs it on each *anchor* (one issue, claim, hypothesis, or product offer with its candidate rows), and *Correct* is the share of executions whose final output matches a non-empty *target*, the records the query should return. *System adapters* run reference programs with *backbone* LLMs on paired *Base* and *Hard tasks* (same anchor and operator family). A task's *contract* states what its output must contain, and a deterministic *verifier* checks the output against the dataset's labels and annotations or against sets derived from the records by fixed rules. *Grounding* is the evidence, attribute set, decision basis, or test names that justify the *decision* (label, match, or top-1 choice). Gaps average over system–backbone–scenario–operator *cells*.

2.1 SCENARIOS

The four scenarios instantiate the same SQPE abstraction under different evidence and verification regimes: the source genre, distractor type, output artifact, and verifier change while the operator vocabulary stays fixed. Each scenario is named after its public dataset and materializes a relational view from it (Table 1); a task is anchored by one source item, and candidate rows link to that anchor through the schema. Construction adds only candidate expansion, hard negatives, constraints, output schemas, and verifier rules around the datasets' own records. The scenarios differ in the grounding they require: decision bases and test names for SWE-bench Verified (Jimenez et al., 2024), evidence sentences for SciFact, clause spans for ContractNLI, and attribute sets among near-duplicate offers for WDC Products. Appendices B.1–B.3 give schemas, pools, and construction and grounding rules.

2.2 TRANSLATION AXIS

Each query template pairs a goal with a reference program in a shared LOTUS-style operator language and is rendered as two queries: a step-by-step query that names one step per reference operator and a goal-only query that states only the goal; both require the same output fields. Step-by-step queries test whether a translator binds each step to the right relations and fields; goal-only queries, which state the question as a user would, also require planning the dataflow. A translator writes one program per query from the public schema and toy syntax examples; a fixed executor runs every generated and reference program on the same anchors, and a verifier frozen before execution compares only the final output with the query's target, so a program that differs from the reference is credited when

Table 1: Operator tasks per scenario, numbered in chain order; bold terms are anchors.

Domain	Scenario	#	Operator	Task formulation
Software remediation	SWE-bench Verified (SWE-bench Team, 2024)	1	FILTER	**issue**; candidate files → relevant files
		2	JOIN	**issue**; relevant files × candidate diffs → resolving file–diff pairs
		3	RANK	**issue**; resolving pairs → top-K fixes
		4	MAP	**issue**; top-K fixes → repair-specification JSON
Scientific claim–evidence reasoning	SciFact (Wadden et al., 2020)	1	FILTER	**claim**; candidate abstracts → topic-relevant abstracts
		2	JOIN	**claim**; abstracts × sentences → evidence sentence pairs
		3	RANK	**claim**; evidence sentence pairs → top-K evidence
		4	MAP	**claim**; top-K evidence → label/evidence JSON
Contract and policy reasoning	ContractNLI (Koreeda & Manning, 2021)	1	FILTER	**hypothesis**; contract clauses → policy-relevant clauses
		2	JOIN	**hypothesis**; relevant clauses → evidence clause pairs
		3	RANK	**hypothesis**; evidence clause pairs → top-K clauses
		4	MAP	**hypothesis**; top-K clauses → label/span JSON
Product entity matching	WDC Products (Peeters et al., 2024)	1	FILTER	**anchor offer**; candidate offers → eligible offers
		2	JOIN	**anchor offer**; eligible offers → candidate match pairs
		3	RANK	**anchor offer**; candidate match pairs → top-K matches
		4	MAP	**anchor offer**; top-K matches → same-product JSON

its output is correct. We report Correct and the reference-conditional rate, which restricts Correct to executions where the reference program is accepted, so that executor losses on the intended dataflow do not count against the translator. A multi-operator program is accepted only if every LLM-backed step is right on the anchor, so we report two executors. Program errors arise before any operator runs, independent of the executor, and the goal-only drop in Correct holds under both executors (Observation 2); other programs are typed by how they differ (Section 2.4).

2.3 PAIRED HARDENING

A Hard task keeps the anchor, operator family, and goal of its Base task and changes four components: added context (distractors and harder negatives, chosen by lexical or structural rules and never by a model), explicit constraints, grounding fields in a structured output schema, and stricter checks. An operator family names the decision a task asks for, not a particular native operator call. Explicit constraints and the output schema specify what a correct output is, while added context and stricter checks make it harder to produce; the gap measures their joint effect, and Figure 3 separates them. Deterministic Python verifiers check each operator's natural output: labels or selected sets for FILTER, matched pairs for JOIN, structured records for MAP, and the top-ranked choice for RANK, plus grounding where the contract lists it (Appendix B.3). The ContractNLI Hard JOIN contract is a label. A *grounding-only* rejection fails grounding but not the decision. *Strict* acceptance uses these verifiers, whose grounding checks score agreement with the annotated evidence or the rule-derived set; *grounding-relaxed* acceptance and human calibration also credit grounding-only rejections (Observations 5 and 6).

Scenario-specific rules. SWE-bench Verified differs from the other scenarios in three places. Its MAP output is a repair specification (edited files, symbols, operations, and tests) derived from the gold patch, outside the public schema, so the translation set uses the other three scenarios. Its Hard MAP context replaces the Base code snippets, oracle file hints, and full failing-test list with one code block, a reduced gold snippet, and one true and one distractor test (Appendix G.6). In Hard+tool runs (Section 2.6), its RANK operator gets eight calls instead of three, and its MAP tools include a deterministic repair-specification helper that proposes the edited files.

2.4 DIAGNOSTIC TAXONOMY

Table 2 lists the diagnostic types, adapted from the categories of NL2SQL-BUGs (Liu et al., 2025c). On the translation axis, reference-relative types locate where a failed generated program departs from

Table 2: Translation and execution diagnostics. Translation types mark differences from one reference program, for generated programs never accepted on an anchor where the reference program is accepted; execution types mark failed verifier checks. Types can overlap.

Translation: differences from reference		**Execution: failed checks**	
Type	*Definition*	*Check*	*Definition*
A1 Missing operator	A reference step has no counterpart.	Format	Output unparsable or fields missing.
A2 Added operator	An extra step appears.	Completeness	Set or ranking incomplete or mis-sized.
A3 Different operator	Aligned steps differ in operator family.	Grounding	Evidence, attributes, basis, or tests rejected.
A4 Different order	Same operators in a different order.	Decision	Label, match, or top-1 choice rejected.
B1 Relation difference	Starting or joined relation differs.	Other fields	Other required fields rejected.
C1 Predicate difference	An LLM judge finds a predicate non-equivalent.		
D1 Parameter difference	Top-K or output fields differ.		

the reference; they explain failures and do not affect Correct. On the execution axis, a rejection is labeled by every verifier check it fails, so failures can be split into grounding and decision failures.

2.5 Multi-Operator Chaining

Chains test whether semantic-operator failures persist when the dataflow becomes longer. A chain runs a scenario's four operators in the order of Table 1. A prefix of length k runs the first k steps, each consuming the anchor and upstream fields, and all-step acceptance requires every scored step to be accepted. A step is accepted only if each of its per-row judgments is, so a one-step prefix is stricter than the single-operator task. A Hard chain runs Hard FILTER and JOIN tasks with Base RANK and MAP instructions; under Hard, the SciFact and SWE-bench Verified MAP steps are scored by the Hard MAP verifier. The chain RANK step inherits its upstream JOIN verdict, so prefixes of two and three operators coincide (Appendix B.6).

2.6 Recovery Diagnostics

Tool runs test whether the gap can be reduced without weakening the verifier: do failures come from interface friction that local lookup repairs, or from the grounding and decisions the operator must produce? In *Hard+tool* runs, each operator of a Hard task may make up to three calls to task-local deterministic tools, such as lexical lookup, candidate retrieval, inspection of shown artifacts, and schema validation. A shared tool runtime adds operator-specific tool guidance to the Hard prompt; the tools see only the task payload, never hidden tests or verifier outputs, and the unchanged Hard verifier scores the output (Appendix B.9).

3 Experimental Setup

Translation. The translation set covers SciFact, ContractNLI, and WDC Products with query templates of one to five operators, each phrased step by step and goal-only. The translators are GPT-5.4 (OpenAI, 2026a), GPT-5.4-mini (OpenAI, 2026b), Claude Sonnet 4.6 (Anthropic, 2026c), and Qwen3-30B-A3B FP8 (Yang et al., 2025) (Qwen3 below), with the prompts of Appendix C. The fixed executor is Qwen3 served locally at temperature 0 with native LOTUS operators, on ten anchors per scenario (70 in an extension; Appendix C); GPT-5.4-mini is the second executor. Targets are frozen from the dataset annotations under each output contract, independent of any program, and about two thirds are non-empty. Intervals resample templates and anchors. The predicate judge is Claude Opus 4.7 (Anthropic, 2026a), cross-checked by Claude Haiku 4.5 (Anthropic, 2025).

Table 3: Generated-program outcomes by translator and query style under the Qwen3 executor and, in the last two columns, the GPT-5.4-mini executor. (a) Shares of executions on non-empty targets (%) under each executor; Ref.-cond.: Correct where the reference program is accepted; Output err.: wrong output form; Prog. err.: the program fails before any operator runs; under Qwen3, verifier rejections make up the rest; brackets: 95% CI. (b) Types of Table 2 (%, overlapping) among the n_0 programs never accepted where the reference is; C1 over judged programs; D1 does not occur.

		(a) Qwen3 executor				**GPT-5.4-mini exec.**	
Translator	**Style**	Correct ↑	Ref.-cond. ↑	Output err.	Prog. err.	Correct	Ref.-cond.
GPT-5.4	Step	14.1 [5, 26]	56.7 [24, 83]	31	0	26.3	55.0
	Goal	12.8 [3, 24]	55.2 [21, 83]	49	0	22.3	46.4
GPT-5.4-mini	Step	14.1 [6, 25]	55.2 [24, 81]	32	0	21.3	45.0
	Goal	9.8 [2, 20]	44.8 [15, 74]	28	0	13.0	27.1
Claude Sonnet 4.6	Step	11.1 [3, 21]	37.3 [10, 68]	36	0	15.3	32.9
	Goal	5.9 [0, 14]	26.9 [0, 57]	30	34	5.7	11.4
Qwen3	Step	16.7 [7, 28]	55.2 [32, 76]	35	0	13.0	23.6
	Goal	7.9 [1, 16]	34.3 [5, 66]	35	38	7.7	12.1
Reference program	–	22.0	–	12	0	46.7	–

(b) Query style	n_0	A1 missing	A2 added	A3 different	A4 order	B1 relation	C1 predicate
Step-by-step	22	9	14	5	0	27	82
Goal-only	53	30	43	36	6	30	88

Execution. LOTUS (Patel et al., 2025), Palimpzest (Liu et al., 2025a), and DocETL (Shankar et al., 2025) run each task with GPT-5.4-mini, Claude Sonnet 4.6, or Qwen3 through adapters, mostly as one MAP-style LLM call on a row that holds the task's candidates. The execution axis thus fixes the program and compares prompt rendering and call paths (Palimpzest's optimizer with a single model, LOTUS without cascades, DocETL without rewrites), not plan optimizers; every operator call goes to the evaluated backbone, and Observation 5 compares native LOTUS operators (Appendices B.7 and D). Eight Hard JOIN cells use label-only prompts (native WDC Products joins in LOTUS and Palimpzest, and two LOTUS SciFact cells); prompting the affected Qwen3 and Claude Sonnet 4.6 joins for the grounding field moves their gap from -33.6 to -25.1 and the primary gap to -25.5 (Appendix B.7). Scores are unweighted means of per-cell acceptance over the 140 cells with Base and Hard records, with 95% record-bootstrap intervals (Appendix B.8). Chains run on LOTUS $\times$ Qwen3 with 100 anchors per scenario; the ablation and most checks use 16 Qwen3 $\times$ Palimpzest or LOTUS cells (Appendix G).

4 RESULTS

4.1 TRANSLATION

Observation 1 (accepted generated programs often differ from the reference). Of the generated programs accepted on at least one non-empty target, 42% use a different operator sequence from their reference (41% on 70 anchors). A translation query is scored on the final output of a whole multi-operator program, whereas an execution task scores one operator, so the executor accepts the reference program on 22.0% of executions with a non-empty target (22.7% on 70 anchors; Table 3): reference acceptance falls from 39.3% on one- and two-operator templates to 7.3% on three to five (61.4% and 33.8% under GPT-5.4-mini), and a repeat run agrees on 93.8% of reference verdicts. Where the reference is accepted, generated programs are accepted on 26.9–56.7%, on 70 anchors 22.9–62.8% (Appendix C). Equivalent rewrites of reference programs change the executor's verdict no more often than a repeat run (agreement 91.9% vs. 90.9% on the rewritten programs).

Observation 2 (goal-only queries cause program errors for two translators). Goal-only queries lower Correct by 4.9 points (95% CI $[1.5, 8.8]$) and the reference-conditional rate by 10.8 points ($[-0.9, 22.6]$); under the second executor the drops are 6.8 ($[2.1, 12.4]$) and 14.8 points ($[6.1, 24.6]$), and on 70 anchors 5.4 ($[2.4, 8.6]$) and 13.6 ($[2.8, 24.0]$). For Claude Sonnet 4.6 and Qwen3, 34–38% of goal-only executions fail with program errors, against none of the step-by-step executions and

Table 4: Reference-program acceptance on Base and Hard tasks (%), unweighted means over the cells of each row. Strict acceptance uses the verifiers as built; grounding-relaxed acceptance also accepts outputs whose only failed check is grounding. Hard−Base in points; brackets: 95% CI.

		Strict acceptance			Grounding-relaxed acceptance		
	Cells	Base	Hard	Hard−Base	Base	Hard	Hard−Base
By scenario							
SWE-bench Verified	36	95.4	68.1	−27.3 [−28.2, −26.3]	95.4	74.9	−20.5 [−21.5, −19.6]
SciFact	32	89.9	76.5	−13.5 [−14.1, −12.7]	89.9	88.8	−1.1 [−1.7, −0.5]
ContractNLI	36	76.8	63.4	−13.4 [−14.5, −12.3]	81.8	76.2	−5.7 [−6.7, −4.7]
WDC Products	36	82.2	34.2	−48.0 [−49.4, −46.5]	82.2	84.7	+2.5 [+1.4, +3.6]
By operator family							
FILTER	35	84.7	60.6	−24.2 [−25.1, −23.2]	84.7	82.8	−1.9 [−2.7, −1.1]
JOIN	35	84.8	57.6	−27.2 [−28.2, −26.2]	84.8	75.7	−9.1 [−10.0, −8.2]
MAP	35	78.0	43.0	−35.0 [−36.2, −33.7]	83.2	75.9	−7.3 [−8.4, −6.1]
RANK	35	96.4	79.2	−17.2 [−18.3, −16.0]	96.4	89.3	−7.0 [−7.9, −6.2]
All cells	140	86.0	60.1	−25.9 [−26.4, −25.3]	87.3	80.9	−6.3 [−6.8, −5.9]

none of the GPT programs; most treat the MAP output as a structured object and read a field such as `answer.label`, although the system message defines it, as native LOTUS does, as the single string column `answer`. The rest comes from a second binding failure: goal-only programs more often write the answer to LOTUS's default column (Appendix C).

Observation 3 (goal-only queries change program structure). Among programs never accepted where the reference is (Table 3b), goal-only programs more often omit an operator (A1: 30% against 9%), add one (A2: 43% against 14%), or switch family (A3: 36% against 5%).

4.2 EXECUTION

Observation 4 (Hard tasks lower acceptance in every scenario, operator family, system adapter, and backbone). Hard tasks lower strict (reference-exact) acceptance from 86.0% to 60.1%, a gap of -25.9 points (95% CI $[-26.4, -25.3]$; Table 4), a reference-exact stress-test score; crediting grounding-only rejections at human-judged rates leaves -16.1, and accepting all of them -6.3 (Observation 6). Under strict scoring, every scenario, operator family, system adapter, and backbone loses at least 12 points; MAP loses the most (-35.0) and RANK the least (-17.2); the three backbones lose 25.4–26.4 points despite an 8.8-point spread in Base acceptance, and the three system adapters 25.2–26.5 (Appendix E). The gap is largest where grounding is a constructed artifact (WDC Products attribute sets, -48.0; SWE-bench repair specifications, -27.3); grounding-relaxed, it is at most 5.7 points outside SWE-bench.

Observation 5 (the gap holds under checks of scoring, inputs, and implementation). In Figure 2, rescoring the primary cells (top rows) shrinks the gap as the grounding checks loosen, from -25.9 (strict) to -19.1 (tolerant, which accepts near matches of the annotated sets) and -6.3 (grounding-relaxed), but never closes it (Appendix G.1). WDC Products attribute sets follow a fixed rule over {brand, title, price}; a subset score, accepting any non-empty subset of the rule's set as the prompts' minimal decisive attributes, gives MAP and RANK gaps of -42.8 and -38.4 (strict -51.4 and -43.9), and without WDC Products the gap is -18.2 (Appendix B.3). The remaining rows are separate runs, each compared with its own reference on the same cells: aliasing public identifiers changes the gap by -1.4 points ($[-4.9, +2.1]$), three permutations of the prompt's row fields stay within 0.7 points, native LOTUS operators, pooled over all 16 scenario–operator pairs, keep at least the adapter's gap (-31.1 against -27.0; -36.6 against -35.7 on the ten grounding-checked cells), and Claude Opus 5.5 keeps -21.5 against -28.7 for Claude Sonnet 4.6 (Appendices G.3 and G.5).

Observation 6 (grounding checks account for most of the acceptance gap). Grounding-relaxed acceptance leaves a gap of -6.3 points (Table 4), and in the ablation reverting the stricter checks raises Hard acceptance more than reverting any other component (Figure 3). Three annotators blind to the verifier outcome judged sampled outputs for correctness, grounding, and completeness (Fleiss

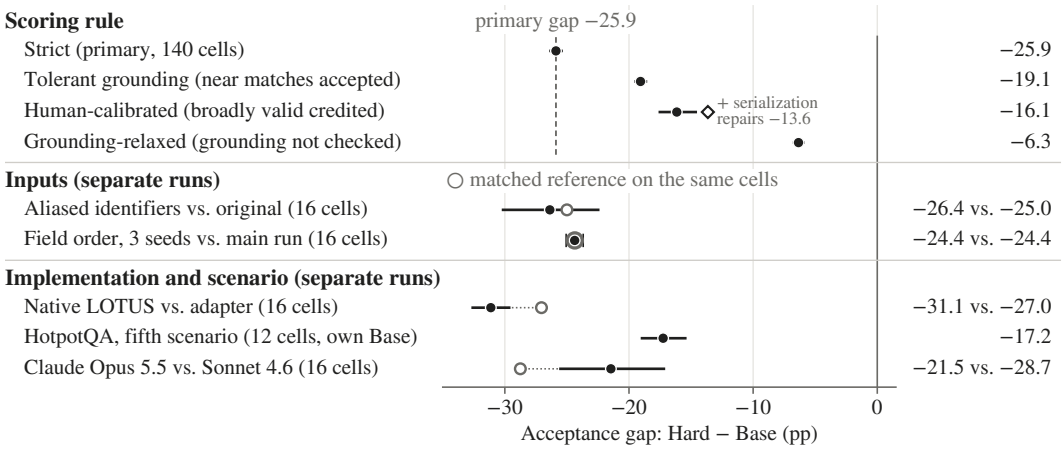


Figure 2: Does the gap survive changes to scoring, inputs, and implementation? Acceptance gap (Hard−Base, pp; bars: 95% CI, or min–max over three seeds). Rescored rows: the 140 primary cells against the primary gap (dashed); separate runs: filled point against the matched reference on the same cells (open circle; values: filled vs. open). Diamond: also counting serialization repairs.

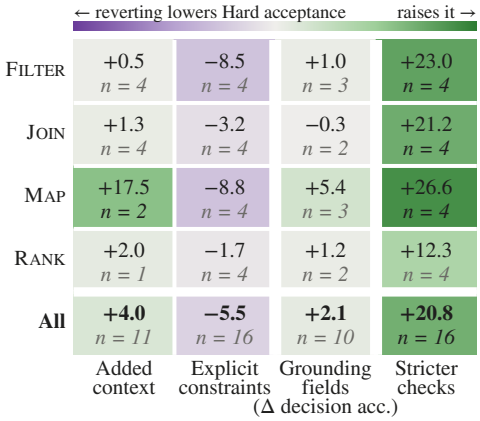
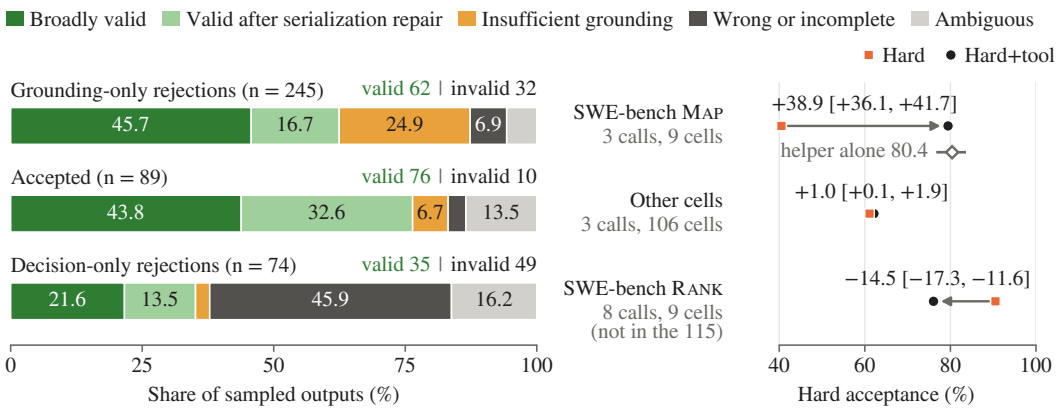


Figure 3: Leave-one-out ablation: mean change in Hard acceptance (points) after reverting one Hard component, over the n cells where it applies; the separated column reports the change in decision accuracy. Constraints specify the task (Section 2.3), so reverting them can lower acceptance.

$\kappa = 0.94$); the majority of three external annotators agrees with their majority on the six-way category in 37 of 40 re-labeled cases (Appendix F). In-group annotators judge 46% of grounding-only rejections broadly valid and 32% invalid, against 10% invalid among accepted outputs (Figure 4a). Crediting grounding-only Hard rejections at each scenario's broadly-valid rate leaves a gap of -16.1 ($[-17.6, -14.5]$; -13.6 with serialization repairs), and also debiting accepted Hard outputs judged invalid gives -21.3 ($[-24.6, -18.3]$). Under both, every operator family, adapter, and backbone loses at least 12 points and every scenario at least 6 (Appendix F).

Observation 7 (operators fail on different checks). Of 34,665 Hard failures, 55.7% fail only grounding, 18.3% only the decision, and 22.7% both (Appendix H); MAP failures are mostly grounding-only (65–71%), RANK failures most often decision-only (36–41%).

Observation 8 (the gap persists along chains). Averaged over scenarios, all-step acceptance falls from 57.5% at one operator to 42.0% at four under Base and from 27.0% to 15.8% under Hard, 25–31 points lower at every length (Figure 5). Hardening one step at a time lowers it by up to 39 points; each clear loss comes from a Hard FILTER or JOIN step or SciFact's Hard MAP rule (Appendix G.6).



(a) Human assessment: are rejected outputs wrong?

(b) Effect of bounded tools

Figure 4: (a) Majority category that annotators assign to sampled outputs, by verifier outcome (% of sampled outputs; brackets: valid and invalid shares). (b) Hard and Hard+tool acceptance (%), labeled with the change in points [95% CI]; diamond: the repair-specification helper alone.

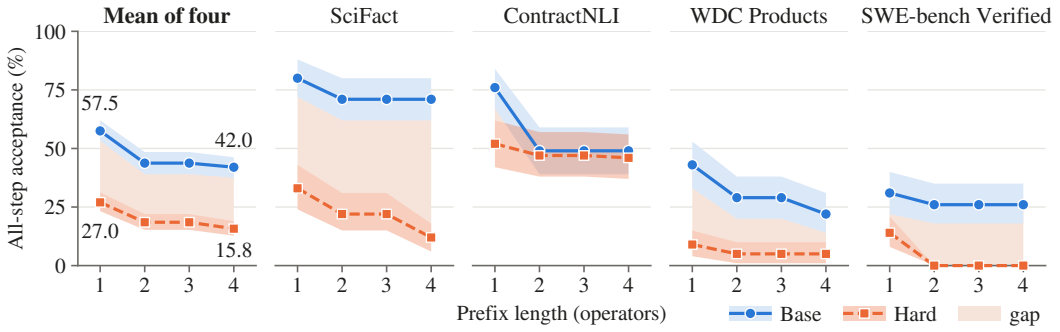


Figure 5: All-step acceptance of chain prefixes (%); shading: the Base–Hard gap; bands: 95% CI.

4.3 Recovery and Generality

Observation 9 (bounded tools do not close the gap). Outside the nine SWE-bench MAP cells, where a repair-specification helper proposes the edited files, up to three task-local tool calls raise Hard acceptance by only 1.0 point ($[0.1, 1.9]$) at 2.24 times the API cost (Figure 4b). With the helper cells (helper alone: 80.4%), the gain over 115 cells is 4.0 ($[3.1, 4.9]$; Appendix I).

Observation 10 (the gap holds on a fifth scenario, other operator implementations, and a frontier backbone). HotpotQA (Yang et al., 2018), a fifth scenario with its own distractors, gives -17.2 ($[-19.1, -15.4]$) strict and -2.7 grounding-relaxed. On a shared FILTER contract, Nirvana's operators (Zhu et al., 2026) and cost-saving approximations inspired by BARGAIN (Zeighami et al., 2025) and CSV (Hou et al., 2026) lose 18.5–29.0 strict, and Claude Opus 5.5 (Anthropic, 2026b) loses 21.5–24.2 strict and at most 2.1 relaxed (Appendix G.5).

5 Discussion and Conclusion

Decision-only evaluation can overstate the reliability of LLM-backed components: outputs that fail only grounding account for most of the strict 25.9-point gap (grounding-relaxed -6.3), and human calibration still leaves -16.1. Generated programs should be scored by final output and schema-checked. The lesson likely extends to other workflows in which an LLM output feeds computation: grounded intermediate artifacts are first-class outputs, usable only when the next step can consume the decision and its support, which calls for typed grounding schemas and verifier-aware operators.

AI USE STATEMENT

In this work, we used generative AI tools to implement methods (benchmark construction, system adapters, and analysis code), to clean and reformat the source datasets into relational views, to draft the translation protocol and the natural-language query templates, to give feedback on experimental design (simulated reviews that proposed robustness checks), to support qualitative data analysis (an LLM judge assigns one translation-failure category, C1), and to help interpret results. We have not used generative AI tools to generate synthetic data sets (Hard tasks are built by deterministic rules) or to propose or refine hypotheses, and theoretical models, mathematical claims, proofs, and translation are not applicable to this work. Additionally, we used generative AI tools to make figures, draft and edit the text, and search for and format references. We have reviewed all AI-assisted work: the authors reviewed and tested the code, every reported number is recomputed from run records by the released scripts, and every reference was checked against its primary source. Every LLM whose outputs enter a reported number is listed in Table 20. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

SemBench-Hard is built from five public datasets and adds no personal data beyond what they contain. The three in-group annotators of Appendix F are members of the authors' research group who are not authors. They participated voluntarily and reported using no AI tools, and no personal information about them was collected. The three external annotators of the check in Appendix F are not members of the authors' research group; they participated without compensation, reported using no AI tools, and no personal information about them was collected. SemBench-Hard is a stress test, and its scores do not certify readiness for unsupervised use in legal, scientific, medical, or other high-stakes settings.

REPRODUCIBILITY STATEMENT

Appendices B–D specify task construction, verifiers, the translation protocol and prompts, models, and the aggregation rule. The anonymized repository at `https://anonymous.4open.science/r/SEMBENCH_HARD/` contains the code, prompts, frozen inputs, and the scripts that run every experiment and compute every table from its run records; the run records and the annotators' labels are released with the de-anonymized version. Appendix K lists the contents and the source snapshots needed to run the models.

REFERENCES

Anthropic. Introducing Claude Haiku 4.5. `https://www.anthropic.com/news/claude-haiku-4-5`, 2025. Anthropic product announcement, accessed 2026-04-30.

Anthropic. Introducing Claude Opus 4.7. `https://www.anthropic.com/news/claude-opus-4-7`, 2026a. Anthropic product announcement, accessed 2026-09-24.

Anthropic. Introducing Claude Opus 5.5. `https://www.anthropic.com/claude-opus-5-5`, 2026b. Anthropic product page, accessed 2026-09-25.

Anthropic. Introducing Claude Sonnet 4.6. `https://www.anthropic.com/news/claude-sonnet-4-6`, 2026c. Anthropic product announcement, accessed 2026-04-30.

Asim Biswal, Liana Patel, Siddharth Jha, Amog Kamsetty, Shu Liu, Joseph E. Gonzalez, Carlos Guestrin, and Matei Zaharia. Text2SQL is not enough: Unifying AI and databases with TAG. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, 2025.

Zhoujun Cheng, Tianbao Xie, Peng Shi, Chengzu Li, Rahul Nadkarni, Yushi Hu, Caiming Xiong, Dragomir Radev, Mari Ostendorf, Luke Zettlemoyer, Noah A. Smith, and Tao Yu. Binding language models in symbolic languages. In *The Eleventh International Conference on Learning Representations, ICLR 2023*, 2023. URL `https://openreview.net/forum?id=lH1PV42cbF`.

Hanjun Dai, Bethany Y. Wang, Xingchen Wan, Bo Dai, Sherry Yang, Azade Nova, Pengcheng Yin, Phitchaya Mangpo Phothilimthana, Charles Sutton, and Dale Schuurmans. UQE: A query engine

for unstructured databases. In *Advances in Neural Information Processing Systems*, volume 37, pp. 29807–29838, 2024. doi: 10.52202/079017-0938.

Jay DeYoung, Sarthak Jain, Nazneen Fatema Rajani, Eric Lehman, Caiming Xiong, Richard Socher, and Byron C. Wallace. ERASER: A benchmark to evaluate rationalized NLP models. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4443–4458. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.408.

Wenkai Dong, Ruyu Li, Sairam Gurajada, and Yifan Wang. Bridge the last-mile gap to semantic analytics: Compiling natural-language queries into semantic operator pipelines. *CoRR*, abs/2606.04641, 2026.

Alex Egg, Martin Iglesias Goyanes, Friso Kingma, Andreu Mora, Leandro von Werra, and Thomas Wolf. DABstep: Data agent benchmark for multi-step reasoning. *CoRR*, abs/2506.23719, 2025.

Tianyu Gao, Howard Yen, Jiatong Yu, and Danqi Chen. Enabling large language models to generate text with citations. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 6465–6488, 2023.

Matt Gardner, Yoav Artzi, Victoria Basmov, Jonathan Berant, Ben Bogin, Sihao Chen, Pradeep Dasigi, Dheeru Dua, Yanai Elazar, Ananth Gottumukkala, Nitish Gupta, Hannaneh Hajishirzi, Gabriel Ilharco, Daniel Khashabi, Kevin Lin, Jiangming Liu, Nelson F. Liu, Phoebe Mulcaire, Qiang Ning, Sameer Singh, Noah A. Smith, Sanjay Subramanian, Reut Tsarfaty, Eric Wallace, Ally Zhang, and Ben Zhou. Evaluating models' local decision boundaries via contrast sets. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 1307–1323. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.findings-emnlp.117.

Parker Glenn, Parag Dakle, Liang Wang, and Preethi Raghavan. BlendSQL: A scalable dialect for unifying hybrid question answering in relational algebra. In *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 453–466. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-acl.25.

Nan Hou, Kangfei Zhao, Jiadong Xie, and Jeffrey Xu Yu. Beyond linear LLM invocation: An efficient and effective semantic filter paradigm. *CoRR*, abs/2603.04799, 2026.

Robin Jia and Percy Liang. Adversarial examples for evaluating reading comprehension systems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pp. 2021–2031. Association for Computational Linguistics, 2017. doi: 10.18653/v1/D17-1215.

Carlos E Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik R Narasimhan. SWE-bench: Can language models resolve real-world github issues? In *The Twelfth International Conference on Learning Representations (ICLR)*, 2024.

Saehan Jo and Immanuel Trummer. ThalamusDB: Approximate query processing on multi-modal data. *Proceedings of the ACM on Management of Data*, 2(3):1–26, 2024.

Yuta Koreeda and Christopher D. Manning. ContractNLI: A dataset for document-level natural language inference for contracts. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pp. 1907–1919, 2021. doi: 10.18653/v1/2021.findings-emnlp.164.

Jiale Lao, Andreas Zimmerer, Olga Ovcharenko, Tianji Cong, Matthew Russo, Gerardo Vitagliano, Michael Cochez, Fatma Özcan, Gautam Gupta, Thibaud Hottelier, H. V. Jagadish, Kris Kissel, Sebastian Schelter, Andreas Kipf, and Immanuel Trummer. SemBench: A benchmark for semantic query processing engines. *Proceedings of the VLDB Endowment*, 19(8):1754–1767, 2026. doi: 10.14778/3811243.3811249.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin Chang, Fei Huang, Reynold Cheng, and Yongbin Li. Can LLM already serve as a database interface? A BIg bench for large-scale database grounded Text-to-SQLs. In *Advances in Neural Information Processing Systems*, volume 36, pp. 42330–42357, 2023.

Yiming Lin, Madelon Hulsebos, Ruiying Ma, Shreya Shankar, Sepanta Zeighami, Aditya G. Parameswaran, and Eugene Wu. Towards accurate and efficient document analytics with large language models. *CoRR*, abs/2405.04674, 2024.

Yin Lin, Tianjing Zeng, Zhongjun Ding, Rong Zhu, Bolin Ding, H. V. Jagadish, and Jingren Zhou. SEMA-SQL: Beyond traditional relational querying with large language models. *CoRR*, abs/2604.23477, 2026.

Chunwei Liu, Matthew Russo, Michael Cafarella, Lei Cao, Peter Baile Chen, Zui Chen, Michael Franklin, Tim Kraska, Samuel Madden, Rana Shahout, and Gerardo Vitagliano. Palimpzest: Optimizing AI-powered analytics with declarative query processing. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, 2025a.

Chunwei Liu, Gerardo Vitagliano, Brandon Rose, Matthew Printz, David Andrew Samson, and Michael Cafarella. PalimpChat: Declarative and interactive AI analytics. In *Companion of the 2025 International Conference on Management of Data, SIGMOD 2025*, pp. 183–186. ACM, 2025b. doi: 10.1145/3722212.3725122.

Shicheng Liu, Jialiang Xu, Wesley Tjangnaka, Sina Semnani, Chen Yu, and Monica Lam. SUQL: Conversational search over structured and unstructured data with large language models. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 4535–4555. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.findings-naacl.283.

Xinyu Liu, Shuyu Shen, Boyan Li, Nan Tang, and Yuyu Luo. NL2SQL-BUGs: A benchmark for detecting semantic errors in NL2SQL translation. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 5662–5673, 2025c. doi: 10.1145/3711896.3737427.

Ruiying Ma, Shreya Shankar, Ruiqi Chen, Yiming Lin, Sepanta Zeighami, Rajoshi Ghosh, Abhinav Gupta, Anushrut Gupta, Tanmai Gopal, and Aditya G. Parameswaran. Can AI agents answer your data questions? A benchmark for data agents. *CoRR*, abs/2603.20576, 2026.

Iman Mirzadeh, Keivan Alizadeh-Vahid, Hooman Shahrokhi, Oncel Tuzel, Samy Bengio, and Mehrdad Farajtabar. GSM-Symbolic: Understanding the limitations of mathematical reasoning in large language models. In *The Thirteenth International Conference on Learning Representations, ICLR 2025*, 2025. URL https://openreview.net/forum?id=AjXkRZIvjB.

OpenAI. Introducing GPT-5.4. https://openai.com/index/introducing-gpt-5-4/, 2026a. OpenAI product announcement, accessed 2026-04-30.

OpenAI. Introducing GPT-5.4 mini and nano. https://openai.com/index/introducing-gpt-5-4-mini-and-nano/, 2026b. OpenAI product announcement, accessed 2026-04-30.

Liana Patel, Siddharth Jha, Melissa Pan, Harshit Gupta, Parth Asawa, Carlos Guestrin, and Matei Zaharia. Semantic operators and their optimization: Enabling LLM-based data processing with accuracy guarantees in LOTUS. *Proceedings of the VLDB Endowment*, 18(11):4171–4184, 2025.

Ralph Peeters, Reng Chiz Der, and Christian Bizer. WDC Products: A multi-dimensional entity matching benchmark. In *Proceedings of the 27th International Conference on Extending Database Technology (EDBT)*, pp. 22–33. OpenProceedings.org, 2024. doi: 10.48786/EDBT.2024.03.

Hannah Rashkin, Vitaly Nikolaev, Matthew Lamm, Lora Aroyo, Michael Collins, Dipanjan Das, Slav Petrov, Gaurav Singh Tomar, Iulia Turc, and David Reitter. Measuring attribution in natural language generation models. *Computational Linguistics*, 49(4):777–840, 2023.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. Beyond accuracy: Behavioral testing of NLP models with CheckList. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 4902–4912. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.acl-main.442.

Matthew Russo, Chunwei Liu, Sivaprasad Sudhir, Gerardo Vitagliano, Michael Cafarella, Tim Kraska, and Samuel Madden. Abacus: A cost-based optimizer for semantic operator systems. *Proceedings of the VLDB Endowment*, 19(5):1060–1073, 2026. doi: 10.14778/3796195.3796215.

Mohammed Saeed, Nicola De Cao, and Paolo Papotti. Querying large language models with SQL. In *Proceedings of the 27th International Conference on Extending Database Technology (EDBT)*, 2024.

Shreya Shankar, Tristan Chambers, Tarak Shah, Aditya G. Parameswaran, and Eugene Wu. DocETL: Agentic query rewriting and evaluation for complex document processing. *Proceedings of the VLDB Endowment*, 18(9):3035–3048, 2025.

Yunxiang Su, Tianjing Zeng, Zhongjun Ding, Yin Lin, Rong Zhu, Zhewei Wei, Bolin Ding, and Jingren Zhou. Large language model-enhanced relational operators: Taxonomy, benchmark, and analysis. *CoRR*, abs/2603.02537, 2026.

SWE-bench Team. SWE-bench Verified. `https://www.swebench.com/verified.html`, 2024. Official benchmark website.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and VERification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pp. 809–819. Association for Computational Linguistics, 2018. doi: 10.18653/v1/N18-1074.

Matthias Urban and Carsten Binnig. CAESURA: Language models as multi-modal query planners. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, 2024.

David Wadden, Shanchuan Lin, Kyle Lo, Lucy Lu Wang, Madeleine van Zuylen, Arman Cohan, and Hannaneh Hajishirzi. Fact or fiction: Verifying scientific claims. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7534–7550, 2020. doi: 10.18653/v1/2020.emnlp-main.609.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9440–9450. Association for Computational Linguistics, 2024. doi: 10.18653/v1/2024.acl-long.511.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, et al. Qwen3 technical report. *CoRR*, abs/2505.09388, 2025.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2369–2380, 2018. doi: 10.18653/v1/D18-1259.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 3911–3921, 2018. doi: 10.18653/v1/D18-1425.

Sepanta Zeighami, Shreya Shankar, and Aditya G. Parameswaran. Cut costs, not accuracy: LLM-powered data processing with guarantees. *Proceedings of the ACM on Management of Data*, 3(6): 1–26, 2025. doi: 10.1145/3769776.

Fuheng Zhao, Divyakant Agrawal, and Amr El Abbadi. Hybrid querying over relational databases and large language models. In *Proceedings of the Conference on Innovative Data Systems Research (CIDR)*, 2025.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and Chatbot Arena. In *Advances in Neural Information Processing Systems*, volume 36, pp. 46595–46623, 2023.

Ruiqi Zhong, Tao Yu, and Dan Klein. Semantic evaluation for text-to-SQL with distilled test suites. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 396–411. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.29.

Junhao Zhu, Lu Chen, Xiangyu Ke, Ziquan Fang, Tianyi Li, Yunjun Gao, and Christian S. Jensen. Beyond relational: Semantic-aware multi-modal analytics with LLM-native query optimization. *Proceedings of the ACM on Management of Data*, 4(1):1–25, 2026. doi: 10.1145/3786628.

A Related Work

Semantic query processing engines and their benchmarks. LOTUS optimizes semantic operators with accuracy guarantees (Patel et al., 2025); Palimpzest (Liu et al., 2025a) and its cost-based optimizer Abacus (Russo et al., 2026) optimize declarative AI-powered analytics; DocETL rewrites document-processing pipelines with agents (Shankar et al., 2025); and ThalamusDB (Jo & Trummer, 2024), UQE (Dai et al., 2024), ZenDB (Lin et al., 2024), and Nirvana (Zhu et al., 2026) query multimodal data or documents. BARGAIN (Zeighami et al., 2025) reduces the cost of LLM-powered data processing with guarantees on result quality, and CSV (Hou et al., 2026) reduces the LLM invocations of semantic filters to sublinear complexity with error guarantees. SemBench (Lao et al., 2026) evaluates three academic systems and BigQuery across scenarios, modalities, and operators, and LROBench (Su et al., 2026) benchmarks LLM-enhanced relational operators under a unified taxonomy. In SemBench-Hard, each Base task has a Hard task on the same anchor and operator family, with distractors and an output contract, including grounding where the contract requires it, checked deterministically.

From natural language to hybrid relational–LLM programs. Binder (Cheng et al., 2023) binds language-model calls into SQL or Python programs; SUQL (Liu et al., 2024) extends SQL with free-text primitives; BlendSQL (Glenn et al., 2024) encodes a decomposed hybrid question-answering plan as one SQLite-superset query; and TAG (Biswal et al., 2025) frames questions over databases as table-augmented generation, with TAG-Bench. Galois (Saeed et al., 2024) queries LLMs with SQL, and HQDL (Zhao et al., 2025) studies hybrid querying over relational databases and LLMs. Closer to SQPE frontends, CAESURA (Urban & Binnig, 2024) translates natural-language queries into multimodal query plans, PalimpChat (Liu et al., 2025b) exposes Palimpzest through a chat agent, SEMA-SQL (Lin et al., 2026) generates hybrid relational-algebra queries with LLM user-defined functions, and NL2Pipe (Dong et al., 2026) compiles questions into semantic-operator pipelines. Most of these works evaluate their own end-to-end systems; CAESURA checks plan correctness, and PalimpChat is a demonstration. Our translation axis instead fixes the operator language and the executor, varies the translator and the query style, and scores the final output of each generated program with the verifier of its query. Data-agent benchmarks (Ma et al., 2026; Egg et al., 2025) evaluate multi-step agents over heterogeneous data, whereas our execution axis fixes the program.

NL-to-SQL evaluation and program equivalence. Spider (Yu et al., 2018) reports exact matching against reference SQL and BIRD (Li et al., 2023) reports execution accuracy; Zhong et al. (2020) showed that Spider's official metric rejects some semantically correct predictions and proposed test-suite accuracy over distilled databases. NL2SQL-BUGs (Liu et al., 2025c) provides a semantic-error taxonomy and detection benchmark, whose categories we adapt for reference-relative types. Unlike deterministic SQL, semantic-operator programs run through an LLM, so equivalent programs can return different outputs.

Evidence selection versus label prediction. FEVER (Thorne et al., 2018) reports 31.87% baseline accuracy when correct evidence is required and 50.91% when it is not; SciFact (Wadden et al., 2020) reports sentence-level rationale selection alongside abstract-level labels; ContractNLI (Koreeda & Manning, 2021) reports evidence-span identification separately from NLI accuracy; and ERASER (DeYoung et al., 2020) scores model rationales against human rationales and for faithfulness; attribution evaluation (Rashkin et al., 2023; Gao et al., 2023) judges whether cited text supports a generated statement, by humans or NLI models, whereas SemBench-Hard checks cited identifiers deterministically and calibrates the check with human judgments. SemBench-Hard measures the label–evidence difference inside SQPE execution as a paired Base–Hard gap per operator, splits the gap into grounding and decision checks by scenario and operator family, and uses three human annotators to check how often grounding-only rejections are judged broadly valid.

Hardened and perturbation-based evaluation. Adversarial distractors (Jia & Liang, 2017), contrast sets (Gardner et al., 2020), behavioral testing (Ribeiro et al., 2020), and templated variants such as GSM-Symbolic (Mirzadeh et al., 2025) show that standard test accuracy can overstate capability. Our Hard tasks add distractors and change output contracts. Reported scores use deterministic verifiers rather than LLM judges, whose biases and agreement with humans have been studied (Zheng et al., 2023; Wang et al., 2024).

Relative to these works, SemBench-Hard contributes paired Base and Hard tasks for SQPE operators with separate grounding and decision checks, and final-output scoring of generated programs under a fixed executor, reported overall and on executions where the reference program is accepted.

B Benchmark Construction and Scoring

This appendix specifies the relational views, the Base→Hard construction, the grounding rules and why they are checked exactly, the operator contracts, the chains, the adapters, the aggregation rule, and the Hard+tool settings.

B.1 Scenarios and Relational Views

The benchmark uses small relational views to make table, field, and intermediate-output bindings inspectable. Base and Hard tasks share the schema and differ in candidate sets and contracts. Table 5 separates task inputs from the evaluator-private `gold_` relations, which only the verifiers read and no prompt contains. The translation prompts render these views as a public schema with task-context columns (Appendix C).

Table 5: Task-input and evaluator-private (`gold_`) relations per scenario.

Scenario	Relational view

SWE-bench Verified	`issues(issue_id PK, repo, title, body, failing_test_log)`
	`candidate_files(issue_id FK, file_path, file_content)`
	`candidate_patches(issue_id FK, patch_id PK, diff_text)`
	Evaluator-private: `gold_repair_spec(issue_id PK, edited_files, edited_symbols, edit_ops)`

SciFact	`claims(claim_id PK, claim_text)`
	`abstracts(abstract_id PK, paper_id, abstract_text)`
	`abstract_sentences(abstract_id FK, sentence_idx, sentence_text), PK=(abstract_id, sentence_idx)`
	Evaluator-private: `gold_evidence(claim_id FK, abstract_id FK, sentence_ids, label)`

ContractNLI	`contracts(contract_id PK, contract_text)`
	`contract_clauses(contract_id FK, clause_id PK, span_start, span_end, clause_text)`
	`hypotheses(hypothesis_id PK, contract_id FK, hypothesis_text)`
	Evaluator-private: `gold_evidence(hypothesis_id FK, clause_id FK, label)`

WDC Products	`offers_left(offer_id PK, brand, title, price, description)`
	`offers_right(offer_id PK, brand, title, price, description)`
	Evaluator-private: `gold_pairs(left_offer_id FK, right_offer_id FK, label, decisive_attributes)`

B.2 Candidate Pools and Construction Rules

Table 6 gives the per-task Base and Hard candidate pools and the added context, and Table 7 the construction and acceptance rules for each scenario and operator. The Ctx. column counts visible failing tests (SWE Filter), same-repository patches (SWE Join), distractor issues (SWE Map), paper-body sections (SciFact), distractor clauses (ContractNLI), and distractor offers (WDC Products). Filter and Join tasks other than ContractNLI Filter keep one candidate and add context, and SWE-bench Filter/Join and ContractNLI Join also replace the Base negative with a harder one (a sibling path, a false-fix mutation of the gold patch, or a high-overlap span), so Base and Hard tasks are paired by anchor and construction role. Single-operator SciFact Filter and Join tasks judge one claim–abstract pair, and Join also returns evidence sentence ids. SciFact Map and the SciFact, ContractNLI, and WDC Products Rank tasks shuffle their candidates with a per-anchor seed. Every distractor is selected by a lexical or structural rule (token overlap, shared directories or tests, brand or price proximity, exception markers); none is chosen by a model.

Label distributions in the released loaders are: SWE-bench FILTER 500/499 and JOIN 323/323 (positive/negative); SciFact pair tasks 196 support, 87 contradict, and 587 no-evidence, and MAP 194/85/304; ContractNLI FILTER 320 per label, JOIN 300 evidence and 600 non-evidence, and MAP 300 per label; WDC Products 500 matches and 500 non-matches.

Table 6: Candidate pools per task: median [min–max], or one value if constant. Cand.: primary candidates; Ctx.: added context units. SWE-bench RANK Hard keeps the 315 issues with at least four strong distractors; restricting Base to them changes the SWE-bench RANK gap from -9.1 to -9.0 points (Appendix G.4). SciFact counts are those of the released loader on our PMC fetch; evaluated counts are in Table 10.

		Base			Hard		
Scenario	**Op.**	n	Cand.	Ctx.	n	Cand.	Ctx.
SWE-bench Verified	FILTER	999	1	0	999	1	1 [1–438]
	JOIN	646	1	0	646	1	3
	RANK	497	4	0	315	7	0
	MAP	500	2 [2–4]	0	500	3 [1–4]	2 [0–2]
SciFact	FILTER	870	1	0	870	1	4 [1–12]
	JOIN	870	1	0	870	1	4 [1–12]
	RANK	279	4	0	279	12	0
	MAP	583	3	0	583	12	0
ContractNLI	FILTER	960	3 [3–19]	0	960	77 [18–354]	5
	JOIN	900	1	0	900	1	5
	RANK	800	5	0	800	12	0
	MAP	900	6 [6–19]	0	900	10	0
WDC Products	FILTER	1,000	1	0	1,000	1	2 [0–3]
	JOIN	1,000	1	0	1,000	1	2 [0–3]
	RANK	313	4 [3–4]	0	313	6 [4–8]	0
	MAP	1,000	1	0	1,000	1	2 [0–3]

B.3 GROUNDING AND ACCEPTANCE RULES

Grounding enters acceptance only where Table 7 lists it. ContractNLI Base MAP already requires exact spans, so its Base–Hard change is the span pool (a window of at least six spans, median 6, becomes a fixed 10-span inventory with distractors) and the basis. ContractNLI JOIN judges one (hypothesis, span) pair and SWE-bench FILTER one (issue, file) pair, so the judged candidate is itself the evidence and their Hard contracts check the label. Every grounding check below compares the cited items with the dataset's annotated evidence or with a set derived from the records by a fixed rule, so it is a deterministic check against that reference. For each scenario, Appendix G.1 rescores the gap under a tolerant rule, and Appendix F reports how often annotators judge a rejected output sufficiently grounded.

WDC Products. Accepted attribute sets are derived deterministically from the source records. An attribute is a *matched* signal when both values are present and agree: brands are equal after case-folding, the titles share at least one lower-cased alphanumeric token, or the prices differ by at most 5% of the candidate offer's price. It is a *conflicting* signal when both values are present and disagree; a missing value produces neither. The gold set of a match is its matched signals; that of a non-match is its conflicting signals, plus title whenever the pair is a WDC hard negative or the titles share a token; an empty set defaults to {title}. Agreeing attributes of a non-match do not separate the offers and so are not decisive, while its title carries the model numbers and capacities that separate variants. Table 8 gives the resulting gold sets.

Hard FILTER, JOIN, and RANK accept an output only if the label (for RANK, the single designated best candidate) is correct and the cited attribute names equal the gold set. Hard MAP also requires the basis (same core product for matches, variant or bundle mismatch for WDC hard negatives, insufficient support otherwise) and requires the union of its matched and conflicting hints to equal the gold set. The Hard RANK gold is the positive with the most shared title tokens, then the smallest price gap; other positives of the same anchor offer are not placed in the pool.

Table 7: Base→Hard construction and acceptance rules by scenario and operator (F/J/R/M: FILTER/JOIN/RANK/MAP). SWE-bench distractor score: 100 per shared file, 60 per shared test module, 15 per shared directory (at most four), and one per shared issue token (at most 12).

Scen.	Op.	Base → Hard construction	Acceptance (Base; Hard)
SWE	F	(issue, file); Base negative: first non-gold path from another same-repo gold patch; Hard negative: non-gold path with the longest shared directory prefix with a gold file; Hard adds constraints and visible failing tests.	Label; label.
	J	(issue, patch); Base negative: least-overlapping valid same-repo patch; Hard negative: false-fix mutation of the gold patch (drop last hunk / swap last $-/+$ pair / drop last addition) plus 3 strongest same-repo context patches; issues with $\geq$2 strong distractors and top score $\geq$100.	Label; label and exact `decision_basis`.
	R	Base: gold + 3 least-overlapping same-repo patches; Hard: gold + up to 4 false-fix variants + strongest same-repo distractors (7 total), issues with $\geq$4 strong distractors and top score $\geq$120; ids anonymized; the gold patch is listed first in both conditions, a conservative placement (rotating it widens the gap by 10.0 points; Appendix G.4).	Top-1 = gold; top-1 = gold (cited files/tests recorded, not required).
	M	Base: issue code blocks, gold-file snippets, one visible-test snippet, all failing tests, oracle file hints; Hard: one code block, one reduced gold snippet (source-tagged), snippets of 2 same-repo distractor issues, one true + one distractor test, no file hints, size caps.	Files overlap and (symbols or ops or tests) overlap; files $\subseteq$ gold, tests $\subseteq$ gold, (symbols or ops) $\subseteq$ gold, caps.
SciFact	F/J	(claim, abstract): each gold abstract + one lexical-overlap distractor per gold abstract; Hard adds the full paper body and a direct-evidence constraint.	Label; label and sentence ids equal to one annotated rationale (empty for no evidence).
	R	4 $\rightarrow$ 12 abstracts: gold + cited non-evidence + lexical-overlap distractors.	Top-1 $\in$ gold; same.
	M	3 $\rightarrow$ 12 abstracts: Base one gold + distractors; Hard all gold + distractors.	Label; label and evidence equal to one single-abstract annotated rationale.
CNLI	F	Base: window of at least 3 spans around the gold spans; Hard: full contract, all spans, 5 distractor clauses (hypothesis overlap, exception markers).	Label; label and span set equal to one annotated variant.
	J	(hypothesis, span): Base negatives low-overlap/far; Hard negatives high-overlap/near, plus 5 distractor clauses.	Label; label.
	R	5 $\rightarrow$ 12 spans: gold + easy $\rightarrow$ hard negatives.	Top-1 $\in$ gold; same.
	M	Window of at least 6 spans $\rightarrow$ 10-span inventory (gold variant + distractors).	Label and span set equal to a visible variant; plus basis from the exception-marker rule.
WDC	F/J	Offer pair; Hard adds up to 3 distractor offers of the same anchor offer (hard negatives, same brand, text overlap, price proximity) and explicit constraints.	Label; label and attribute set equal to the gold set.
	R	$\leq$4 (all positives + easiest negatives) $\rightarrow$ $\leq$8 (best positive + $\leq$3 hardest same-group negatives + same-brand/title-overlap offers of other anchor offers).	Top-1 $\in$ positives; top-1 = designated positive and attribute set = gold set.
	M	Offer pair; Hard adds up to 3 distractor offers and explicit constraints.	Decision, ids, cluster flag, non-empty disjoint hints; plus basis and matched $\cup$ conflict hints = gold set.

WDC Products prompts and the exact-set rule. The Hard prompts ask for the names of the decisive attributes (`decisive_attributes`, or matched and conflicting hints in MAP), and the MAP and RANK prompts ask for "the minimal decisive attributes". Strict scoring accepts the cited set when it equals the gold set, the visible attributes that agree (for a match) or conflict (for a non-match) under the rule above. Three results quantify the effect of this rule. Under the tolerant rule (Appendix G.1), which accepts any non-empty subset of the gold set and so any minimal set of gold signals, the WDC Products gap is -30.9 points (strict -48.0): -42.8 in MAP and -38.4 in RANK, whose prompts ask for the minimal attributes (strict -51.4 and -43.9), -29.4 in FILTER (strict -46.6), and -12.9 in JOIN (strict -50.1), which includes the native joins that return no attribute set (Appendix B.7). Accepting minimal answers thus changes the MAP and RANK gaps by less than 9 points. Of the Hard grounding-only rejections that remain under the tolerant rule, 45% of the 2,810 in MAP give a wrong basis and 55% cite no attribute or one that is not a signal of the decision under the rule, as do all 627 in RANK. The prompts fix the attribute vocabulary to {brand, title, price}; model numbers and capacities appear in titles and are cited as title. Without WDC Products, the gap is -18.2 (Table 28). Annotators judge 33 of 70 sampled WDC Products grounding-only rejections broadly valid and 29 insufficiently grounded (Appendix F), and the calibration credits that scenario's rejections at its own rate.

Table 8: WDC Products gold attribute sets (task counts).

Gold set	F/J/M match	F/J/M non-match	RANK
{title}	397	97	247
{price, title}	43	368	29
{brand, title}	55	5	34
{brand, price, title}	4	30	2
{brand}	1	0	1

SciFact. Hard pair and MAP tasks accept cited evidence only if it equals, as a set of abstract sentence indices, exactly one annotated SciFact rationale for the cited abstract. Multi-rationale unions, subsets, and supersets are rejected, and no-evidence claims require an empty citation. MAP evidence must come from a single abstract of the 12-abstract pool. The full paper body is shown as context; evidence pointers are abstract sentences. A full RANK ranking must be a permutation of the pool.

ContractNLI. Hard FILTER and both MAP tasks accept cited clauses only if the span-id set equals one annotated evidence variant for the gold label (empty for not mentioned). Hard MAP also requires the basis implied by an exception-marker rule: insufficient support for not mentioned, exception clause if a gold span contains a marker such as "except", "unless", or "notwithstanding", and direct support otherwise.

SWE-bench Verified. The gold repair specification is derived from the gold patch: edited files from the diff headers, edited symbols by a regular-expression rule (at most four), edit operations by line-pattern rules, and tests from the `FAIL_TO_PASS` list. Hard MAP accepts a repair specification when its files and addressed tests are non-empty subsets of the gold values (tests matched by name family) and either its symbols or its edit operations match the gold values (a symbol matches when it shares an identifier token with a gold symbol, and a lone logic-change operation matches any non-test gold operation), within per-instance size caps; Base MAP requires overlap instead of subsets. Returned test names are compared with the gold names.

B.4 OPERATOR CONTRACTS AND NATURAL OUTPUTS

Each verifier checks the operator's natural output (Table 9) and, in Hard, the grounding fields the contract adds to the output schema; no operator is scored on free-form evidence text.

B.5 FORMAL PAIRING AND A SCIFACT EXAMPLE

The Base–Hard gap measures the combined effect of the changes defined below on the same anchors. A Base task is

$$T_{\text{base}} = (x, q, y^{\star}_{\text{base}}),$$

Table 9: Operator families, LOTUS-style signatures, natural outputs, and checked outputs.

Family and signature	Natural output	Checked output
FILTER: `df.sem_filter(predicate)`	Boolean label or selected row set	Label or set; in Hard, the cited grounding where Table 7 requires it.
JOIN: `df.sem_join(other_df, predicate)`	Matched row pairs or pair label	Pair decision or pair set; in Hard, decision basis (SWE-bench), evidence sentences (SciFact), or attribute set (WDC Products).
MAP: `df.sem_map(instruction)`	Generated field or JSON artifact	Schema and required fields; repair specification, evidence, spans and basis, or product attributes.
RANK: `df.sem_topk(criterion, K)`	Ordered top-K row identifiers	Top-1 choice and, where required, a complete permutation; attribute set in WDC Products Hard only.

where x is the visible input record or candidate set, q is the natural-language operator specification, and $y^{\star}_{\text{base}}$ is the expected output. The Hard task is

$$T_{\text{hard}} = (x, c, q, \Gamma, \Sigma, V, y^{\star}_{\text{hard}}),$$

where c is auxiliary context, Γ the explicit constraints, Σ the output schema, V the verifier, and $y^{\star}_{\text{hard}}$ the expected output under the Hard contract, which can differ from $y^{\star}_{\text{base}}$ when the constraints, schema, or stricter checks change it. The transformation

$$\mathcal{H}(T_{\text{base}}) \to T_{\text{hard}}$$

keeps the anchor and operator family; each component can be reverted on its own for the leave-one-out ablation (Appendix G.2).

SciFact MAP example. A Base SciFact MAP task asks for a support/contradict/no-evidence label for a claim given three candidate abstracts. Its Hard task expands the pool to 12 abstracts with distractors and requires a structured label/evidence output (Figure 6); Appendix B.3 gives the evidence rule.

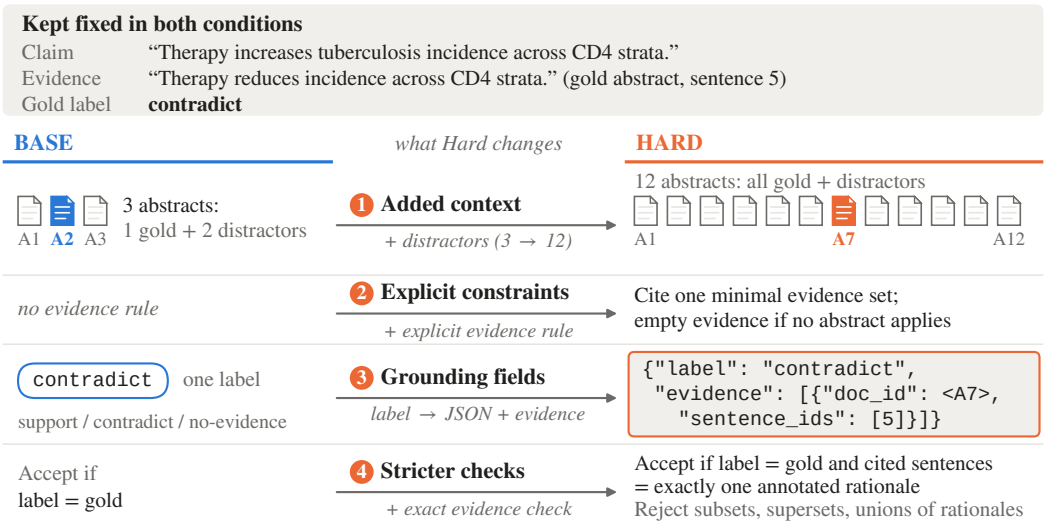


Figure 6: A SciFact MAP Base/Hard pair with the claim, evidence, and gold label fixed. Claim and evidence are paraphrased; pool positions and the sentence index are examples.

B.6 CANONICAL CHAINS

Chains run LOTUS × Qwen3 on the 100 anchors of each scenario; the chain runner executes FILTER steps as a structured `sem_map` followed by a drop and JOIN steps as a cross join followed by a

per-pair `sem_map`. The chain RANK step checks that its top-1 row is a gold row, which holds whenever the upstream JOIN step is accepted, so prefixes of two and three operators coincide. A step can contain several per-row sub-tasks (for example, one FILTER judgment per candidate), and it is accepted only if all of them are accepted; a step whose input is empty because an upstream step correctly removed every row is not scored. All-step acceptance conjoins the scored step verdicts, and errors count as failures. The prefix of length k is read from the four-operator run of the same anchor and condition. The RANK and MAP steps receive the rows passed on by the upstream steps, and under Hard the SciFact and SWE-bench MAP steps are scored by the Hard MAP verifier. Before any run, gold outputs fed through every step and prefix are accepted in both conditions.

B.7 EXECUTION ADAPTERS

The adapters run FILTER, MAP, and RANK tasks as structured MAP-style LLM calls in all three systems, and SciFact, ContractNLI, and SWE-bench JOIN use the same adapter. WDC Products JOIN uses native singleton-pair joins in LOTUS and Palimpzest, which take the offer pair and constraints and return a label; DocETL uses a MAP-based pair decision. Palimpzest runs its greedy optimizer with the evaluated backbone as its single model, LOTUS runs without cascades, and DocETL fills the LOTUS prompt text into its operation template and applies single operations through its pandas accessor, without its pipeline rewrite optimizer, because the execution axis fixes the program. Hard+tool runs call the backbone through a shared task-local tool runtime with the system's Hard prompt (Appendix B.9).

Label-only cells. The native WDC Products joins return a label, and the parser reads a missing attribute set as the default {title}, the gold set of most tasks (Table 8); Hard acceptance in these cells is thus an upper bound. The LOTUS SciFact Hard JOIN prompt, whose text DocETL also fills in, requests a label field and, as every Hard SciFact pair task, is scored on label and sentence ids; 0–83% of these outputs carry ids. Eight cells carry a grounding field in under 10% of Hard outputs: the six native WDC Products joins (LOTUS and Palimpzest with the three backbones) and LOTUS SciFact Hard JOIN with GPT-5.4-mini and Claude Sonnet 4.6. Without these eight cells the gap is -24.9 points ($[-25.5, -24.3]$; grounding-relaxed -6.7), and without all five LOTUS and DocETL SciFact JOIN cells (LOTUS with the three backbones, DocETL with GPT-5.4-mini and Qwen3) it is -26.1 (Table 28; grounding-relaxed -6.6).

Grounding-field rerun. We reran the Hard JOIN tasks of LOTUS and DocETL on SciFact and of LOTUS and Palimpzest on WDC Products with the grounding field requested: the SciFact Hard JOIN prompt requests the label and the sentence ids in the wording of the Hard FILTER prompt, and the WDC Products joins run on the structured MAP-style path of WDC Products Hard FILTER, which returns the attribute set. The rerun covers the seven Qwen3 and Claude Sonnet 4.6 cells on the task keys of their primary Hard records with the settings of the main runs (4,786 records); the GPT-5.4-mini cells and the Base condition use the primary records. Every rerun WDC Products output carries an attribute set, and every rerun SciFact output that predicts an evidence label carries sentence ids. On these seven cells the gap changes from -33.6 points ($[-36.1, -31.0]$) to -25.1 ($[-27.8, -22.5]$): from -15.0 to -4.8 on the three SciFact cells and from -47.5 to -40.3 on the four WDC Products cells. Replacing their Hard records in the primary cells gives a gap of -25.5 ($[-26.0, -24.9]$) and a grounding-relaxed gap of -6.1, against -25.9 and -6.3 (Table 28); every scenario, operator family, system, and backbone still loses at least 12 points (SciFact -12.5).

B.8 INVENTORY AND AGGREGATION

Aggregation rule. A cell's rate in a condition (Base, Hard, or Hard+tool) is its accepted records divided by its observed records; no record is extrapolated or reweighted. The primary set contains the 140 cells with observed Base and Hard records (GPT-5.4-mini 48, Qwen3 48, Claude Sonnet 4.6 44); the four cells of the 144-cell grid without observed records are DocETL $\times$ Claude Sonnet 4.6 on SciFact, and no cell is imputed. Keeping only the two backbones with complete grids gives a gap of -25.7 (Table 28). Every reported row is the unweighted mean of its cells' rates, and a gap is the mean of per-cell Hard$-$Base differences; because rows contain different numbers of cells, the All row is not the mean of the displayed rows. Intervals are 95% percentile intervals of a stratified

bootstrap that resamples records within each cell and condition independently (2,000 replicates, seed 20260925); Appendix G.1 adds a task-cluster bootstrap.

Cell inventory. The primary cells contain 83,972 Base, 82,880 Hard, and 64,518 Hard+tool records. Claude Sonnet 4.6 runs on the full inventory in 12 of its Base and Hard cells (LOTUS ContractNLI and SciFact, Palimpzest SciFact) and on label-balanced subsamples of 30–100 records in the other 32, taken as the first label-balanced tasks in loader order; its Hard+tool cells are subsamples of 15–100 records. Its gap, -26.4, is within 1.0 point of the fully evaluated backbones (Table 22). Hard+tool runs cover 124 of the primary cells (Claude Sonnet 4.6 40, GPT-5.4-mini 36, Qwen3 48), the tool-run cells: 115 allow up to three tool calls per operator (the three-call cells), and the nine SWE-bench RANK cells allow eight (Appendix I).

Infrastructure records. Records whose verdict is a system error, or that carry a runner error and no raw output, are infrastructure records (Base 651, Hard 570, Hard+tool 448 in the primary cells). They count as non-accepted, as in the recorded verdict; Appendix G.1 excludes them as a check.

Task inventory and pairing. Table 10 gives the evaluated Base inventory. Qwen3 has 12,099 Base records per system, with SciFact counts of 864/864/580/276 for FILTER/JOIN/MAP/RANK. Pairing by anchor and construction role (Appendix B.2) yields 11,947 pairs per system for GPT-5.4-mini and 11,917 for Qwen3 (Appendix G.1). SciFact counts are the evaluated task keys of the full-text snapshot (Appendix K); the released loader builds its tasks from the paper bodies fetched from PMC (870/870/583/279 in our fetch).

Table 10: Evaluated Base tasks per system (counts, GPT-5.4-mini); the Qwen3 SciFact counts are given in the text.

Scenario	FILTER	JOIN	MAP	RANK	Total
SWE-bench Verified	999	646	500	497	2,642
SciFact	874	874	585	281	2,614
ContractNLI	960	900	900	800	3,560
WDC Products	1,000	1,000	1,000	313	3,313
Total	**3,833**	**3,420**	**2,985**	**1,891**	**12,129**

B.9 HARD+TOOL SETTINGS

The Hard+tool condition (Observation 9) tests whether task-local deterministic tools repair Hard tasks under the unchanged verifier. Each operator may make up to three tool calls (eight for SWE-bench RANK). Each tool has a Python-style call signature, JSON input/output schema, deterministic semantics, and per-call logging. Calls are recorded in a `ToolTranscript` stored with each run record and count against the call budget; records without a tool transcript are errors or early exits and keep their recorded verdict. Tool calls carry no separate API charge; operator-level cost comes from the surrounding model invocations (Appendix D).

Tool profiles are assigned per backbone outside SWE-bench (Table 11), and Table 37 reports the tool change per backbone: $+4.5$ (GPT-5.4-mini), $+0.8$ (Claude Sonnet 4.6), and $+6.2$ points (Qwen3) on the three-call cells. Qwen3 runs use scenario-specific profiles, except ContractNLI MAP; GPT-5.4-mini runs use the SciFact local profile and, for ContractNLI, the shared retrieval profile; Claude Sonnet 4.6 runs use the shared retrieval profile for SciFact, ContractNLI, and WDC Products. The shared and SciFact local profiles include `doc_retrieve`, a keyword lookup over the documents of the current task.

The WDC pair helper takes no argument and reads the current task's `left_offer` and `right_offer`. Its rank counterpart compares the anchor offer with each visible candidate and returns attribute signals in input order. Output helpers validate and serialize an operator-specific decision. Complete tool schemas are in `modes/tools.py`. The Qwen3 WDC Products and ContractNLI profiles provide the tool guidance and output helpers; their scenario helpers take no task fields. In the six Claude Sonnet 4.6 WDC Products JOIN and RANK cells, `doc_retrieve` indexes the full offer records (with cluster ids, and candidate labels in RANK).

Tool guidance. Hard+tool runs call the backbone through the shared tool runtime rather than through the system's operators. Its system message states the tool-call protocol, and its user turn holds the system's Hard prompt (the LOTUS prompt, the Palimpzest description and instruction, or the DocETL prompt), the task row, and, per tool profile, operator-specific guidance (`systems/adapters.py`): SWE-bench FILTER states that the labels are balanced and asks to prefer `relevant` when in doubt; SWE-bench JOIN describes false-fix variants as near-gold patches with a subtle reversal or omission; SWE-bench RANK lists the false-fix mutation patterns (a dropped first, middle, or last hunk; a reversed first or last $-/+$ pair) and asks for the candidate with the most hunks and no reversed pair; SWE-bench MAP asks to call the repair-specification helper first; the Qwen3 SciFact FILTER, JOIN, and MAP and ContractNLI FILTER profiles ask for the minimum evidence set, "usually exactly one id"; ContractNLI FILTER adds that most contracts entail or contradict the hypothesis and asks to use `not_mentioned` sparingly; ContractNLI JOIN asks to match modal classes such as permission and prohibition; ContractNLI RANK states that the verifier scores the top-1 choice only; the WDC Products profiles fix the format of the attribute arrays and, for FILTER and JOIN, ask for the attributes whose match status supports the decision; and the shared retrieval profile recommends searching and opening items before deciding. All guidance and output routing are set to favor recovery, so Hard+tool acceptance measures tool repair under favorable settings. In SWE-bench, the output path takes the fields from the helpers: in MAP, when the repair-specification helper was called, the edited files are the ones it proposes from the shown snippets, returned symbols are narrowed to those it also proposes (if any), and the test is chosen by a token-overlap ranker over the predicted, proposed, and shown tests; in JOIN, the label and decision basis come from the `propose_join_decision` call when it returns valid values; and in RANK, the chosen patch id comes from the final output or, if absent, from the `propose_rank_choice` call.

Table 11: Tool profiles, the operators they serve, and the backbones that use them (G: GPT-5.4-mini; S: Claude Sonnet 4.6; Q: Qwen3); a profile can expose different tools per operator.

Profile id	Op.	Used by	Active tools
`local_retrieval_and_bounded_eval`	all	S (SciFact, ContractNLI, WDC); G (ContractNLI); Q (ContractNLI M)	`doc_retrieve`, `sentence_search`, `eval_expression`
`scifact_local_tools`	F/J/M/R	G	`doc_retrieve`, `sentence_search`, `sentence_features`, `numeric_compare`, `eval_expression`
`scifact_evidence_tools`	F/J/M	Q	`sentence_search`, `body_context_search`, `numeric_compare`
`scifact_rank_tools`	R	Q	`compare_candidates_for_claim`
`contractnli_evidence_tools`	F/J	Q	F: `sentence_search`, `classify_clause_type`; J: `classify_clause_type`
`contractnli_rank_tools`	R	Q	`classify_clause_type`
`wdc_pair_tools`	F/J/M	Q	`compare_offer_pair_attributes`
`wdc_rank_tools`	R	Q	`rank_offers_by_attribute_match`
`swebench_filter_path_tools`	F	G, S, Q	`extract_path_signals`, `module_proximity`, `propose_filter_label`
`swebench_join_decision_tools`	J	G, S, Q	`artifact_search`, `open_artifact`, `test_target_search`, `inspect_candidate_patch`, `propose_join_decision`
`swebench_map_semantic_tools`	M	G, S, Q	`propose_repair_spec`, `artifact_search`, `open_artifact`, `test_target_search`
`swebench_rank_choice_tools`	R	G, S, Q	`compare_rank_candidates`, `read_candidate_patch`, `propose_rank_choice`

Table 12: Representative tool signatures, return fields, and uses; an `accepted` field reports whether the proposed output is well formed.

Tool family	Signature and return fields	Typical use
SciFact retrieval	`def sentence_search(query: str, top_k: int) -> list[SentenceHit] [{"doc_id", "sentence_id", "section", "text", "score"}]`	Retrieve admissible abstract evidence; companion helpers expose body-only context and numeric comparisons.
Contract structure	`def classify_clause_type( clause_text: str) -> ClauseTypeReport {"clause_type", "exceptions", "obligations", "score"}`	Expose clause type and exception cues for entailment and evidence-span reasoning.
WDC attribute grounding	`def compare_offer_pair_attributes() -> PairAttributeReport {"brand": {...}, "title": {...}, "price": {...}}`	Compare the task-bound offer pair using visible brand, title, and price signals.
SWE artifact retrieval	`def artifact_search(query: str, top_k: int) -> list[ArtifactHit] [{"artifact_id", "title", "snippet", "score"}]`	Search shown files, diffs, and failing-test text within the current SWE-bench task.
SWE output helpers	`def propose_join_decision(label: str, decision_basis: str) -> JoinDecision {"label", "decision_basis", "rationale", "accepted"}`	Validate and serialize a JOIN decision; separate helpers serve other operators.

C TRANSLATION PROTOCOL AND ADDITIONAL RESULTS

This appendix supports Observations 1–3 and Table 3. The translation axis asks whether a translator writes a program whose final output answers a natural-language query over a public schema. Scoring ignores program structure: a program that differs from the reference program in operators, order, or decomposition is accepted whenever its final output passes the query's verifier. The reference-relative types of Appendix C.4 are diagnostics and do not affect Correct. Because the reference-conditional rate depends on which executions the executor accepts, every style comparison is reported under two executors, on 70 anchors, and under a repeat run, and equivalent rewrites of reference programs check that the executor does not penalize a different program structure (below).

C.1 QUERIES AND PROMPTS

Queries. The query set covers SciFact, ContractNLI, and WDC Products: 15 query templates per scenario, three for each program length from one to five operators, with varied operator compositions (for example FILTER→JOIN→MAP, FILTER→RANK→MAP, and JOIN→FILTER→MAP). Each template has two queries: the step-by-step query joins one clause per reference operator in operator order, so it closely mirrors the reference operator sequence, and the goal-only query is one sentence that states only the goal. Both end with the same output contract, which fixes the requested fields, identities, and output form (90 queries). Programs use only the four semantic operators, and JOIN predicates state key equalities in natural language together with a semantic condition, for reference and generated programs alike. Some templates share a goal, so the 30 queries of a scenario yield 24 (SciFact), 24 (ContractNLI), and 26 (WDC Products) distinct prompts; queries that share a prompt share its generated program and have the same frozen target. Figure 8 shows one template.

Prompts. A prompt consists of a fixed system message with scenario-specific interface rules (Figure 7), five operator-syntax demonstrations on toy tables that are not task tables, the public schema of the scenario, and the query. It contains no evaluator relations, sample rows, reference programs, or scenario data. Both queries of a template receive the same prompt apart from the query text.

Translation system message

```
You translate a natural-language data-processing question into a
LOTUS-style semantic-operator program.  Output ONLY the program --
no commentary, no markdown fences.
The program is a sequence of operator calls on a dataframe, starting
with df = <table> (a relational table from the schema).  Decide the
chain length, primary table, and operator sequence yourself based on
the question.
Available operators:
df.sem_filter('<predicate over current rows>')
Keep rows for which the predicate holds.  Predicate is a
natural-language sentence referencing existing columns by
table.column.
df.sem_join(other_table_name, predicate='<predicate over the joined
pair>')
Pair current rows with rows in other_table_name (a relational table
from the schema), keeping pairs that satisfy the predicate.
df.sem_topk('<criterion>', K=<int>)
Sort rows by the natural-language criterion and keep the top K.
df.sem_map('<instruction emitting a structured per-row output>')
Produce a new column (or columns) per row by following the
instruction.  Used for emitting verdicts / labels / repair specs.
Do not invent operator names.  Output the program as plain
text only.  Do not wrap it in code fences.  Do not include any
natural-language explanation.  Do not include comments.
Additional executable-interface rules (SciFact):
- sem_map(instruction, suffix='answer') puts one output string in
column answer for every input row; it does not change the row count
or project source fields.
- Qualified source columns remain available after operators.
Reference columns explicitly as table.column inside instructions.
The singleton claims relation is available as context without a
Join; other relations require explicit Join.
- Empty relations remain empty through subsequent operators,
including Map.
- Intermediate relation variables and fluent chains are allowed.
Use only the declared four operators; no Python imports, loops,
helper calls or other APIs.
```

Figure 7: Translation system message, verbatim; the closing interface rules are scenario-specific (SciFact shown).

C.2 Generation, Execution, and Verification

Translators. Every translator runs with tools disabled and writes one program per distinct prompt; the GPT models run in an empty working directory.

Normalization and scored executions. Two normalizations are applied and logged in the scores. One surrounding Markdown code fence is removed, and for the GPT models, progress text before the first `df = <relation>` assignment is dropped without changing the program; the second affects 23 GPT-5.4-mini generations. Each translator is scored on 45 templates per style (450 executions); the GPT-5.4-mini executor is scored on 445 template–anchor pairs: five pairs with an infrastructure error in any execution are excluded for every program, so all rows stay paired.

Executors and anchors. The executor, Qwen3, runs every generated and every reference program through an interpreter for the four-operator language and unmodified native LOTUS operators; constructs outside the four operators make a program invalid. The second executor, GPT-5.4-mini, runs every generated and reference program with the same interpreter, anchors, targets, and verifier. The reference-relative types and all translation tables except Table 14 and the last two columns of

Template SciFact L1a: public schema, queries, and reference program

Public schema (in the prompt). `claims: claim_id, claim_text; abstracts: abstract_id, paper_id, abstract_text, title, body_sections, claim_text, candidate_abstract_text, policy_text; abstract_sentences: abstract_id, sentence_idx, sentence_text.`
Step-by-step. Keep rows whose `abstracts.abstract_text` contains evidence supporting or contradicting `claims.claim_text`.
Goal-only. Find all supplied abstracts containing evidence that supports or contradicts the claim.
Shared output contract. Use the supplied claim and candidate records only. Evidence means support or contradiction, not merely topical similarity. Sentence membership is judged in its full document context. [...] Preserve each returned record's abstract_id. Return unique record identities. Return every eligible record, not just one representative. If no records are eligible, return an empty collection.
Reference program (evaluator-private).

```
df = abstracts
df = df.sem_filter('abstracts.abstract_text contains scientific
evidence supporting or contradicting claims.claim_text; topical
similarity alone is insufficient')
```

Figure 8: One template (SciFact L1a) with its step-by-step and goal-only queries; "[...]" marks an omitted sentence. The public schema lists the task-context columns shared by all scenarios (Appendix B.1).

Table 3 use the first executor. Each scenario uses ten anchors: a seeded permutation (seed 20260925) of the scenario's 100-anchor table, keeping the first ten whose Base views and all 15 query targets can be constructed. SciFact anchors are drawn from claims with complete full text. Every program runs on the same anchors (a crossed program $\times$ anchor design).

Verification and outcome classes. The verifier checks only the extracted final output. Every program–anchor execution receives one class: *accepted*; *rejected* by the verifier; *output error* (the final relation lacks the requested form or fields); *program error* (raised before any operator runs: the program is invalid or reads a field absent from its relation; abbreviations such as "i.e." in a predicate are not treated as field references); *executor error* (a native operator failed on a model response); or *infrastructure error* (transport failure; retried and not scored). No executor errors occurred. Under the first executor, verifier rejections make up 55 and 39% (GPT-5.4, step-by-step and goal-only), 54 and 62% (GPT-5.4-mini), 53 and 30% (Claude Sonnet 4.6), 48 and 20% (Qwen3), and 66% for the reference program; Table 3 gives the other classes.

Empty targets. A target is empty when no record qualifies on an anchor; an empty output matches it. Among the reference executions, empty targets make up 0% of SciFact, 43.3% of ContractNLI, and 53.3% of WDC Products (145 of 450, 32.2%). Table 15 reports all-target rates, pooled and by scenario.

Uncertainty. Intervals come from a crossed bootstrap (2,000 replicates, seed 20260925) that resamples templates, keeping both queries of a template together, and anchors independently within each scenario; style differences (step-by-step minus goal-only) are paired within the same bootstrap draws.

C.3 Style Differences, Executors, and Paired Outcomes

Table 13 gives the paired style differences. Pooled over translators, goal-only queries lower the reference-conditional rate by 10.8 points on non-empty targets and by 15.4 points on all targets. Some templates share a goal-only prompt (29 distinct prompts for 45 templates), so we also resample the 29 goals, each with all its templates and both styles, instead of the templates. The pooled intervals move by at most 1.1 points: $[+1.5, +9.4]$ for Correct and $[-0.6, +22.3]$ for the reference-conditional rate on non-empty targets, and $[+2.3, +13.5]$ and $[+6.3, +25.2]$ under the second executor.

Table 13: Paired style differences, step-by-step minus goal-only (points), Qwen3 executor; brackets: 95% CI. Goal-clust.: the interval of Correct on non-empty targets when resampling the 29 goal-only prompts instead of templates.

	Non-empty targets		All targets		Goal-clust.
Translator	Correct	Ref.-cond.	Correct	Ref.-cond.	Correct
GPT-5.4	+1.3 [−1.6, +5.7]	+1.5 [−6.6, +11.3]	+6.9 [+1.3, +14.2]	+14.6 [+2.2, +29.2]	[−1.8, +5.6]
GPT-5.4-mini	+4.3 [+0.0, +10.1]	+10.4 [−6.2, +32.5]	+1.3 [−3.8, +6.4]	+6.6 [−4.2, +18.6]	[+0.0, +10.2]
Claude Sonnet 4.6	+5.2 [−0.3, +13.0]	+10.4 [−8.3, +33.9]	+12.4 [+5.6, +20.0]	+21.9 [+6.9, +37.7]	[−0.4, +13.9]
Qwen3	+8.9 [−0.3, +18.8]	+20.9 [−12.9, +52.9]	+8.7 [−2.2, +19.8]	+18.5 [−4.2, +39.9]	[−1.2, +20.5]
Pooled	+4.9 [+1.5, +8.8]	+10.8 [−0.9, +22.6]	+7.3 [+3.3, +11.9]	+15.4 [+7.0, +23.6]	[+1.5, +9.4]

Second executor. Table 14 compares the two executors on non-empty targets. GPT-5.4-mini accepts the reference program on 46.7% of executions (140 of 300, from 31 templates; 54 of these executions come from three- to five-operator templates), against 22.0% (67, from 20 templates) for Qwen3, and the reference-conditional rate of generated programs ranges from 11.4 to 55.0%. Goal-only queries lower Correct by 6.8 points and the reference-conditional rate by 14.8 points, and both intervals exclude zero. Of the 64 generated programs accepted on a non-empty target under the first executor, 54 are also accepted under the second, including 23 of the 27 that differ from their reference; under the second executor, 32 of 75 accepted programs differ from their reference (16 of 53 among programs accepted on every non-empty target where the reference program is accepted).

Translator–executor affinity. To check for an affinity between translator and executor, we compared each translator's change between executors with the mean change of the other three; pooled over scenarios, no such difference has an interval that excludes zero. Leaving out the translator that is also the executor, the pooled style difference in Correct is $+3.6$ points ($[0.8, 7.0]$) under the first executor and $+6.3$ ($[1.1, 12.1]$) under the second.

Repeat run. A second run of all 4,050 executions on Qwen3 agrees with the first on 93.8% of reference and 98.5% of generated executions with a non-empty target (Cohen's κ 0.82 and 0.93). On the 504 generated executions where both executors accept the reference, the two Qwen3 runs agree on 95.4% ($\kappa = 0.91$) and the two executors on 81.0% ($\kappa = 0.62$). The reference program is accepted on 67 of the 305 non-empty executions in the first run, 68 in the second, and 58 in both; requiring acceptance in both runs gives a pooled reference-conditional style difference of $+11.6$ points ($[-1.4, +25.5]$).

Equivalent rewrites. To check that the executor treats equivalent programs alike, we rewrote 24 multi-operator reference programs (33 rewrites) by moving a FILTER across a JOIN or a final MAP and by fusing adjacent FILTERs, and ran each rewrite and a repeat of the original on all 30 anchors. On the 518 non-empty executions of 22 templates, a rewrite agrees with the verdict of the original on 91.9% ($\kappa = 0.54$), against 90.9% ($\kappa = 0.50$) for the repeat, and changes acceptance by -2.3 points ($[-7.5, +2.2]$), against -1.4 ($[-6.3, +3.3]$) for the repeat.

Program length. Under GPT-5.4-mini, the pooled reference-conditional rate is 57.6% (step-by-step) and 34.9% (goal-only) on the 86 reference-accepted executions of one- and two-operator templates, and 9.7% and 7.4% on the 54 of three- to five-operator templates.

Seventy anchors. The Qwen3 executor also ran the same programs on 60 further anchors per scenario, drawn by the same seeded rule: 25,200 generated-program executions over 70 anchors, 484 of them reference-accepted with a non-empty target, from 26 templates. The reference-conditional rate ranges from 22.9 to 62.8% (25.9–62.4% on the first 30 anchors), the two GPT translators have the highest rates in both styles, about 40% of accepted generated programs differ from their reference, and goal-only queries lower Correct by 5.4 points ($[2.4, 8.6]$; 5.3 on 30 anchors) and the reference-conditional rate by 13.6 points ($[2.8, 24.0]$; 12.3 on 30 anchors).

Output column. The system message and every query with a MAP step name the output column `answer` (Figure 7); a final MAP that writes LOTUS's default column is an output error. We also

rescored the stored final relations of the 30 anchors, the first ten from the repeat run, reading the default column as the answer and keeping every other check. The pooled style difference becomes $+2.8$ points ($[-1.5, 7.7]$) for Correct and $+10.4$ ($[-3.0, 25.2]$) for the reference-conditional rate, against $+5.4$ and $+13.4$ under the strict rule on the same records (first ten anchors from the repeat run, hence the difference from 5.3 and 12.3 above), and the program-error shares are unchanged.

Table 14: Correct on non-empty targets (%) under the two executors. Program: program-error share; last row: pooled style difference, step-by-step minus goal-only (points); brackets: 95% CI.

		Qwen3		**GPT-5.4-mini**		**Qwen3**
Translator	**Style**	Correct	Ref.-cond.	Correct	Ref.-cond.	Program
GPT-5.4	Step	14.1	56.7	26.3	55.0	0.0
	Goal	12.8	55.2	22.3	46.4	0.0
GPT-5.4-mini	Step	14.1	55.2	21.3	45.0	0.0
	Goal	9.8	44.8	13.0	27.1	0.0
Claude Sonnet 4.6	Step	11.1	37.3	15.3	32.9	0.0
	Goal	5.9	26.9	5.7	11.4	34.4
Qwen3	Step	16.7	55.2	13.0	23.6	0.0
	Goal	7.9	34.3	7.7	12.1	37.7
Reference program	–	22.0	–	46.7	–	0.0
Style difference, pooled		+4.9 [+1.5, +8.8]	+10.8 [−0.9, +22.6]	+6.8 [+2.1, +12.4]	+14.8 [+6.1, +24.6]	

Reference acceptance by scenario. On non-empty targets the reference program is accepted on 67 of 305 executions (SciFact 17 of 150, ContractNLI 9 of 85, WDC Products 41 of 70), from 20 templates; under GPT-5.4-mini it is accepted on 31 of the 42 templates with a non-empty target, including all 20. Of the 202 verifier rejections, 121 return a set of the wrong size. Reference acceptance falls with program length: 55 of the 67 accepted reference executions under Qwen3 come from one- and two-operator templates (39.3% of their 140 executions, against 7.3% of 165 for three to five), and 86 of 140 under GPT-5.4-mini (61.4% against 33.8%). Table 15 gives the all-target results, pooled and by scenario; SciFact has no empty targets, so its columns are also non-empty rates.

Table 15: Correct on all targets (%), pooled and by scenario. Ref.-cond.: as in Table 3; last row: reference-accepted executions (count).

		Pooled	**SciFact**		**ContractNLI**		**WDC Products**	
Translator	**Style**	All	All	Ref.-cond.	All	Ref.-cond.	All	Ref.-cond.
GPT-5.4	Step	34.9	12.0	94.1	31.3	78.9	61.3	67.5
	Goal	28.0	12.0	100.0	18.0	43.9	54.0	63.6
GPT-5.4-mini	Step	33.1	10.7	76.5	27.3	71.9	61.3	71.4
	Goal	31.8	6.7	58.8	25.3	66.7	63.3	66.2
Claude Sonnet 4.6	Step	34.2	10.7	52.9	29.3	73.7	62.7	66.2
	Goal	21.8	2.7	23.5	20.0	52.6	42.7	45.5
Qwen3	Step	23.8	8.0	64.7	17.3	40.4	46.0	49.4
	Goal	15.1	2.7	23.5	4.0	8.8	38.7	45.5
Reference, all targets		33.6	11.3	–	38.0	–	51.3	–
Reference, non-empty targets		22.0	11.3	–	10.6	–	58.6	–
Reference-accepted executions, all targets		151	17		57		77	

Table 16 crosses each generated execution with the reference execution on the same template and anchor. On targets that contain a record, 6–15 executions per translator accept the generated program and reject the reference; most generated-only acceptances (44–78 per translator) are on empty targets. Of the 64 generated programs accepted on at least one non-empty target, 27 use an operator sequence different from their reference (GPT-5.4 4 of 16, GPT-5.4-mini 4 of 17, Claude Sonnet 4.6 7 of 11, Qwen3 12 of 20). Counting only programs accepted on every non-empty target where the reference is accepted, as a test-suite criterion would, 14 of 36 differ; a structural score would have counted them as errors.

Table 16: Generated versus reference outcomes per program–anchor execution (counts, both styles); generated-only acceptances are split by whether the target contains a record.

		Generated only			
Translator	**Both pass**	Non-empty	Empty	**Reference only**	**Both fail**
GPT-5.4	204	7	72	98	519
GPT-5.4-mini	208	6	78	94	514
Claude Sonnet 4.6	171	9	72	131	517
Qwen3	116	15	44	186	539

C.4 Structural and Relation Types

Typed cohort. Templates that share a goal-only prompt share one generated program, which is typed against each template's reference. The typed programs (n_0) are identified under the first executor, on reference-accepted anchors including empty targets. Typing covers programs whose template has a reference-accepted anchor (all but 12 per translator and style). Table 17 reports the types by translator. Restricting typing to non-empty targets gives 44 step-by-step and 59 goal-only programs never accepted there, with the same pattern: A1 2% against 34%, A2 2% against 25%, and A3 2% against 31%. The second executor accepts 5 of the 75 typed programs on a non-empty target where it accepts the reference; without them, A1–A3 are 10%, 15%, and 5% against 30%, 44%, and 36%.

Alignment. Programs are parsed in source order, resolving dataframe aliases and fluent chains. A kind-aware edit script aligns their operators to the reference (match cost 0; substitution, insertion, and deletion cost 1; ties resolved by position). A1 marks a reference operator without an aligned counterpart, including one whose predicate is folded into a neighboring generated step (at least 30% of its predicate tokens recur there); A2 an inserted operator; A3 an aligned operator of a different kind, including names outside the language. A4 marks a permutation of an unchanged operator multiset.

Relations, parameters, and scope. B1 records a different starting or joined relation, and D1 compares top-K and `sem_map` output names; no typed program carries D1. One unparsable Claude Sonnet 4.6 program is counted in n_0 without a structural type or a C1 judgment.

Table 17: Reference-relative types (%, overlapping) among the n_0 typed programs, by translator and style. Acc.: programs accepted on at least one reference-accepted anchor; C1 over parsable programs (all but one Claude Sonnet 4.6 goal-only program).

Translator	**Style**	Programs	Acc.	n_0	A1	A2	A3	A4	B1	C1
GPT-5.4	Step	45	30	3	0	0	0	0	67	67
	Goal	45	25	8	38	38	0	0	12	62
GPT-5.4-mini	Step	45	29	4	25	25	0	0	25	75
	Goal	45	27	6	17	33	17	0	17	83
Claude Sonnet 4.6	Step	45	29	4	0	25	25	0	50	50
	Goal	45	18	15	33	40	33	13	20	86
Qwen3	Step	45	22	11	9	9	0	0	9	100
	Goal	45	9	24	29	50	54	4	46	100

C.5 Predicate Judge (C1)

For every aligned operator pair in the typed cohort, a fixed prompt gives Claude Opus 4.7 the query, the public schema that the translators saw (including the anchor-context fields), the operator, the reference predicate with its K and output fields, and the generated predicate. The judge returns whether the two predicates target the same evidence unit and the same decision, treat abstention alike, and impose the same output constraint, followed by an overall equivalence judgment and a one-sentence rationale. C1 fires for a program when any of its aligned pairs is judged non-equivalent, and Claude Haiku 4.5 (Anthropic, 2025) judges the same pairs with the same prompt as a cross-check. Claude Opus 4.7 judges 75.4% of the 179 aligned pairs non-equivalent. At the program level, C1 fires

Table 18: Examples of reference-relative types (S: step-by-step, G: goal-only).

Type, scenario	Reference	Generated
A1 Missing operator ContractNLI (GPT-5.4, G)	FILTER → MAP	MAP
A2 Added operator SciFact (Claude Sonnet 4.6, G)	FILTER	FILTER → MAP
A3 Different operator WDC (GPT-5.4-mini, G)	FILTER → RANK → MAP	JOIN → RANK → MAP
A4 Different order WDC (Claude Sonnet 4.6, G)	JOIN → FILTER → MAP	FILTER → JOIN → MAP
B1 Relation difference WDC (GPT-5.4, S)	Starts from `offers_left`.	Starts from `offers_right`.
C1 Predicate difference SciFact FILTER (Claude Sonnet 4.6, G)	Whole abstract (`abstracts.abstract_text`) supports or contradicts the claim.	Single sentence (`abstract_sentences.sentence_text`), read in context, supports or contradicts the claim.

for 18 of the 22 step-by-step programs (82%) and 46 of the 52 judged goal-only programs (88%); every typed Qwen3 program carries C1 (Table 17). Claude Haiku 4.5 agrees with Claude Opus 4.7 on 85% of the 179 aligned predicates (Cohen's $\kappa = 0.52$); it judges more pairs non-equivalent (87.2%). C1 is diagnostic and does not enter Correct.

C.6 CORRESPONDENCE TO NL2SQL-BUGS

Table 19 adapts selected NL2SQL-BUGs categories. A1/A2 correspond to clause omission/redundancy; A3/A4 extend the mapping to operator substitution and order. SQL comparison and logical operators occur inside our natural-language predicates. Function-, subquery-, and `DISTINCT`-related types presuppose constructs absent from the operator language.

Table 19: Selected NL2SQL-BUGs categories adapted to Table 2; A3 and A4 are added types.

NL2SQL-BUGs	This paper
Clause-related (missing / redundancy)	A1 / A2
Table-related (including join condition and type)	B1
Condition-related (explicit / implicit)	C1
Value-related (value mismatch, data format)	C1 (literals), D1 (K, output schema)
Operator-related (comparison, logical)	— (natural-language predicates)
Function-, subquery-, `DISTINCT`-related	— (no such constructs)
—	A3 different operator / A4 order

D MODELS, RUNTIME, AND COST

D.1 MODELS

Table 20 lists the models by role. API models were called through their vendors' command-line interfaces with the aliases `gpt-5.4-mini` and `sonnet` and their default reasoning settings. GPT-5.4-mini ran through `codex exec` with a read-only sandbox in the benchmark working directory (Appendix G.1 reruns flagged and random records with file access disabled), and Claude Sonnet 4.6 through `claude -p` with tools disabled. The additional-system runs of Appendix G.5 call Claude Sonnet 4.6 and Claude Haiku 4.5 through the Amazon Bedrock Converse API at temperature 0. Qwen3 was served locally with vLLM at temperature 0 and receives only its prompt and, in Hard+tool runs, the tool outputs. Qwen3 execution runs use thinking in SciFact, ContractNLI, and WDC Products; in SWE-bench Verified only the RANK Hard+tool runs use it, with 8,192-token turns; these nine cells are reported apart from the 115 three-call cells (Appendix I.1). Translation and

translation execution run without thinking. Runs use at most 32 concurrent tasks, LOTUS runs with its cache disabled at temperature 0, and DocETL MAP calls bypass the cache.

Table 20: Models by role.

Role	Model
Execution backbones	GPT-5.4-mini (`gpt-5.4-mini`), Claude Sonnet 4.6 (`sonnet`), Qwen3-30B-A3B FP8 (`Qwen3-30B-A3B-FP8`)
Translators	GPT-5.4 (`gpt-5.4`), GPT-5.4-mini, Claude Sonnet 4.6, Qwen3-30B-A3B FP8
Translation executor	Qwen3-30B-A3B FP8
Second translation executor	GPT-5.4-mini
Predicate judges	Claude Opus 4.7 (`claude-opus-4-7`), Claude Haiku 4.5 (`claude-haiku-4-5`)
Additional-system runs	Claude Sonnet 4.6; Claude Haiku 4.5 as the proxy of the BARGAIN-inspired row
Frontier-backbone check	Claude Opus 5.5 (`claude-opus-5-5`)

D.2 LATENCY AND COST

Table 21 reports mean latency and API cost per record on the tool-run cells. Over those cells, the Hard+tool runs of the two API backbones cost 2.28 times as much as their Hard runs and take 1.75 times as long, as ratios of mean cell means (medians of the per-cell ratios 2.44 and 2.08). Qwen3 runs on a local GPU and has no API cost. On the 115 three-call cells, the API-backbone ratios are 2.19 for cost and 1.74 for latency (medians 2.37 and 1.95), and on the 106 of them outside the SWE-bench MAP helper cells, the population of Observation 9's $+1.0$, 2.24 and 1.75.

Table 21: Mean per-record latency (s) and API cost (USD) on the tool-run cells, by system and backbone; API latency includes one command-line process per call, so compare conditions within a row.

System	Backbone	Cells	Base (s)	Hard (s)	Hard+tool (s)	Base ($)	Hard ($)	Hard+tool ($)
LOTUS	GPT-5.4-mini	12	47.11	48.05	101.15	0.0053	0.0108	0.0194
LOTUS	Claude Sonnet 4.6	12	37.71	57.14	97.64	0.0117	0.0234	0.0614
LOTUS	Qwen3	16	1.96	4.78	13.97	0.0000	0.0000	0.0000
Palimpzest	GPT-5.4-mini	12	13.29	17.95	59.32	0.0046	0.0083	0.0219
Palimpzest	Claude Sonnet 4.6	16	41.69	52.80	77.54	0.0158	0.0318	0.0608
Palimpzest	Qwen3	16	2.64	7.95	14.71	0.0000	0.0000	0.0000
DocETL	GPT-5.4-mini	12	39.96	38.55	33.43	0.0047	0.0074	0.0180
DocETL	Claude Sonnet 4.6	12	38.76	43.35	87.28	0.0105	0.0226	0.0603
DocETL	Qwen3	16	2.04	7.66	14.10	0.0000	0.0000	0.0000

Cost computation. API cost is computed from recorded token usage and a public price list. For an API-backed operator execution,

$$\text{cost} = \frac{t_{\text{in}}}{10^6} r_{\text{in}} + \frac{t_{\text{cache_write}}}{10^6} r_{\text{cache_write}} + \frac{t_{\text{cache_read}}}{10^6} r_{\text{cache_read}} + \frac{t_{\text{out}}}{10^6} r_{\text{out}},$$

summed over the LLM calls of the execution.

E DETAILED EXECUTION BREAKDOWNS

This appendix supports Observation 4: every backbone, system adapter, and system–backbone pair loses 24–28 points (Tables 22–24). It also splits the Hard acceptance that grounding relaxation adds by the failed grounding check (Observation 6; Table 25) and gives the chain values of Figure 5. Tool recovery is in Appendix I.

On Base, relaxation adds 1.3 points, almost all from ContractNLI MAP evidence spans.

Table 22: Execution acceptance by backbone (%); Δ: Hard−Base (points); brackets: 95% CI.

Backbone	Cells	Base ↑	Hard ↑	Δ
GPT-5.4-mini	48	89.5 [89.2, 89.8]	63.6 [63.1, 64.1]	−25.9 [−26.4, −25.4]
Claude Sonnet 4.6	44	87.9 [87.1, 88.7]	61.5 [60.2, 62.7]	−26.4 [−27.9, −24.9]
Qwen3	48	80.7 [80.3, 81.1]	55.3 [54.8, 55.8]	−25.4 [−26.0, −24.8]
All	140	86.0 [85.7, 86.3]	60.1 [59.7, 60.5]	−25.9 [−26.4, −25.3]

Table 23: Execution acceptance by system (%), as in Table 22.

System	Cells	Base ↑	Hard ↑	Δ
LOTUS	48	84.6 [84.2, 85.0]	59.4 [58.7, 60.1]	−25.2 [−26.1, −24.4]
Palimpzest	48	86.6 [86.0, 87.1]	60.6 [59.7, 61.4]	−26.0 [−27.0, −25.0]
DocETL	44	86.9 [86.2, 87.5]	60.4 [59.6, 61.2]	−26.5 [−27.5, −25.4]
All	140	86.0 [85.7, 86.3]	60.1 [59.7, 60.5]	−25.9 [−26.4, −25.3]

Table 24: Execution acceptance by system and backbone (%), as in Table 22.

System	**Backbone**	Cells	Base ↑	Hard ↑	Δ
LOTUS	GPT-5.4-mini	16	88.9 [88.3, 89.4]	63.0 [62.2, 63.8]	−25.9 [−26.8, −24.9]
LOTUS	Claude Sonnet 4.6	16	85.5 [84.4, 86.4]	60.0 [58.2, 61.8]	−25.5 [−27.6, −23.4]
LOTUS	Qwen3	16	79.5 [78.9, 80.1]	55.1 [54.3, 55.9]	−24.4 [−25.3, −23.4]
Palimpzest	GPT-5.4-mini	16	90.0 [89.5, 90.5]	64.2 [63.4, 65.0]	−25.8 [−26.8, −24.8]
Palimpzest	Claude Sonnet 4.6	16	89.7 [88.3, 91.2]	62.2 [60.1, 64.4]	−27.5 [−30.1, −24.9]
Palimpzest	Qwen3	16	79.9 [79.3, 80.6]	55.3 [54.4, 56.1]	−24.7 [−25.8, −23.6]
DocETL	GPT-5.4-mini	16	89.7 [89.2, 90.2]	63.7 [62.8, 64.5]	−26.1 [−27.0, −25.1]
DocETL	Claude Sonnet 4.6	12	88.6 [86.6, 90.7]	62.5 [59.7, 65.2]	−26.1 [−29.4, −22.7]
DocETL	Qwen3	16	82.7 [82.1, 83.3]	55.5 [54.7, 56.3]	−27.2 [−28.2, −26.2]

Table 25: Hard acceptance points added by grounding relaxation, by failed grounding check. Other: the remaining grounding-only rejections, such as the basis failing together with another grounding check; rows may not sum because of rounding.

	Cells	Evidence	Attributes	Basis	Tests	Other	Total
SWE-bench Verified	36	0.0	0.0	1.7	5.1	0.0	6.8
SciFact	32	12.2	0.0	0.0	0.0	0.1	12.4
ContractNLI	36	8.6	0.0	2.4	0.0	1.7	12.8
WDC Products	36	0.0	45.2	1.4	0.0	3.9	50.5
FILTER	35	8.6	13.6	0.0	0.0	0.0	22.3
JOIN	35	3.6	12.7	1.7	0.0	0.0	18.1
MAP	35	7.9	9.9	4.0	5.2	5.9	32.9
RANK	35	0.0	10.2	0.0	0.0	0.0	10.2
All	140	5.0	11.6	1.4	1.3	1.5	20.8

Table 26: All-step acceptance of chain prefixes (%) by prefix length (values of Figure 5); Mean: over the four scenarios. SWE-bench Hard reaches 0 at two steps: a Hard FILTER alone lowers the two-step Base chain by 7 points and a Hard JOIN alone by 20 (Appendix G.6).

Scenario	Condition	1	2	3	4
SciFact	Base	80	71	71	71
	Hard	33	22	22	12
ContractNLI	Base	76	49	49	49
	Hard	52	47	47	46
WDC Products	Base	43	29	29	22
	Hard	9	5	5	5
SWE-bench Verified	Base	31	26	26	26
	Hard	14	0	0	0
Mean	Base	57.5	43.8	43.8	42.0
	Hard	27.0	18.5	18.5	15.8

F HUMAN VALIDATION OF VERIFIER OUTCOMES

This appendix supports Observation 6. A strict grounding check scores agreement with the annotated evidence or the rule-derived set (Appendix B.3). To compare its outcomes with human judgment, three human annotators judged a stratified sample of Base and Hard outputs without seeing the verifier outcome. The results: annotators judge grounding-only rejections invalid (insufficiently grounded or wrong) three times as often as accepted outputs (31.8% against 10.1%), and broadly valid in 45.7% of cases; crediting these rejections at each scenario's broadly-valid rate leaves a gap of -16.1 points, and also debiting accepted outputs judged invalid leaves -21.3.

What the calibration replaces. The calibration substitutes human judgment for the grounding check, the check that Hard adds: it credits Hard rejections whose sole failed check is grounding, and in its two-sided form debits accepted Hard outputs judged invalid. Decision checks, which Base and Hard share, and Base acceptance are scored by the verifier. Crediting Base grounding-only rejections the same way would raise Base acceptance and widen the gap; verifier-scored Base gives the narrower gap.

Sample and rubric. Annotators labeled 528 outputs, stratified by scenario–operator pair and verifier outcome (seed 20260918): in each pair where the outcome occurs, up to 25 grounding-only rejections, 6 grounding-and-decision rejections, 6 decision-only rejections, 6 accepted Hard outputs, and 4 Base grounding failures. We report the 461 from primary-cell records; the other 67 come from non-primary runs of the same cells. On all 528 labeled outputs grounding-only rejections are broadly valid in 42.2% of cases and the calibrated gap is -17.5 ($[-18.8, -16.1]$; -14.7 counting serialization repairs). Each case shows the request with its sources and the model output. The annotators are three members of the authors' research group who are not authors; they received the rubric, the displayed request, the sources, and the output, but not the gold evidence, the verifier outcome, the sampling stratum, the system, or the backbone. Annotators assign one primary category, in priority order: *ambiguous* (the displayed information is insufficient to judge), *wrong semantic decision*, *incomplete set or ranking*, *correct decision with insufficient grounding*, *serialization only* (semantics, grounding, and completeness are correct and only a meaning-preserving serialization repair is needed), and *broadly valid* (the output is semantically correct, sufficiently grounded, complete, and usable under the displayed request, even if it differs from a possible reference). *Broadly valid* in Table 27 is the share whose majority category is broadly valid; + *serialization-only* also counts serialization only.

Human judgment of sampled outputs. Each annotator labeled every sampled case under this rubric, recording decision, grounding, completeness, and surface validity before the primary category. The gold labels and evidence, verifier outcome, sampling stratum, system, and backbone were withheld, and annotators worked independently. Labels are aggregated by majority (at least two of three); no case had three different categories. Agreement on the six primary categories is high (Fleiss $\kappa = 0.94$ on the 461 cases; all three annotators agree on 429; pairwise Cohen κ 0.93–0.94). Intervals

come from a stratified task-cluster bootstrap (2,000 replicates). Cases with an ambiguous majority (14, 2, 12, 12, and 1 by stratum in Table 27) stay in the denominators.

Table 27: Human judgment of sampled outputs by verifier stratum (%; majority of three annotators); rejections and accepted outputs are Hard outputs; brackets: 95% CI.

Stratum	n	Broadly valid	+ serialization-only
Grounding-only rejections	245	45.7 [39.8, 51.3]	62.4 [56.7, 67.9]
Grounding-and-decision rejections	45	28.9 [17.8, 40.0]	48.9 [37.8, 60.0]
Decision-only rejections	74	21.6 [13.5, 29.7]	35.1 [27.0, 43.2]
Accepted	89	43.8 [33.7, 52.8]	76.4 [69.7, 82.0]
Base grounding failures	8	12.5 [0.0, 25.0]	62.5 [25.0, 87.5]

Reading. Annotators judge 31.8% of grounding-only rejections invalid (24.9% insufficiently grounded, 6.9% wrong), against 10.1% of accepted outputs and 48.6% of decision-only rejections; broadly valid shares are 45.7%, 43.8%, and 21.6% (28.9% for grounding-and-decision rejections), and 62.4%, 76.4%, and 35.1% counting outputs that need only a serialization repair. Of the 89 accepted outputs, 39 are broadly valid, 29 need only a serialization repair, which the verifier tolerates, 12 have an ambiguous majority, and 9 (10%) are judged invalid: insufficient grounding (6) or a wrong decision (3).

Population weighting and coverage. Weighting each scenario–operator pair by its number of recorded outputs in that stratum gives 50.6% broadly valid for grounding-only rejections (95% interval 43.6–57.7; 63.4% + serialization-only), 41.7% (77.1%) for accepted outputs, and 22.6% (32.3%) for decision-only rejections. The other majority categories of the 245 grounding-only cases are serialization only (16.7%) and ambiguous (5.7%). By scenario, grounding-only rejections judged broadly valid, broadly valid or serialization only, and insufficiently grounded are 43, 57, and 11 of 75 for SciFact; 24, 38, and 5 of 50 for ContractNLI; 33, 37, and 29 of 70 for WDC Products; and 12, 21, and 16 of 50 for SWE-bench Verified. Crediting every Hard rejection whose only failed check is grounding at its scenario's broadly-valid rate raises Hard acceptance by 9.7 points ($[8.4, 11.2]$), and the gap becomes -16.1 points ($[-17.6, -14.5]$; -13.6 counting serialization-only repairs), with verifier-scored Base acceptance; estimating the rates from the 229 of the 245 grounding-only cases that grounding-relaxed scoring of their records also accepts gives -16.1 too. By scenario, the calibrated gap is -25.7 (SWE-bench Verified), -6.4 (SciFact), -7.3 (ContractNLI), and -24.2 points (WDC Products); by operator family -13.4 (FILTER), -18.7 (JOIN), -20.1 (MAP), and -12.4 (RANK); by system -15.4 to -16.6 and by backbone -15.8 to -16.7, every interval excluding zero. Without the SWE-bench MAP cells, the strict, grounding-relaxed, and calibrated gaps are -23.8, -4.4, and -13.8 points. Also debiting accepted Hard outputs at each scenario's judged-invalid rate (4.2–4.3% outside WDC Products, 33.3% in WDC Products) gives -21.3 ($[-24.6, -18.3]$). Debiting only lowers Hard acceptance, so under the two-sided calibration every scenario, operator family, system, and backbone loses at least as much as under the one-sided values above.

External check. To check the in-group labels, three annotators outside the authors' research group, who had not seen the paper and were not told its hypotheses, labeled 50 cases under the same rubric and blinding: 40 drawn from the 461 analyzed cases (24 grounding-only rejections, including 12 from WDC Products; 6 accepted outputs; 5 decision-only and 5 grounding-and-decision rejections) and 10 control cases built from held-out Hard tasks, 5 with the gold grounding and 5 with the gold decision but wrong grounding. Their majority agrees with the in-group majority on 37 of 40 cases for the six-way category (Cohen's $\kappa = 0.895$) and on 37 of 40 for the binary broadly-valid judgment ($\kappa = 0.849$). Agreement among the external annotators on the six-way category is Fleiss $\kappa = 0.861$ on the 40 overlap cases. Each external annotator labels all 10 controls as expected. On the 24 grounding-only cases, the external majority judges 54.2% broadly valid, against 58.3% for the in-group majority on the same cases; on the 12 WDC Products grounding-only cases, both judge 58.3% broadly valid.

G CHECKS OF THE GAP

This appendix supports Observation 5 (Figure 2), the ablation of Observation 6 (Figure 3), the partial hardening of Observation 8, and Observation 10; Figure 9 plots the checks. Appendix G.1 reanalyzes the records of the primary cells, and Appendices G.2–G.6 report separate runs that compare conditions on the same tasks, with intervals from paired bootstraps over tasks or anchors (10,000 replicates); they use Qwen3 (Appendix D), except the SciFact MAP candidate-density check and the local-file reruns (GPT-5.4-mini), the additional-system runs (Claude Sonnet 4.6, with a Claude Haiku 4.5 proxy), and the frontier check (four backbones); the HotpotQA and frontier intervals use the stratified record bootstrap of the main results (2,000 replicates). The grounding-relaxed gap is in Table 4, and the human-calibrated gap in Appendix F.

G.1 REANALYSES OF THE PRIMARY CELLS

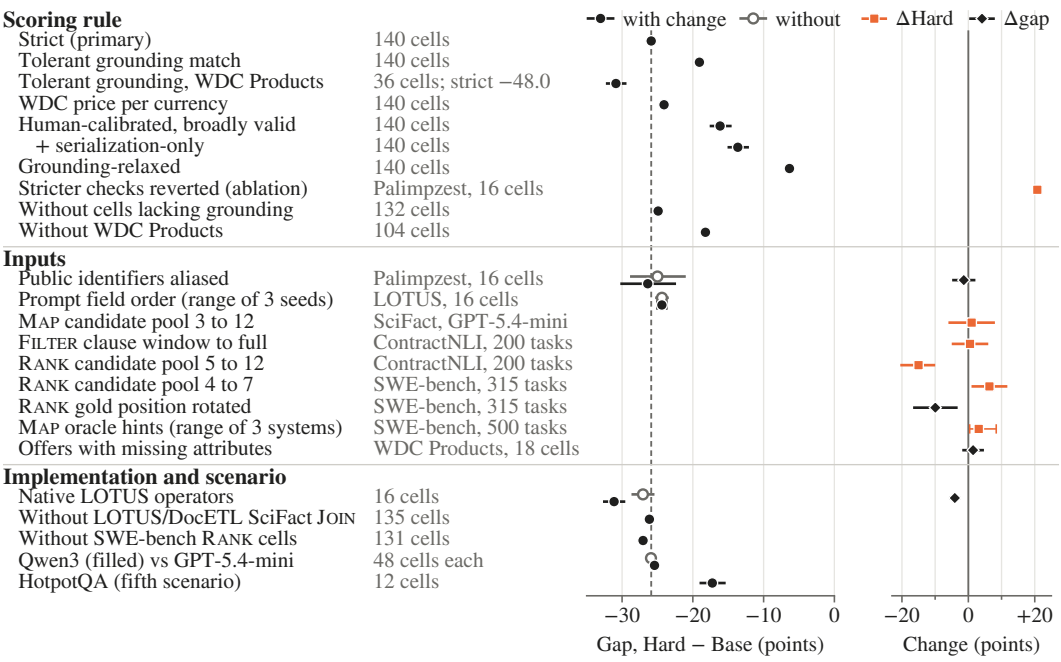


Figure 9: All checks of the gap (points; bars: 95% CI, or min–max where labeled a range): left, the gap with each change (filled) and on the same cells without it (open), dashed at the primary gap; right, the paired change in Hard acceptance (squares) or in the gap (diamonds).

Across reanalyses of the primary cells the gap stays between -23.8 and -27.0 points; the tolerant rule gives -19.1, and leaving out one scenario -18.2 to -30.2. Table 28 collects the reanalyses, and the paragraphs below describe its rows.

Task-cluster bootstrap. Because the same tasks recur across systems and backbones, this bootstrap resamples tasks within each scenario–operator pair and applies the same draw to every cell and condition. For the pooled rows, its intervals differ from the record-level ones by at most 0.4 points at either end (at most 0.8 for single-scenario gaps): grounding-relaxed gap $[-6.9, -5.8]$, WDC Products grounding-relaxed gap $[+1.2, +3.9]$, three-call recovery $[3.2, 4.7]$, and grounding-relaxed tool change over the tool-run cells $[-1.3, -0.2]$. No interval changes whether it excludes zero.

Infrastructure records and backbones. Excluding the infrastructure records (Appendix B.8) gives a gap of -26.2 points. Keeping only GPT-5.4-mini and Qwen3, whose cells cover their full task inventories, gives -25.7 points.

Local files. The GPT-5.4-mini execution runs call `codex exec` in the benchmark working directory with a read-only sandbox (Appendix D). To measure the effect of file access, we reran

Table 28: Reanalyses of the primary cells: the gap, Hard−Base (points; brackets: 95% CI).

Analysis	Cells	Gap
Primary analysis (all observed records)	140	−25.9 [−26.4, −25.3]
Task-cluster bootstrap	140	−25.9 [−26.6, −25.1]
Infrastructure errors excluded	140	−26.2 [−26.7, −25.6]
GPT-5.4-mini and Qwen3 only	96	−25.7 [−26.0, −25.3]
Local-file outputs removed (GPT-5.4-mini, 735 records)	140	−26.0 [−26.5, −25.4]
Local-file outputs, tools disabled	140	−26.0
Paired Base–Hard records (71,592 pairs)	96	−26.1 [−26.5, −25.7]
Tolerant grounding	140	−19.1 [−19.6, −18.5]
Cells without a grounding field removed	132	−24.9 [−25.5, −24.3]
One scenario left out	104–108	−30.2 to −18.2
Without WDC Products	104	−18.2 [−18.8, −17.7]
Without SWE-bench MAP cells	131	−23.8 [−24.4, −23.3]
WDC Products prices compared per currency	140	−24.1 [−24.6, −23.5]
Without SWE-bench RANK cells	131	−27.0 [−27.6, −26.5]
SWE-bench RANK Hard with cited evidence required	140	−26.3 [−26.9, −25.8]
Without LOTUS and DocETL SciFact JOIN cells	135	−26.1 [−26.7, −25.6]
Seven JOIN cells rerun with the grounding field requested	140	−25.5 [−26.0, −24.9]

GPT-5.4-mini with file access disabled on all 735 Base and Hard records whose output mentions inspecting local files and, to cover reads that leave no trace in the output, on 735 random other records of the same cells and conditions and on 386 random Palimpzest and Hard+tool records. Removing the 735 records gives a gap of -26.0 points, and using their tool-free outputs also gives -26.0. In the random samples, acceptance changes by $+2.2$ points for other LOTUS and DocETL Hard records, 0.0 and -6.4 for Palimpzest Base and Hard, and -4.7 for Hard+tool records, none significantly (exact McNemar, $p \geq 0.10$). Qwen3, which receives only its prompt (Appendix D), has a gap of -25.4 points, against -25.9 for GPT-5.4-mini (Table 22).

Paired records. To check the gap on paired tasks, we paired Base and Hard records by anchor and construction role within each of the 96 cells of GPT-5.4-mini and Qwen3 (71,592 pairs): 37,631 pairs are accepted in both conditions, 9,959 in neither, 21,333 only in Base, and 2,669 only in Hard. The pair-weighted gap is -26.1 points ($[-26.5, -25.7]$; pair-level multinomial bootstrap), against the cell-mean gap of -25.7 for the same backbones, which weights each cell equally.

Cells without a grounding field. To check the eight cells whose Hard outputs carry no grounding field (Appendix B.7), we removed them: the gap is -24.9 points and the grounding-relaxed gap -6.7.

Tolerant grounding. To check how much the exact-set rule adds to the gap, we rescored the stored outputs with a tolerant rule and kept every other check. A ContractNLI span set passes if its F1 with an annotated variant is at least 0.5 or it contains the variant. SciFact evidence passes if it contains an annotated rationale and at most two further sentences, or covers a rationale and cites only sentences of the task's rationales. A WDC Products attribute set passes if it is a non-empty subset of the gold set, so every cited attribute is a signal of the stated decision; SWE-bench checks are unchanged. The gap is then -19.1 points ($[-19.6, -18.5]$; task-cluster bootstrap $[-19.8, -18.3]$): -6.0 for SciFact, -10.6 for ContractNLI, -30.9 for WDC Products, and -27.3 for SWE-bench Verified.

Prices and cell groups. To check the price rule, we rebuilt the WDC Products gold sets with prices in different currencies treated as neither matched nor conflicting: the WDC Products gap is -41.0 points, the primary gap -24.1, and the grounding-relaxed gap is unchanged. To check the SWE-bench RANK cells, whose loader lists the gold patch first, we removed them (-27.0) and rescored their Hard outputs with the cited evidence required (-26.3); rotating the gold patch over all positions widens their gap (Appendix G.4). Removing the LOTUS and DocETL SciFact JOIN cells gives -26.1 and a grounding-relaxed gap of -6.6.

Scenario mix. Leaving out one scenario at a time gives gaps of -25.4 (without SWE-bench Verified), -29.6 (SciFact), -30.2 (ContractNLI), and -18.2 points (WDC Products), and grounding-relaxed gaps of -1.4 to -9.4 points; every task-cluster interval excludes zero.

G.2 Component Ablation

To check which Hard component carries the gap, each arm reverts one component of the Hard tasks of Qwen3 $\times$ Palimpzest and keeps the others: the explicit constraints, the added context, the grounding fields of the output schema, or the stricter checks. The run covers 16 cells with 300 tasks per cell, balanced by gold label where the cell has one, or all 279 SciFact RANK tasks; every arm uses the same tasks, and an output whose evidence field alone is unparsable is scored on the decision in its raw output. Figure 3 reports each arm against a Hard run on the same tasks; on these cells the gap of the main runs is -24.7 points.

Stricter checks are the dominant component: scoring the same Hard outputs by their decision alone raises acceptance by 20.8 points (95% CI $[19.8, 21.8]$). The gain is 44–49 points in the WDC Products cells, 38.3 in SciFact MAP, 21–26 in SciFact FILTER and JOIN and ContractNLI FILTER and MAP, and 9.7 in SWE-bench JOIN, whose Hard contract checks the decision basis; the other cells change by at most one point, and SWE-bench MAP, whose output is a repair specification, uses its Hard verifier in every arm. Removing the explicit constraints changes acceptance by -5.5 points ($[-6.8, -4.3]$), most in ContractNLI FILTER (-21.3) and MAP (-22.3), whose constraints state the span requirement, and in WDC Products RANK (-12.0). Removing the added context raises acceptance by 4.0 points over 11 cells ($[2.5, 5.6]$), mostly from SWE-bench MAP: restoring its Base code snippets and full failing-test list gives $+37.3$ points, and the other ten cells change by -4.7 to $+5.3$. Removing the grounding fields from the output schema raises decision accuracy by 2.1 points over 10 cells ($[+0.7, +3.5]$): by $+3.3$ to $+10.7$ in WDC Products, by -3.7 to $+4.3$ in SciFact and ContractNLI ($+1.3$ in SciFact MAP), and by -4.0 and -1.0 in SWE-bench JOIN and RANK.

Requesting versus checking grounding. Two arms of the ablation form a 2×2 design on the ten cells whose Hard contract requests grounding (Table 29): requesting the grounding fields lowers decision accuracy by 2.1 points ($[0.7, 3.5]$), whereas checking them lowers acceptance by 30.5 points on the same Hard outputs (the 16-cell mean in Figure 3, which includes cells without grounding checks, is 20.8). Checking grounding without requesting it is not defined.

Table 29: Requesting versus checking grounding on the ten ablation cells whose Hard contract requests grounding (Qwen3 $\times$ Palimpzest, 300 tasks per cell; acceptance, %). Decision only: the Hard outputs scored by their decision.

	Grounding checked	**Decision only**
Grounding fields requested	42.6	73.1
Grounding fields not requested	not defined	75.2

G.3 Inputs and Implementations

Identifier obfuscation. To check whether public identifiers drive the gap, we ran Qwen3 $\times$ Palimpzest on 16 cells with 50 label-stratified tasks per cell and condition, once with public identifiers and once with synthetic aliases that are mapped back before scoring. The aliases replace SWE-bench repository names, instance IDs, and commit hashes, paper titles, document IDs, and offer IDs; SWE-bench file paths are kept, and ContractNLI rows carry no such identifier. The gap is -25.0 points ($[-28.9, -21.0]$) with public identifiers and -26.4 points ($[-30.2, -22.4]$) with aliases, a difference of -1.4 points ($[-4.9, +2.1]$), or -1.8 points ($[-5.8, +2.2]$) without ContractNLI. Aliasing changes Base acceptance by $+0.3$ points and Hard acceptance by -1.1 points.

Native LOTUS operators. To check that the adapters do not create the gap, we ran all 16 scenario–operator pairs with native LOTUS operators and, on the same tasks, with a fresh adapter run (Qwen3, 300 label-balanced tasks per cell and condition). Two-label decisions use `sem_filter` (`sem_join` for WDC Products JOIN), three-way labels and MAP outputs `sem_extract`, and RANK `sem_topk`; in Hard, an added `sem_extract` returns grounding outside the decision

operator's output. The unchanged verifiers score every output. The native gap is -31.1 points against -27.0 for the adapter over the 16 cells (difference -4.1, $[-5.8, -2.5]$) and -36.6 against -35.7 over the ten grounding-checked cells (-0.9, $[-2.9, +1.2]$; Table 30). The difference changes sign across cells; in WDC Products MAP the adapter's Base outputs write `non-match`, and the SWE-bench FILTER adapter gap varies with the prompt's row-field order (both arms use the same hash seed; Appendix G.3). In SciFact, native decision accuracy barely changes between Base and Hard, so the native loss there is a grounding loss.

Table 30: Native LOTUS operators against the adapter on the same tasks (Qwen3, 300 tasks per cell and condition): strict gap, Hard−Base (points). Grounding: whether the Hard contract checks grounding.

Scenario	Op.	Grounding	Native	Adapter	Native−adapter
SWE-bench Verified	FILTER	no	−0.7	+16.3	−17.0
	JOIN	no	−65.3	−36.3	−29.0
	RANK	no	−52.3	−34.3	−18.0
	MAP	yes	−52.3	−58.3	+6.0
SciFact	FILTER	yes	−29.0	−29.7	+0.7
	JOIN	yes	−24.7	−37.3	+12.7
	RANK	no	−1.1	−1.8	+0.7
	MAP	yes	−37.3	−47.7	+10.3
ContractNLI	FILTER	yes	−27.0	−24.0	−3.0
	JOIN	no	−6.0	−0.7	−5.3
	RANK	no	−6.7	−18.7	+12.0
	MAP	yes	−26.3	−14.0	−12.3
WDC Products	FILTER	yes	−45.3	−34.0	−11.3
	JOIN	yes	−50.7	−53.0	+2.3
	RANK	yes	−32.0	−54.7	+22.7
	MAP	yes	−41.3	−4.7	−36.7
All 16 cells			−31.1	−27.0	−4.1
Ten grounding-checked cells			−36.6	−35.7	−0.9

Prompt field order. To check sensitivity to prompt field order, the 16 LOTUS × Qwen3 cells ran under three hash seeds that permute the row fields of the LOTUS prompt, on 200 tasks per cell shared by Base and Hard. The mean gap is -25.1, -24.4, and -23.7 points under the three seeds, against -24.4 in the main runs of these cells (Table 24), and mean Base acceptance varies by 0.3 points.

G.4 CANDIDATE POOLS AND OFFER FIELDS

SciFact MAP candidate density. To check that the SciFact MAP result is not tied to its 12-abstract pool, GPT-5.4-mini ran the Hard contract on the first 100 SciFact claims with support or contradict evidence, growing nested candidate pools one abstract at a time from 3 candidates, the Base pool size, to 12. The gold abstracts stayed in positions 1–3 and added candidates were appended after them; the task, verifier, and order were otherwise fixed, and a direct adapter rendered the LOTUS MAP prompt. Acceptance stayed between 49 and 53 of 100 at every density (50 at 3 candidates, 51 at 12; paired difference $+1$ claim, 95% bootstrap interval $[-6, +8]$), and no density differed from the 3-candidate setting (exact McNemar with Holm correction, all adjusted $p = 1.0$). The decision was correct for 79–84 claims at each density and the evidence set for 49–53; nearly all decision errors were no-evidence answers to evidence-bearing claims, so rejections came mainly from locating the evidence. On the same claims the evaluated Hard run, which places the 12 candidates in shuffled order, accepts 55 of 100.

RANK candidate density. To check how much of the RANK gap comes from pool size, LOTUS × Qwen3 held the Hard contract and the Hard negatives fixed and grew nested pools from the Base to the Hard pool size, with the gold candidates rotated over the first positions of the Base pool size. On 200 ContractNLI tasks, acceptance is 93.5, 95.0, 91.0, 91.0, 88.5, and 78.5% with 5, 6, 7, 8, 10, and 12 candidates (-15.0 points from 5 to 12, $[-20.5, -10.0]$); in the same run, the Base and

Hard tasks in loader order accept 98.0% and 81.0%, so the added Hard candidates carry 15.0 of this 17.0-point drop. On the 315 SWE-bench tasks, acceptance is 51.7, 57.1, 57.5, and 58.1% with 4 to 7 candidates ($+6.3$ points, $[+1.0, +11.7]$); with four candidates the Hard task, whose negatives are false-fix variants of the gold patch, is already well below the Base task, so pool size does not explain the SWE-bench RANK drop. At every pool size, acceptance equals top-1 accuracy, so no ranking was rejected as incomplete. Hard SWE-bench RANK keeps the 315 issues with at least four strong distractors; restricting Base to the same issues changes the SWE-bench RANK gap from -9.1 to -9.0 points and leaves the SWE-bench and primary gaps at -27.3 and -25.9.

RANK gold position. To check candidate position, we ran the SWE-bench Base and Hard RANK tasks once in loader order, which lists the gold patch first, and once with the gold patch rotated over all positions; rotation changes the gap by -10.0 points ($[-16.6, -3.2]$), so loader order makes the SWE-bench RANK gap smaller.

ContractNLI FILTER pool. ContractNLI FILTER has the largest pool change: Base shows a window of at least three clauses around the evidence, and Hard the full contract. To check how much of its gap the pool carries, LOTUS $\times$ Qwen3 ran 200 label-balanced Hard tasks with the Hard contract fixed and the clause pool cut to windows around the evidence (the Base window rule with 3, 10, 20, or 40 clauses, plus the five highlighted distractors; median 7, 13, 21, and 40 clauses) and with the full contract (median 85 on these tasks). Strict acceptance is 50.5, 52.5, 50.5, 48.5, and 51.0%, and the full contract differs from the smallest window by $+0.5$ points ($[-5.0, +6.0]$; exact McNemar with Holm correction, all adjusted $p = 1$). Decision accuracy is 63.5–68.5% at every size. On the same tasks, the main-run Base records accept 72.0%, so pool size does not explain the ContractNLI FILTER gap.

WDC Products offer fields. To check whether offers without a brand, price, or description drive the WDC Products gap, we split its 36 cells by whether a task shows such an offer (93% of tasks do). The strict gap is -49.2 points for complete tasks (18 cells with at least 20 Base and 20 Hard records in the stratum) and -47.4 for incomplete tasks (36 cells); on the 18 cells with both, the incomplete tasks differ by $+1.4$ points ($[-1.8, +4.7]$). The grounding-relaxed gaps are $+5.9$ and $+2.5$.

G.5 OTHER SCENARIOS, SYSTEMS, AND BACKBONES

Fifth scenario: HotpotQA. To check that the gap is not specific to the four scenarios, we built the same four operator families on the HotpotQA distractor setting (Yang et al., 2018): multi-hop questions over Wikipedia paragraphs, with the dataset's own distractors and sentence-level supporting facts. FILTER and JOIN judge a question against one paragraph, RANK selects a gold paragraph, and MAP returns the answer. Base shows two gold and two lexically close paragraphs and checks the label or answer; Hard shows the two gold and the eight HotpotQA distractors, adds explicit constraints and an output schema, and requires the cited sentences to equal the supporting facts, so Hard MAP acceptance is HotpotQA's joint exact match. Distractors and pools come from the dataset and fixed rules, and gold outputs pass every verifier. Qwen3 ran 300 tasks per cell and condition, balanced by label or question type, on the three systems (12 cells). The gap is -17.2 points ($[-19.1, -15.4]$) under strict acceptance and -2.7 ($[-4.5, -1.0]$) under grounding-relaxed acceptance; the MAP gap is -37.7 points strict and -2.2 grounding-relaxed, and the RANK gap, which checks no grounding, is $+3.2$.

Additional systems. To check that the gap is not specific to the three adapters, adapter-level runs evaluate additional operator implementations with Claude Sonnet 4.6 (Table 31). The FILTER runs use 100 Base and 100 Hard tasks per scenario, and each rate is the mean of four per-scenario rates. Their LOTUS, DocETL, and Palimpzest rows reuse the main-run Claude Sonnet 4.6 outputs on these tasks; the SciFact DocETL rates come from a run of the DocETL adapter prompt on them. Each row's gap compares Base and Hard under the same route and settings. The BARGAIN-inspired row calibrates a Claude Haiku 4.5 proxy against the LOTUS Claude Sonnet 4.6 outputs as oracle (Zeighami et al., 2025), and the CSV-inspired row labels about a quarter of each TF–IDF cluster with the oracle outputs and assigns the rest by similarity voting (Hou et al., 2026). Nirvana (Zhu et al., 2026) uses its released `MapOperation` for structured tasks and `JoinOperation` for WDC Products JOIN; its separate four-operator run uses 25 tasks per scenario–operator–condition (400 per condition). Computed from

unrounded rates, the FILTER drops are 18.5–29.0 points; over its four operators, Nirvana loses 28.2 points (Table 31b).

Table 31: Adapter-level runs with Claude Sonnet 4.6 (acceptance, %). (a) Shared FILTER contract; (b) Nirvana on the four operators, a separate sample.

(a) Shared FILTER contract

System	Base	Hard
LOTUS	84.2	57.5
DocETL	88.5	62.5
Palimpzest	90.0	66.0
BARGAIN-inspired	79.8	57.2
CSV-inspired	53.8	35.2
Nirvana	90.5	61.5

(b) Nirvana contracts

Operator	Base	Hard
FILTER	93.0	62.0
JOIN	92.0	62.0
MAP	83.0	50.0
RANK	98.0	79.0
All	**91.5**	**63.3**

Frontier backbone. To check whether a frontier backbone closes the gap, Claude Opus 5.5 (`claude-opus-5-5`, through `claude -p` with tools disabled, as for Claude Sonnet 4.6) ran the 16 Palimpzest cells in both conditions on 30 label-balanced tasks per cell and condition, the same task keys for every backbone in Table 32. Table 32 reports the recorded verdicts. With `non-match` normalized to `non_match` (16 of its 30 Base WDC Products MAP outputs), Claude Opus 5.5 has a strict gap of -24.2 ($[-28.3, -20.0]$) and a grounding-relaxed gap of -2.1 ($[-5.2, +0.8]$). Of its 148 Hard rejections, 129 fail a grounding check.

Table 32: Frontier backbone on the 16 Palimpzest cells, 30 tasks per cell and condition with the same task keys for every row; rows other than Claude Opus 5.5 are the primary records of these keys. Strict and grounding-relaxed acceptance (%; brackets: 95% CI).

	Strict			Grounding-relaxed		
Backbone	Base	Hard	Δ	Base	Hard	Δ
Claude Opus 5.5	90.6	69.2	−21.5 [−25.6, −17.1]	91.7	92.3	+0.6 [−2.5, +3.8]
Claude Sonnet 4.6	88.3	59.6	−28.7 [−33.1, −24.2]	89.4	83.5	−5.8 [−9.8, −1.9]
GPT-5.4-mini	90.0	60.4	−29.6 [−34.4, −24.8]	91.0	81.5	−9.6 [−14.0, −5.2]
Qwen3	82.3	53.1	−29.2 [−33.8, −24.4]	83.3	76.7	−6.7 [−11.2, −2.1]

G.6 SWE-BENCH MAP CONTEXT AND CHAINS

Oracle file hints. To check how much of the SWE-bench Hard MAP drop the oracle file hints explain, we added the Base hints to all 500 Hard MAP tasks, kept the Hard constraints, caps, and verifier, and ran Qwen3 on each system, paired with a Hard run on the same tasks. The hints alone change acceptance by $+8.4$ points in LOTUS ($[+5.4, +11.6]$), $+0.4$ in Palimpzest ($[-3.4, +4.4]$), and $+0.6$ in DocETL ($[-2.4, +3.6]$). Adding the Base code snippets and full failing-test list as well raises acceptance by 48.4, 44.0, and 23.8 points, 43–62% of the gap between the Base and Hard runs of this check. The Hard constraints, caps, and verifier account for the rest.

Partial hardening. To check which step of a chain carries the loss, LOTUS × Qwen3 ran every mix of Base and Hard steps on all 100 anchors per scenario: FILTER→JOIN chains in the four scenarios, the four-step SciFact chain, and the three-step SWE-bench chain (4,000 chains). A step's effect is the paired change in all-step acceptance when that step alone of the all-Base chain is made Hard. In the two-step chains, a Hard FILTER changes acceptance by -39 points in SciFact ($[-49, -29]$), -15 in WDC Products ($[-25, -5]$), -7 in SWE-bench ($[-18, +4]$), and -8 in ContractNLI ($[-17, +1]$); a Hard JOIN changes it by -35 ($[-45, -25]$), -1 ($[-7, +5]$), -20 ($[-28, -12]$), and $+9$ points ($[+4, +15]$), the last for the ContractNLI JOIN, which checks the label only. In the four-step SciFact chain, the effects are -36 (FILTER), -35 (JOIN), 0 (RANK), and -39 points (MAP), and in the three-step SWE-bench chain -14 (FILTER, $[-25, -3]$), -24 (JOIN, $[-33, -16]$), and 0 points (RANK).

H Failure Analysis and Representative Cases

This appendix supports Observation 7. It applies every verifier check to each rejected output and splits Hard failures into grounding only, decision only, both, and neither (Table 33); pooled, 55.7% of Hard failures fail grounding without the decision. Checks map to the execution types of Table 2, and Table 36 shows one failure for each of four checks, with both kinds of grounding.

Cohorts. The failure analysis uses observed, non-infrastructure rejected records, pooled over records. The Hard cohort covers the 140 primary cells (34,665 failures). The Hard+tool cohort covers the tool-run cells (23,568 failures) and is compared with Hard restricted to the same cells (27,926 failures). Pooling weights backbones by their record counts; Table 35 gives each cohort's composition (Claude Sonnet 4.6: 9.8% of Hard and 5.0% of Hard+tool failures). The Tool protocol type, used only in Hard+tool, marks a violated tool-call protocol; Appendix I compares Hard and Hard+tool failures.

Full-check verification. Each deterministic verifier returns at its first failing check, which the recorded rejection reason names. For the failure analysis, we apply every check of the corresponding verifier to every rejected output and record a vector of pass, fail, or not-applicable outcomes (a check is not applicable when the contract does not require it for that output, for example the evidence check on a no-evidence task). The full check reproduces the recorded first failing check for 99.1% of Hard, 98.4% of Base, and 99.8% of Hard+tool failures. Every record keeps its recorded verdict; the vectors only annotate failures.

Table 33: Verifier-check flags among Hard failures by system and operator (%). Grounding and Decision overlap; G only, Both, D only, and Neither (only other checks fail) are disjoint.

System	Op.	Failures	Grounding	Decision	G only	Both	D only	Neither
LOTUS	FILTER	4,068	79.0	43.5	55.7	23.4	20.1	0.8
	JOIN	3,626	64.7	59.7	40.0	24.7	35.0	0.3
	MAP	4,187	92.9	25.5	71.3	21.6	4.0	3.1
	RANK	706	48.0	46.3	42.9	5.1	41.2	10.8
Palimpzest	FILTER	3,327	76.1	44.3	54.4	21.7	22.6	1.3
	JOIN	3,123	67.1	52.0	45.5	21.6	30.4	2.6
	MAP	3,938	91.2	31.4	65.3	25.9	5.5	3.3
	RANK	899	31.3	40.6	26.8	4.4	36.2	32.6
DocETL	FILTER	3,319	83.0	43.6	56.4	26.6	17.0	0.0
	JOIN	3,098	80.8	49.5	50.5	30.3	19.2	0.0
	MAP	3,648	93.0	25.0	71.3	21.8	3.2	3.8
	RANK	726	34.3	43.4	30.4	3.9	39.5	26.2
All		34,665	78.4	41.1	55.7	22.7	18.3	3.3

Table 34: Failure types (%, overlapping); Hard (same cells): Hard on the tool-run cells.

Type	Hard	Hard (same cells)	Hard+tool
Grounding	78.4	75.1	72.8
Decision	41.1	47.1	60.8
Completeness	6.0	7.3	6.6
Other fields	3.8	4.8	1.4
Format	2.2	2.3	0.9
Tool protocol	—	—	1.3
Failures (denominator)	34,665	27,926	23,568

Table 35: Composition of the failure cohorts (% of failures).

Share (%)	Hard	Hard (same cells)	Hard+tool
GPT-5.4-mini	40.5	28.2	28.9
Claude Sonnet 4.6	9.8	10.0	5.0
Qwen3	49.8	61.8	66.2
LOTUS	36.3	35.4	33.6
Palimpzest	32.6	33.5	33.1
DocETL	31.1	31.2	33.3
SWE-bench Verified	14.9	18.5	15.1
SciFact	14.0	15.2	14.4
ContractNLI	28.8	35.7	36.1
WDC Products	42.4	30.6	34.3
Failures	34,665	27,926	23,568

Table 36: Representative Hard failures, one per failed check.

Type	**Task**	**Expected**	**Rejected output**	**Verifier reason**
Format	ContractNLI JOIN (LOTUS × GPT-5.4-mini)	One JSON object with the label.	Two concatenated JSON objects.	Parse error.
Completeness	SciFact RANK (DocETL × GPT-5.4-mini, claim 105)	Permutation of all candidates.	Returns 11 ids for a 12-candidate ranking.	Missing candidate.
Grounding (evidence)	SciFact FILTER (LOTUS × Qwen3, claim 1006)	Support, citing the sentence that a C4C4 motif in RTEL1 mediates its interaction with TRF2.	Support, citing only the sentence that TRF2 recruits RTEL1 to telomeres.	Evidence is not an annotated rationale.
Grounding (attributes)	WDC FILTER (Palimpzest × GPT-5.4-mini)	Match, gold set {price, title}: titles share tokens, prices 262.17 and 258.32 (within 5%).	Match, citing {brand, title}, although one offer has no brand.	Attribute set mismatch.
Decision	WDC MAP	Match.	Returns non-match and sets the same-cluster flag to false.	Decision mismatch.

I Recovery Diagnostics

This appendix supports Observation 9. On the 115 cells whose operators may make up to three tool calls, Hard+tool runs raise Hard acceptance by 4.0 points ($[3.1, 4.9]$; Table 37), and Base−Hard+tool leaves 21.2 points. The nine SWE-bench MAP cells contribute 3.0 of the 4.0 points: there the repair-specification helper proposes the edited files, and the helper alone, with no model call and the test chosen by the token-overlap ranker (Appendix B.9), is accepted on 402 of the 500 Hard MAP tasks (80.4%, Wilson interval $[76.7, 83.6]$). The helper reads only the task payload, whose Hard MAP context shows one reduced snippet of a gold file and tags its source (Table 7). On the other 106 cells, the change is $+1.0$ point ($[0.1, 1.9]$) at 2.24 times the API cost of the Hard runs (Appendix D); on the 22 MAP cells outside SWE-bench it is -1.5 ($[-3.6, +0.6]$). The nine SWE-bench RANK cells, whose operators may make eight calls, are reported separately (Appendix I.1); over all 124 tool-run cells the change is $+2.7$ ($[1.8, 3.5]$).

Table 37: Tool recovery on the 115 three-call cells by operator, scenario, backbone, and system (%), with the nine eight-call SWE-bench RANK cells and all 124 tool-run cells below. Recovered: Hard+tool − Hard; Residual: Base − Hard+tool (points); brackets: 95% CI.

	Cells	Base ↑	Hard ↑	Hard+tool ↑	Recovered	Residual
FILTER	31	84.3	63.4	62.3	−1.1 [−2.6, +0.4]	22.0 [20.6, 23.4]
JOIN	31	84.9	59.9	63.3	+3.5 [+2.0, +5.0]	21.6 [20.2, 22.9]
MAP	31	76.7	43.3	53.5	+10.3 [+8.6, +11.9]	23.2 [21.7, 24.8]
RANK	22	96.0	76.3	79.5	+3.1 [+0.9, +5.3]	16.6 [14.8, 18.3]
SWE-bench Verified	27	94.0	60.6	73.4	+12.8 [+11.2, +14.3]	20.6 [19.4, 21.8]
SciFact	28	92.5	78.8	77.2	−1.6 [−3.1, −0.0]	15.3 [13.8, 16.7]
ContractNLI	36	76.8	63.4	64.7	+1.3 [−0.0, +2.6]	12.1 [10.9, 13.3]
WDC Products	24	76.8	29.8	34.4	+4.6 [+2.0, +7.2]	42.4 [40.2, 44.5]
GPT-5.4-mini	33	87.3	68.9	73.4	+4.5 [+3.8, +5.3]	13.9 [13.3, 14.6]
Claude Sonnet 4.6	37	88.6	58.8	59.6	+0.8 [−1.6, +3.4]	29.0 [26.9, 31.2]
Qwen3	45	79.5	53.2	59.4	+6.2 [+5.6, +6.9]	20.1 [19.5, 20.7]
LOTUS	37	83.9	58.7	63.7	+5.0 [+3.6, +6.3]	20.2 [19.0, 21.4]
Palimpzest	41	84.8	60.0	63.7	+3.7 [+2.0, +5.3]	21.1 [19.7, 22.6]
DocETL	37	85.3	59.7	63.1	+3.3 [+1.9, +4.8]	22.2 [21.0, 23.4]
Three-call cells	115	84.7	59.5	63.5	+4.0 [+3.1, +4.9]	21.2 [20.4, 22.0]
SWE-bench RANK, eight calls	9	99.6	90.5	76.0	−14.5 [−17.3, −11.6]	23.6 [20.9, 26.1]
All tool-run cells	124	85.7	61.7	64.4	+2.7 [+1.8, +3.5]	21.4 [20.6, 22.1]

I.1 Tool Budget and Relaxed Acceptance

Under grounding-relaxed acceptance, the tool change is $+0.3$ points on the 115 three-call cells ($[-0.3, +1.0]$) and -0.8 points over all tool-run cells ($[-1.4, -0.1]$), because the nine eight-call SWE-bench RANK cells score the ranking alone and keep their -14.5. On these nine cells, with the same tools, the tool change is $+6.9$ points for GPT-5.4-mini ($[+4.3, +9.4]$), -13.7 for Qwen3 ($[-17.1, -10.2]$), and -36.7 for Claude Sonnet 4.6 ($[-44.0, -29.3]$). With three calls instead of eight on the same 315 tasks, Qwen3 Hard+tool acceptance changes by -0.6 points ($[-3.4, +2.1]$, three systems), so the budget does not cause the Qwen3 decrease on SWE-bench RANK.

I.2 Failed Checks with Tools

On the tool-run cells, the decision check fails in 47.1% of Hard failures and 60.8% of Hard+tool failures (Table 34), and grounding still fails in 72.8% of Hard+tool failures, against 75.1% of Hard failures on the same cells. Table 38 splits the Hard+tool failures by system and operator.

I.3 Negative Tool-Recovery Audit

To check how much of the tool change comes from records that break the tool protocol, we scored the 297 Hard+tool records rejected for a malformed tool response or no tool call at the verdict of their

Table 38: Verifier-check flags among Hard+tool failures (%), as in Table 33; Table 34 gives Hard on the same cells.

System	Op.	Failures	Grounding	Decision	G only	Both	D only	Neither
LOTUS	FILTER	2,508	68.9	61.9	36.0	32.9	29.0	2.1
	JOIN	2,330	67.1	69.2	29.5	37.6	31.6	1.3
	MAP	2,524	90.3	47.4	47.4	42.9	4.5	5.2
	RANK	547	24.1	68.2	19.9	4.2	64.0	11.9
Palimpzest	FILTER	2,548	68.4	62.6	36.1	32.3	30.3	1.4
	JOIN	2,141	67.6	74.5	24.6	43.0	31.4	1.0
	MAP	2,564	92.9	47.5	49.6	43.2	4.3	2.8
	RANK	559	27.5	64.8	22.4	5.2	59.6	12.9
DocETL	FILTER	2,513	68.8	62.8	35.5	33.4	29.4	1.8
	JOIN	2,290	67.1	71.1	27.3	39.8	31.3	1.6
	MAP	2,502	92.6	49.0	48.4	44.3	4.7	2.7
	RANK	542	26.8	72.3	19.6	7.2	65.1	8.1
All		23,568	72.8	60.8	36.4	36.4	24.3	2.9

Hard record. Recovery over the tool-run cells then rises from $+2.7$ to $+4.3$ points ($[+3.6, +5.1]$), and Claude Sonnet 4.6, whose recorded tool change over its 40 tool-run cells is -2.0 points, gains 2.9 points ($[+0.8, +5.2]$); 169 of the 297 records are Claude Sonnet 4.6 outputs. Pairing the Hard and Hard+tool records of DocETL $\times$ Claude Sonnet 4.6 by task key gives 930 matched pairs: 123 regressions (accepted in Hard, rejected in Hard+tool) and 77 improvements. No regression records budget exhaustion, and one carries a record-level error. Table 39 groups the regressions by their recorded rejection reason; most concern the tool-call protocol or the cited grounding.

Table 39: Recorded rejection reasons of DocETL $\times$ Claude Sonnet 4.6 regressions (Hard accepted, Hard+tool rejected), grouped into four causes.

Cause	Count	Share (%)	Recorded reasons (count)
Tool/protocol drift	58	47.2	Malformed tool response (55), no tool call exercised (2), parse error (1).
Grounding check failed	55	44.7	Attribute set (33), tests addressed (13), attribute grounding (4), edit operations and tests (3), span variant (1), basis (1).
Decision drift	9	7.3	Label (9).
Constraint violation	1	0.8	Edited files exceed the allowed limit (1).

J LIMITATIONS

SemBench-Hard targets verifier-checkable outputs (labels, record sets, pairs, rankings, and schema-bound artifacts) over public text datasets prepared as relational views; free-form generation, other modalities, and private data are outside its scope. The translation axis measures how LLM translators write programs in one shared operator language, and the execution axis measures tasks of the four operator families, without aggregation, under fixed programs and task-local tools, with native operators and other systems covered by the checks of Appendices G.3 and G.5. Strict grounding checks score agreement with the datasets' annotated evidence and, for WDC Products, with rule-derived attribute sets (Appendix B.3); sufficiency is estimated from human judgments of a stratified sample (Appendix F), and grounding-relaxed and tolerant rates report the gap under looser agreement rules.

K PUBLIC ARTIFACT CONTENTS

The artifact entry point is `https://anonymous.4open.science/r/SEMBENCH_HARD/`. It contains the task loaders and verifiers, the Base and Hard prompt renderers, the adapter configurations and run manifests with pinned system versions, the execution, chain, and translation runners, the translation prompt packets and frozen anchors, the task keys of the subsampled cells and of the

local-file check, the runners of the checks of Appendix G, the scripts that aggregate run records into every table and figure, the human-validation rubric and aggregation scripts, and data-acquisition scripts with checksums of the evaluated source snapshots. The de-anonymized version adds the run records and the annotators' case-level labels. Reproducing model outcomes also requires the source snapshots, including the SciFact full-text snapshot of the evaluated records, and the model configurations of Appendix D.